HITACHI
Inspire the Next

Hitachi Advanced Database Setup and Operation Guide

3000-6-501-H0(E)

Notices

■ Relevant program products

P-9W62-C411 Hitachi Advanced Data Binder version 05-01 (for Red Hat(R) Enterprise Linux(R) Server 6 (64-bit x86_64) and Red Hat(R) Enterprise Linux(R) Server 7 (64-bit x86_64))

P-9W62-C311 Hitachi Advanced Data Binder Client version 05-01 (for Red Hat(R) Enterprise Linux(R) Server 6 (64-bit x86_64) and Red Hat(R) Enterprise Linux(R) Server 7 (64-bit x86_64))

P-2462-C114 Hitachi Advanced Data Binder Client version 05-01 (for Windows 7, Windows 8.1, Windows 10, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, and Windows Server 2016)

This manual can be used for products other than the products shown above. For details, see the *Release Notes*.

Hitachi Advanced Data Binder is the product name of Hitachi Advanced Database in Japan.

■ Trademarks

HITACHI, HA Monitor, HiRDB, Job Management Partner 1 and JP1 are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

Access is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

AMD is a trademark of Advanced Micro Devices, Inc.

Excel is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Intel is a trademark of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

MSDN is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.

Red Hat Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.

UNIX is a trademark of The Open Group.

Visual Studio is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Windows is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Windows Server is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Other company and product names mentioned in this document may be the trademarks of their respective owners.

1. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

2. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

3. This product includes software written by Tim Hudson (tjh@cryptsoft.com).

4. This product uses OpenSSL Toolkit software in accordance with the OpenSSL License and Original SSLeay License, which are described as follows.

LICENSE ISSUES

==========

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit.
See below for the actual license texts. Actually both licenses are BSD-style
Open Source licenses. In case of any license issues related to OpenSSL
please contact openssl-core@openssl.org.

OpenSSL License

/* ==

* Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.

*

* Redistribution and use in source and binary forms, with or without

* modification, are permitted provided that the following conditions

* are met:

*

* 1. Redistributions of source code must retain the above copyright

* notice, this list of conditions and the following disclaimer.

*

* 2. Redistributions in binary form must reproduce the above copyright

* notice, this list of conditions and the following disclaimer in

* the documentation and/or other materials provided with the

* distribution.

*

* 3. All advertising materials mentioning features or use of this

* software must display the following acknowledgment:

* "This product includes software developed by the OpenSSL Project

* for use in the OpenSSL Toolkit. (http://www.openssl.org/)"

*

* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to

* endorse or promote products derived from this software without

* prior written permission. For written permission, please contact

* openssl-core@openssl.org.

*

* 5. Products derived from this software may not be called "OpenSSL"

* nor may "OpenSSL" appear in their names without prior written

* permission of the OpenSSL Project.

*

* 6. Redistributions of any form whatsoever must retain the following

* acknowledgment:

* "This product includes software developed by the OpenSSL Project

* for use in the OpenSSL Toolkit (http://www.openssl.org/)"

*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* ==
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

Original SSLeay License

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.

Hitachi Advanced Data Binder was developed based on the results of a leading-edge research and development effort called FIRST (*Funding Program for World-Leading Innovative R&D on Science and Technology*). In particular, Hitachi Advanced Data Binder was based on the FIRST research entitled *Development of the Fastest Database Engine for the Era of Very Large Database and Experiment and Evaluation of Strategic Social Services Enabled by the Database Engine*, spearheaded by Professor Kitsuregawa (Director General, National Institute of Informatics) of Tokyo University and conducted under the auspices of the Cabinet Office (CAO).

Professor Kitsuregawa and Associate Professor Goda, both of Tokyo University, also devised the out-of-order type execution principles used by this engine.

■ Microsoft product screen shots

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names:

Abbreviation			Full name or meaning
Windows	Windows 7	Windows 7 x86	Microsoft(R) Windows(R) 7 Professional (32-bit)
			Microsoft(R) Windows(R) 7 Enterprise (32-bit)
			Microsoft(R) Windows(R) 7 Ultimate (32-bit)

Abbreviation			Full name or meaning
		Windows 7 x64	Microsoft(R) Windows(R) 7 Professional (64-bit)
			Microsoft(R) Windows(R) 7 Enterprise (64-bit)
			Microsoft(R) Windows(R) 7 Ultimate (64-bit)
	Windows 8.1	Windows 8.1 x86	Windows(R) 8.1 Pro (32-bit)
			Windows(R) 8.1 Enterprise (32-bit)
		Windows 8.1 x64	Windows(R) 8.1 Pro (64-bit)
			Windows(R) 8.1 Enterprise (64-bit)
	Windows 10	Windows 10 x86	Windows(R) 10 Pro (32-bit)
			Windows(R) 10 Enterprise (32-bit)
		Windows 10 x64	Windows(R) 10 Pro (64-bit)
			Windows(R) 10 Enterprise (64-bit)
	Windows Server 2008 R2		Microsoft(R) Windows Server(R) 2008 R2 Standard
			Microsoft(R) Windows Server(R) 2008 R2 Enterprise
			Microsoft(R) Windows Server(R) 2008 R2 Datacenter
	Windows Server 2012		Microsoft(R) Windows Server(R) 2012 Standard
			Microsoft(R) Windows Server(R) 2012 Datacenter
	Windows Server 2012 R2		Microsoft(R) Windows Server(R) 2012 R2 Standard
			Microsoft(R) Windows Server(R) 2012 R2 Datacenter
	Windows Server 2016		Microsoft(R) Windows Server(R) 2016 Standard
			Microsoft(R) Windows Server(R) 2016 Datacenter

■ Restrictions

Information in this document is subject to change without notice and does not represent a commitment on the part of Hitachi. The software described in this manual is furnished according to a license agreement with Hitachi. The license agreement contains all of the terms and conditions governing your use of the software and documentation, including all warranty rights, limitations of liability, and disclaimers of warranty.

Material contained in this document may describe Hitachi products not available or features not available in your country.

■ Issued

Apr. 2020

■ Copyright

Preface

This manual describes how to design, set up, and operate Hitachi Advanced Database systems.

Note that, in this manual, and in the information output by the product (messages, command output results, and so on), *HADB* is often used in place of *Hitachi Advanced Database*.

■ Intended readers

This manual is intended for:

- System engineers who design and set up HADB systems, and system administrators
- Application developers

Readers of this manual must have:

- A basic knowledge of Linux or Windows system management
- A basic knowledge of RDBMS operation management
- A basic knowledge of SQL

■ Organization of this manual

This manual is organized into the following parts, chapters, and appendixes:

PART 1: Description

1. Overview

   This chapter provides an overview of HADB, describes its features, and explains system configurations.

2. Architecture

   This chapter provides an explanation about the logical structure of an HADB database, and an overview of DB areas, users, and schemas, as well as descriptions of the following items: the user authorization function, privileges, the database access processing method, transaction control, lock control, database recovery, the client-group facility, function for centrally managing client definitions, the background-import facility, the chunk archiving function, data retrieval from CSV files, text data retrieval, the audit trail facility, the multi-node function, and the memory structure of the HADB server.

PART 2: Exercise

3. Guide for Building a Hands-on Environment

   This chapter explains how to build a hands-on environment that will allow users to gain first-hand experience in installing and operating HADB.

PART 3: Design

4. System Design

   This chapter provides the workflows for designing the databases, resources, and server definitions necessary to configure HADB. It also explains how to estimate the amounts of disk space and memory that will be required.

5. Designing a Database

   This chapter explains how to design tables, indexes and DB areas, as well as how to estimate the sizes of DB areas.

B. Dictionary Tables

This appendix explains the information stored in dictionary tables, base tables that are locked when dictionary tables are referenced, the indexes defined for dictionary tables, and how to search for information in dictionary tables.

C. System Tables

This appendix explains the information stored in system tables, base tables that are locked when system tables are referenced, the indexes defined for system tables, and how to search for information in system tables.

D. Maximum and Minimum Values in HADB

This appendix provides the maximum and minimum values for system and databases settings.

E. Process That Starts When the HADB Server Starts

This appendix explains the process that is started when the HADB server starts.

■ Related publications

This manual is part of a related set of manuals. The manuals in the set are listed below (with the manual numbers):

- *Hitachi Advanced Database Application Development Guide* (3000-6-502(E))
- *Hitachi Advanced Database Command Reference* (3000-6-503(E))
- *Hitachi Advanced Database SQL Reference* (3000-6-504(E))
- *Hitachi Advanced Database Messages* (3000-6-505(E))
- *HA Monitor Cluster Software Guide (for Linux(R) (x86) Systems)* (3000-9-201(E))
- *Job Management Partner 1 Version 10 Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide* (3021-3-320(E))
- *JP1 Version 11 JP1/Base User's Guide* (3021-3-A01(E))

In references to Hitachi Advanced Database manuals, this manual uses *HADB* in place of *Hitachi Advanced Database*.

Example: *HADB Application Development Guide*

In references to the HA Monitor manual, this manual uses *HA Monitor for Linux(R) (x86)* in place of *HA Monitor Cluster Software Guide (for Linux(R) (x86) Systems)*.

Example: *HA Monitor for Linux(R) (x86)*

In references to the Job Management Partner 1/Automatic Job Management System 3 manual, this manual uses *Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide* in place of *Job Management Partner 1 Version 10 Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide*.

Example: *Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide*

In references to the JP1/Base manual, this manual uses *JP1/Base User's Guide* in place of *JP1 Version 11 JP1/Base User's Guide*.

Example: *JP1/Base User's Guide*

■ Conventions: Abbreviations for product names

This manual uses the following abbreviations for product names:

Abbreviation		Full name or meaning
HADB	HADB server	Hitachi Advanced Database
	HADB client	Hitachi Advanced Database Client
Linux	Linux	Linux(R)
	Red Hat Enterprise Linux Server 6	Red Hat(R) Enterprise Linux(R) Server 6 (64-bit x86_64)
	Red Hat Enterprise Linux Server 6 (64-bit x86_64)	
	Red Hat Enterprise Linux Server 7	Red Hat(R) Enterprise Linux(R) Server 7 (64-bit x86_64)
	Red Hat Enterprise Linux Server 7 (64-bit x86_64)	
HDLM		Hitachi Dynamic Link Manager Software
JP1/AJS3		Job Management Partner 1/Automatic Job Management System 3
JP1/Audit		JP1/Audit Management - Manager
Red Hat Enterprise Linux Server 6 (64-bit x86_64)		Red Hat(R) Enterprise Linux(R) Server 6 (64-bit x86_64)
Red Hat Enterprise Linux Server 7 (64-bit x86_64)		Red Hat(R) Enterprise Linux(R) Server 7 (64-bit x86_64)

■ Conventions: Acronyms

This manual also uses the following acronyms:

Acronym	Full name or meaning
APD	Application Parameter Descriptor
API	Application Programming Interface
ARD	Application Row Descriptor
BI	Business Intelligence
BLOB	Binary Large Object
BNF	Backus-Naur Form
BOM	Byte Order Mark
CLI	Call Level Interface
CLOB	Character Large Object
CPU	Central Processing Unit
CSV	Character-Separated Values
DB	Database

Acronym	Full name or meaning
DBMS	Database Management System
DMMP	Device Mapper Multipath
DNS	Domain Name System
ER	Entity Relationship
HBA	Host Bus Adapter
ID	Identification number
IEF	Integrity Enhancement Facility
IP	Internet Protocol
IPD	Implementation Parameter Descriptor
IRD	Implementation Row Descriptor
JAR	Java Archive File
JDBC	Java Database Connectivity
JDK	Java Developer's Kit
JNDI	Java Naming and Directory Interface
JRE	Java Runtime Environment
JTA	Java Transaction API
LOB	Large Object
LRU	Least Recently Used
LV	Logical Volume
LVM	Logical Volume Manager
MSDN	Microsoft Developer Network
NFS	Network File System
NIC	Network Interface Card
NTP	Network Time Protocol
ODBC	Open Database Connectivity
OS	Operating System
PP	Program Product
RAID	Redundant Array of Independent Disks
RDBMS	Relational Database Management System
TLB	Translation Lookaside Buffer
URL	Uniform Resource Locator
VG	Volume Group
WWN	World Wide Name

■ Conventions: Fonts and symbols

The following table explains the fonts used in this manual:

Font	Convention
Bold	**Bold** type indicates text on a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example: • From the **File** menu, choose **Open**. • Click the **Cancel** button. • In the **Enter name** entry box, type your name.
Italics	*Italics* are used to indicate a placeholder for some actual text to be provided by the user or system. For example: • Write the command as follows: `copy` *source-file target-file* • The following message appears: `A file was not found. (file =` *file-name*`)` *Italics* are also used for emphasis. For example: • Do *not* delete the configuration file.
`Code font`	A `code font` indicates text that the user enters without change, or text (such as messages) output by the system. For example: • At the prompt, enter `dir`. • Use the `send` command to send mail. • The following message is displayed: `The password is incorrect.`

The table below shows the symbols used in this manual for explaining commands and operands, such as the operands used in server definitions.

Note that these symbols are used for explanatory purposes only; do not specify them in the actual operand or command.

Symbol	Meaning	Example
`|`	In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR.	`adb_sql_text_out = {Y|N}` In this example, the vertical bar means that you can specify either `Y` or `N`.
`[ ]`	In syntax explanations, square brackets indicate that the enclosed item or items are optional.	`adbsql [-V]` In this example, the square brackets mean that you can specify `adbsql`, or you can specify `adbsql -V`.
`{ }`	In syntax explanations, curly brackets indicate that only one of the enclosed items is to be selected.	`adbcancel {--ALL|-u` *connection-ID*`}` In this example, the curly brackets mean that you can specify either `--ALL` or `-u` *connection-ID*.
`...`	In syntax explanations, an ellipsis (`...`) indicates that the immediately preceding item can be repeated as many times as necessary.	`adbbuff -n` *DB-area-name*`[, DB-area-name] ...` In this example, the ellipsis means that you can specify *DB-area-name* as many times as necessary.
`{{ }}`	In syntax explanations, double curly brackets indicate that the enclosed items can be repeated as a single unit.	`{{adbinitdbarea -n` *data-DB-area-name*`}}` In this example, the double curly brackets mean that you can specify `adbinitdbarea -n` *data-DB-area-name* as many times as necessary.

Symbol	Meaning	Example
X (underline)	In syntax explanations, underlined characters indicate a default value.	adb_import_errmsg_lv = {0|1} In this example, the underline means that the value 0 is assumed by HADB when the operand is omitted.
~	A swung dash indicates that the text following it explains the properties of the specified value.	adb_sys_max_users = *maximum-number-of-concurrent-connections* ~ <integer> ((1 to 1024)) <<10>> In this example, the text following the swung dash means that you can specify an integer in the range from 1 to 1024. If the operand is not specified, the value 10 is assumed by HADB.
< >	Single angle brackets explain the data type of the specified value.	
(( ))	Double parentheses indicate the scope of the specified value.	
<< >>	Double angle brackets indicate a default value.	

■ Conventions: Path names

- $INSTDIR is used to indicate the server directory path (for installation).
- $ADBDIR is used to indicate the server directory path (for operation).
- $DBDIR is used to indicate the DB directory path.
- %ADBCLTDIR% (for a Windows HADB client) or $ADBCLTDIR (for a Linux HADB client) is used to indicate the client directory path.
- %ADBODBTRCPATH% is used to indicate the folder path where HADB's ODBC driver trace files are stored.

■ Conventions: Symbols used in mathematical formulas

The following table explains special symbols used by this manual in mathematical formulas:

Symbol	Meaning
↑ ↑	Round up the result to the next integer. Example: The result of ↑34 ÷ 3↑ is 12.
↓ ↓	Discard digits following the decimal point. Example: The result of ↓34 ÷ 3↓ is 11.
MAX	Select the largest value as the result. Example: The result of MAX(3 × 6, 4 + 7) is 18.
MIN	Select the smallest value as the result. Example: The result of MIN(3 × 6, 4 + 7) is 11.

■ Conventions: Syntax elements

Syntax element notation	Meaning
<path name>	The following characters can be used in path names: • In Linux Alphanumeric characters, hash mark (#), hyphen (-), forward slash (/), at mark (@), and underscore (_) • In Windows

Syntax element notation	Meaning
	Alphanumeric characters, hash mark (#), hyphen (-), forward slash (/), at mark (@), underscore (_), backslash (\), and colon (:) Note, however, that the characters that can be used might differ depending on the operating system.
<OS path name>	For an OS path name, all characters that can be used in a path name in the operating system can be used. For details about available characters, see the documentation for the operating system you are using.
<character string>	Any character string can be specified.
<integer suffixed by the unit>	Specify the value in a format consisting of a numeric character (in the range from 0 to 9) followed by a unit (MB (megabyte), GB (gigabyte), or TB (terabyte)). Do not enter a space between the numeric character and the unit. • Examples of correct specification 1024MB 512GB 32TB • Example of specification that causes an error 512 GB

■ Conventions: KB, MB, GB, TB, PB, and EB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is $1{,}024^2$ bytes.
- 1 GB (gigabyte) is $1{,}024^3$ bytes.
- 1 TB (terabyte) is $1{,}024^4$ bytes.
- 1 PB (petabyte) is $1{,}024^5$ bytes.
- 1 EB (exabyte) is $1{,}024^6$ bytes.

■ Conventions: Version numbers

The version numbers of Hitachi program products are usually written as two sets of two digits each, separated by a hyphen. For example:

- Version 1.00 (or 1.0) is written as 01-00.
- Version 2.05 is written as 02-05.
- Version 2.50 (or 2.5) is written as 02-50.
- Version 12.25 is written as 12-25.

The version number might be shown on the spine of a manual as *Ver. 2.00*, but the same version number would be written in the program as *02-00*.

Contents

Appendixes 1751

1 Overview

Hitachi Advanced Database (HADB) is a database management system (DBMS) that can manage a large volume of data and enables high-speed data retrieval. This chapter provides an overview of HADB, describes its features, and explains system configurations.

1.1 HADB for managing data in the information explosion age

HADB is a DBMS that allows a user to build a large-scale relational database and retrieve the desired data at high speeds.

Social infrastructures have undergone radical changes in recent years, as exemplified by the spread of cloud computing. This has resulted in an explosive growth in the volume of data kept by corporations and society. The widespread use of large-capacity storage media is making this trend even more prominent.

These changes in the social infrastructures have been accompanied by a rapid increase in the volume of data that must be managed by DBMSs. As a result, existing DBMSs might no longer be able to provide satisfactory retrieval performance. How to manage the burgeoning volume of data is becoming a critical issue. At the same time, the level of interest in how to provide services that utilize this massive amount of data is also increasing.

Because of these factors, DBMSs must now provide the following features:

- Ability to manage massive amounts of data, in the petabyte or even exabyte range
- Ability to perform high-speed retrieval of desired information from massive amounts of data

HADB is a DBMS developed to satisfy these requirements.

HADB is designed to manage petabytes or even exabytes of data. Using HADB allows you to build a database capable of managing a massive amount of data. Additionally, HADB uses a retrieval method that effectively utilizes advanced hardware capabilities (out-of-order execution), making it possible to retrieve data at high speeds from a large volume of data.

Note

One petabyte is approximately 1,000 terabytes. One exabyte is approximately 1,000 petabytes.

1.2 HADB features

This subsection describes the features of HADB.

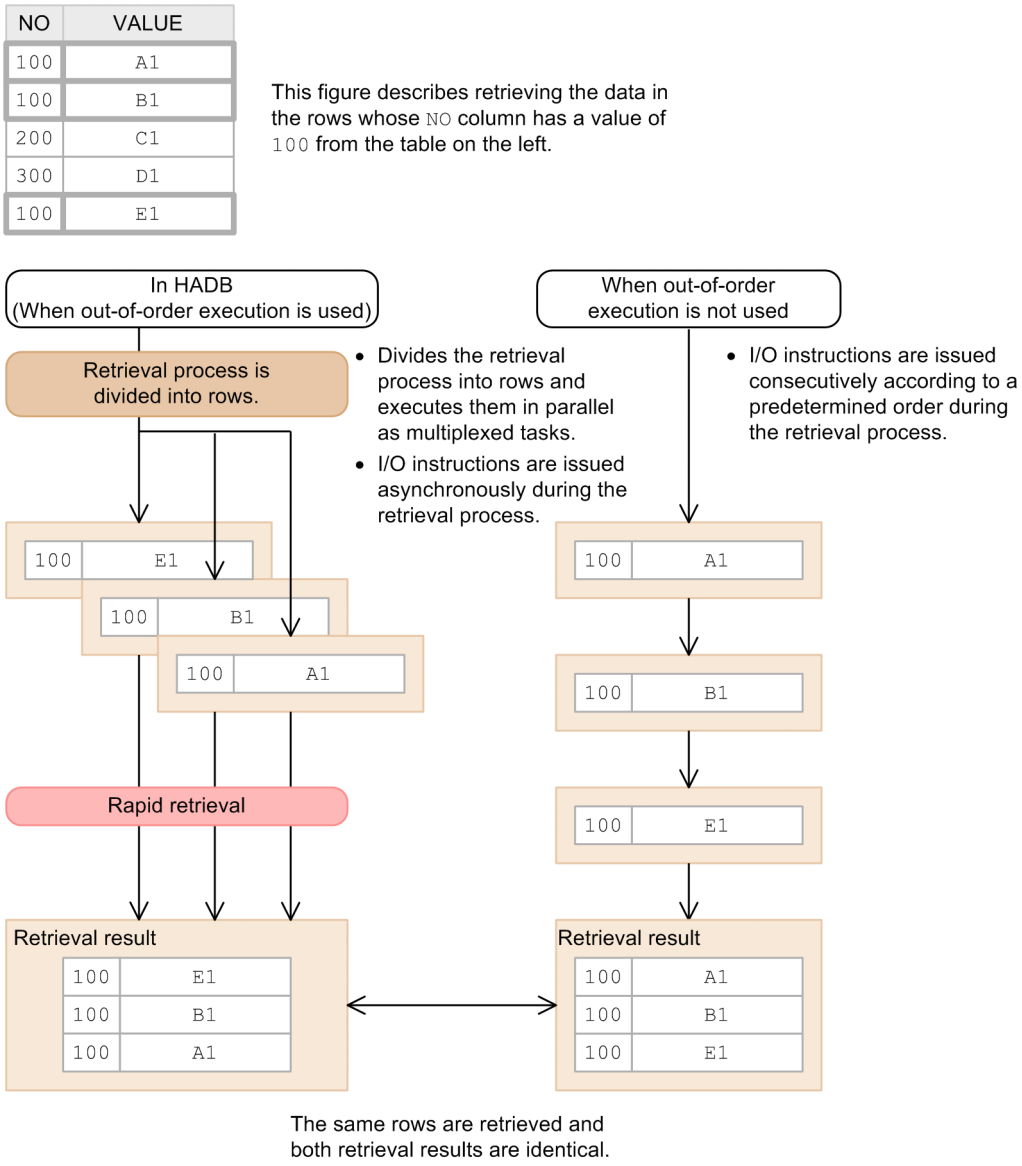
1.2.1 High-speed retrieval of data from a large volume of data

HADB allows you to execute complex retrieval operations using SQL on a large volume of data.

HADB uses a retrieval method called *out-of-order execution*, which is based on the out-of-order execution principle. This method breaks the retrieval process into rows, assigns those rows to multiple tasks, and then executes these tasks in parallel. Therefore, HADB can execute a large number of tasks in parallel, unlike DBMSs that do not use out-of-order execution. This provides for very high-speed data retrieval.

The following figure provides an overview of retrieval processing when out-of-order execution is used.

Figure 1-1: High-speed retrieval when out-of-order execution is used



The out-of-order execution method was developed based on the relational database execution principle, which states, "retrieval processing consists of set operations on rows with no rules governing the sequence in which retrieval results are returned by a DBMS, except for sort processes."

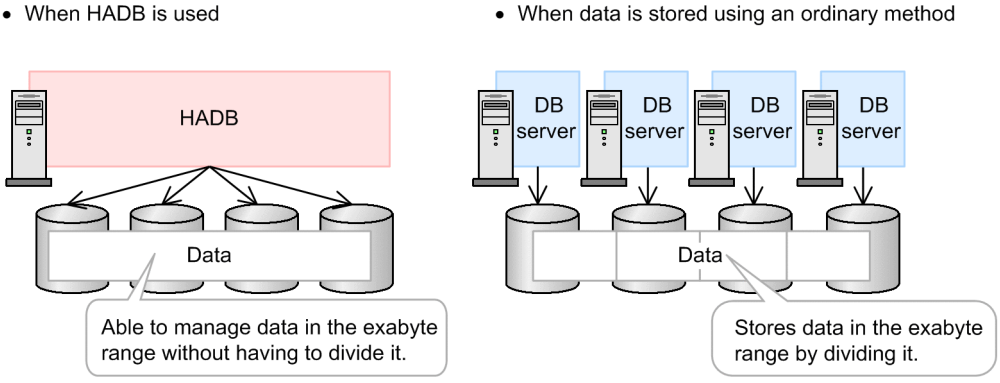
1.2.2 Capability to build a large-scale database (up to exabytes)

HADB allows you to build a database for managing massive amounts of data in the exabyte range.

To manage a large volume of data, data is normally divided into smaller segments and stored in separate areas. However, this means that the data must be divided and stored elsewhere if the volume of data increases. Moreover, storing all the pieces of divided data creates more overhead.

HADB can create a single logical area that spans multiple disks. By storing tables in this area, you can manage up to exabytes of data using a single table, without having to divide the data. There is no need to divide and store the data elsewhere, even if the volume of data increases. The following figure provides a conceptual view of the structure of such a large-scale database.

Figure 1-2: Structure of a large-scale database capable of handling petabytes or even exabytes of data



HADB also provides other facilities, such as the data import facility to help in creating a large-scale database.

1.2.3 Scale-out for load distribution (Multi-node function)

One database can be shared by multiple HADB servers, which enables multiple HADB servers to coordinate the processing. This makes it possible to use the scale-out technique to realize the load distribution. Server machines can be added as necessary afterwards, which allows for scaling out based on the expansion of business operations.

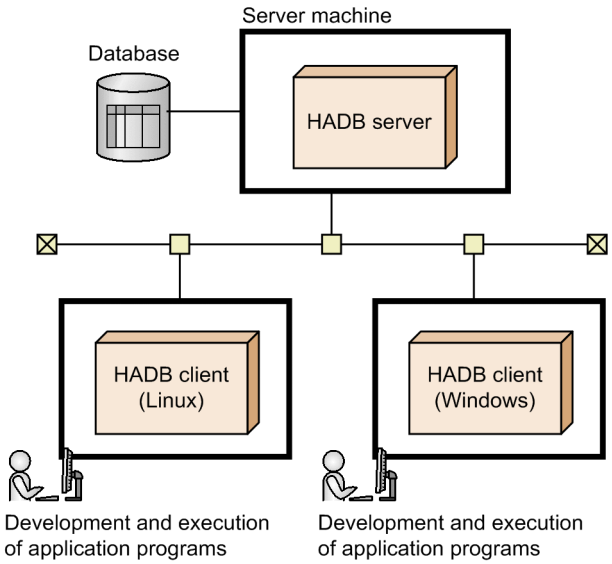
Note

The function that executes processing by linking multiple HADB servers is called the *multi-node function*.

1.3 HADB system configuration

HADB is installed in a client-server network environment. The following figure shows the system configuration of HADB.

Figure 1-3: HADB system configuration example



1.3.1 HADB server

The HADB server resides on a server machine and manages the HADB database. It performs tasks such as controlling transactions executed on the database. You can also develop and execute application programs on the HADB server.

The HADB server is compatible with the following OS:

- Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  The HADB server runs on version 6.2 or later.
- Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  The HADB server runs on version 7.1 or later.

Note that HADB runs only on the Intel 64 architecture. It does not run on the AMD 64 architecture.

1.3.2 HADB client

A client system connected to the HADB server is called an *HADB client*. You can develop and execute applications on an HADB client.

You can use an HADB client under Linux or Windows. For details about how to use an HADB client, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

2 Architecture

This chapter explains the logical structure of an HADB database and how database access is processed.

2.1 Types of tables

HADB manages data using tables.

As in other relational databases, tables in HADB consist of rows and columns. The horizontal and vertical directions of a table are referred to as rows and columns, respectively. The data in a particular column is always in the same format for all rows.

Each row consists of one or more columns. A row is the basic unit used for performing operations on a table.

Tables can be classified into base tables and viewed tables. The following subsections explain these in detail.

2.1.1 Base tables

A base table is a table that contains data that is actually stored in a database.

The two basic types of base tables are row store tables and column store tables, each of which stores data in a different format. These tables can be further categorized into single-chunk tables in which only one chunk can be created, and multi-chunk tables in which multiple chunks can be created. When defining a base table, you select whether to define it as a row store table or column store table, and also whether to make it a single or multi-chunk table. This means that a base table you define will be one of the following types:

- Single-chunk row store table
- Multi-chunk row store table
- Single-chunk column store table
- Multi-chunk column store table

For details about the features of row store tables, column store tables, single-chunk tables, and multi-chunk tables, see 2.2 Base table types.

2.1.2 Viewed tables

A virtual table created by defining the result of the query expression specified in the `CREATE VIEW` statement as a new table is called a *viewed table*. For example, you can define the result of a set operation for multiple tables as a viewed table, or can define the specific rows or columns in a table as a viewed table.

By predefining a viewed table for the results of a conditioned search that will be performed many times, you can simplify the database retrieval operation. The following figure shows an example of a viewed table.

Figure 2-1: Example of a viewed table

■ STOCK (base table)

PNO (Product No.)	PCODE (Product code)	PNAME (Product name)	PRICE (Unit price)	QUANTITY (Inventory quantity)
01010	101	Blouse	3500	62
01011	101	Blouse	3500	85
02021	202	Shirt	3640	67
03530	353	Shirt	4760	18
03531	353	Shirt	4760	26
04121	412	Sweater	8400	8
05910	591	Socks	300	300
05911	591	Socks	300	90

■ VSTOCK (view table)

PCODE	PRICE	QUANTITY
202	3640	67
353	4760	18
353	4760	26

Explanation

The VSTOCK viewed table is created from the STOCK table, which is the base table. The VSTOCK viewed table consists of the columns PCODE (product code), PRICE (unit price), and QUANTITY (inventory quantity), and contains only the rows whose entry in the PNAME (product name) column is Shirt.

Note

Defining a viewed table has advantages like the following in terms of security:

- By defining a viewed table that allows only specific columns or rows in the table to be referenced, you can hide data that is irrelevant and data to which access must be restricted.
- Because there is no need to notify database users of the table name of the base table that actually stores the data, the use of viewed tables has the effect of preventing unauthorized access using base table names. This applies when you create a viewed table whose underlying table is a viewed table.

You can allow database users to reference data by directly referencing the base table in which the data is actually stored. However, if you want to allow users to only reference certain data within the table (such as a specific column), we recommend that you use viewed tables.

(1) Updatable viewed tables and read-only viewed tables

Viewed tables are categorized into the following two types. A viewed table can be defined as an updatable viewed table or read-only viewed table.

- **Updatable viewed table**

  You can insert, update, and delete rows for this type of viewed table, in the same way as for a base table. However, you cannot add columns and define indexes.

  When you perform a row insertion, update, or deletion operation for an updatable viewed table, the same operation (row insertion, update, or deletion) is also performed for the underlying table of the updatable viewed table.

- **Read-only viewed table**

You cannot insert, update, and delete rows for this type of viewed table. You cannot add columns and define indexes, either.

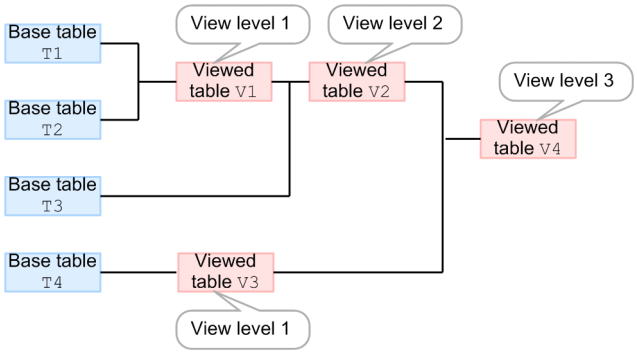
(2) About underlying tables of viewed tables

The table that underlies a viewed table is called an underlying table. The table specified in the query expression of a `CREATE VIEW` statement becomes the underlying table.

- What is a view level?

  A value that indicates the hierarchical depth of a defined viewed table from the base table. The following figure shows an example of view levels.

  Figure 2-2: Example of view levels



  Explanation

  - The view level of viewed table `V1` is 1 because only base tables underlie `V1`.
  - The view level of viewed table `V2` is 2 because a base table and a viewed table at view level 1 underlie `V2`.
  - The view level of viewed table `V3` is 1 because only a base table underlies `V3`.
  - The view level of viewed table `V4` is 3 because the underlying tables of `V4` are at view levels 1 and 2.

> **Note**
>
> - Dictionary tables and system tables are treated as viewed tables at view level 1.
> - Derived tables that are derived by table value constructors are treated the same as base tables.

(3) Invalidating viewed tables

If you perform any of the following operations for a base table or viewed table, all viewed tables that are dependent on the operation-target table are invalidated.

1. Using the `DROP TABLE` statement (without specifying drop behavior) to delete a base table

   In this case, all viewed tables that were dependent on that table (deleted base table) are invalidated.
2. Using the `DROP VIEW` statement (without specifying drop behavior) to delete a viewed table

   In this case, all viewed tables that were dependent on that table (deleted viewed table) are invalidated.
3. Using the `ALTER TABLE` statement to rename a column of a base table

   In this case, all viewed tables that were dependent on that table (base table in which a column was renamed) are invalidated.
4. Using the `ALTER TABLE` statement to change a regular multi-chunk table into an archivable multi-chunk table

In this case, all viewed tables that were dependent on that table (table that was changed to an archivable multi-chunk table) are invalidated.

5. Using the ALTER TABLE statement to change an archivable multi-chunk table into a regular multi-chunk table

   In this case, all viewed tables that were dependent on that table (table that was changed to a regular multi-chunk table) are invalidated.

6. Using the ALTER VIEW statement to re-create a viewed table

   In this case, all viewed tables that were dependent on the viewed table that was re-created are invalidated.

7. Using the REVOKE statement to revoke the SELECT privilege for a table#

   In this case, all viewed tables that were dependent on that table (table for which the SELECT privilege was revoked) are invalidated.

   #

   - This is the case where both the SELECT privilege granted by specifying an authorization identifier and the SELECT privilege granted by specifying PUBLIC are revoked.
   - Viewed tables might be invalidated when a SELECT privilege with the grant option is revoked, or only the grant option of a SELECT privilege is revoked. If the grant option used to grant the SELECT privilege to another HADB user is revoked, that SELECT privilege will also be revoked. If the table for which the SELECT privilege was revoked is an underlying table, the viewed tables that depend on that underlying table are invalidated. For specific examples, see *Examples* under *Revoking access privileges* in the manual *HADB SQL Reference*.

     For details about grant options, see (3) Granting access privileges under 2.7.5 Access privileges.

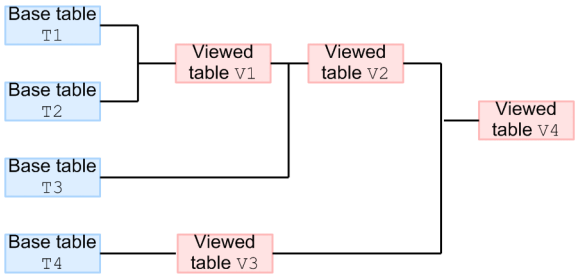
- **What are invalidated viewed tables?**

  If a viewed table that can be accessed becomes inaccessible, we say that the viewed table was invalidated. If you want to make an invalidated viewed table accessible again, you must resolve the issue that caused the viewed table to be invalidated, and then use an ALTER VIEW statement to re-create the viewed table. Alternatively, after resolving the issue that caused the viewed table to be invalidated, you can use a DROP VIEW statement to delete the viewed table, and then redefine the viewed table using a CREATE VIEW statement.

- **What are dependent viewed tables?**

  The viewed tables that are affected by the preceding seven operations are called dependent viewed tables. The following figure shows an example of dependent viewed tables.

Figure 2-3: Example of dependent viewed tables



Explanation

- Viewed tables V1, V2, and V4 are dependent on base table T1.
- Viewed tables V1, V2, and V4 are dependent on base table T2.
- Viewed tables V2 and V4 are dependent on base table T3.
- Viewed tables V3 and V4 are dependent on base table T4.
- Viewed tables V2 and V4 are dependent on viewed table V1.

- Viewed table `V4` is dependent on viewed table `V2`.
- Viewed table `V4` is dependent on viewed table `V3`.
- There is no dependent viewed table for viewed table `V4`.

2.2 Base table types

This section describes the types of base tables (row store, column store, single-chunk, and multi-chunk).

2.2.1 Row store tables and column store tables

The two basic types of base tables are row store tables and column store tables, each of which stores data in a different format. You can expect to see an improvement in search performance by using the table type that is appropriate in terms of the intended use of the base table and the operation to which it relates.

(1) What are row store tables and column store tables?

There are two table-data storage formats: row store format and column store format. You can specify the table-data storage format when defining the table. A table defined with row store format as the table-data storage format is called a *row store table*. A table defined with column store format as the table-data storage format is called a *column store table*.

Row store tables store data in row store format. Column store tables store data in column store format. However, if the `INSERT` or `UPDATE` statement is executed for a column store table, the added or updated data is stored in the column store table in row store format.

■ Relationship between the column store table and table-data storage format

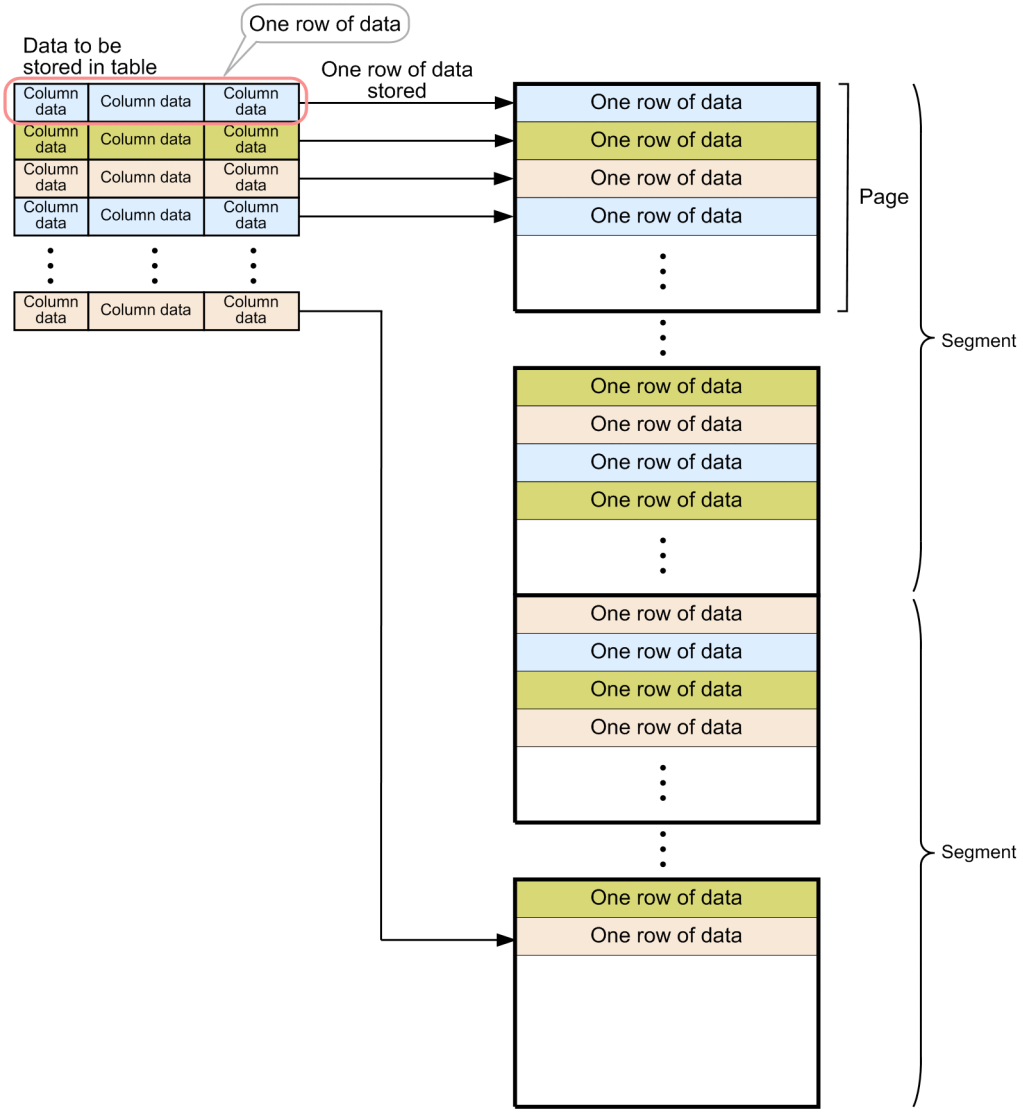
- If the `adbimport` command is used to import data into a column store table, the data is stored in the table in column store format.
- If the `INSERT` or `UPDATE` statement is used to add or update data for a column store table, the data is stored in the table in row store format.

> **Note**
>
> All tables defined in versions of HADB server earlier than **04-01** are row store tables.

(2) What is row store format?

A format in which data is stored in the database at the row level is called *row store format*. In row store format, one row of data is stored in the database as one record. The following figure shows how data is stored in row store format:

Figure 2-4: Data storage in row store format



Explanation

- One row of data is stored in the database as one record.
- A maximum of 255 rows of data can be stored in one page.

> **Note**
>
> For details about pages and segments, see 2.4.3 DB area structure (segments and pages).

■ Retrieval methods that benefit from row store tables

Because data is stored at the row level, row store tables are suited to the following manners of data retrieval:

- Data retrieval that accesses data at the row level

  One example of this manner of data retrieval is executing a `SELECT` statement with `*` specified in the selection expression, as follows:

  ```
  SELECT * FROM "T1" WHERE "C1">=DATE'2017-09-06'
  ```

  Row store format remains suitable when the selection expression specifies almost every column.

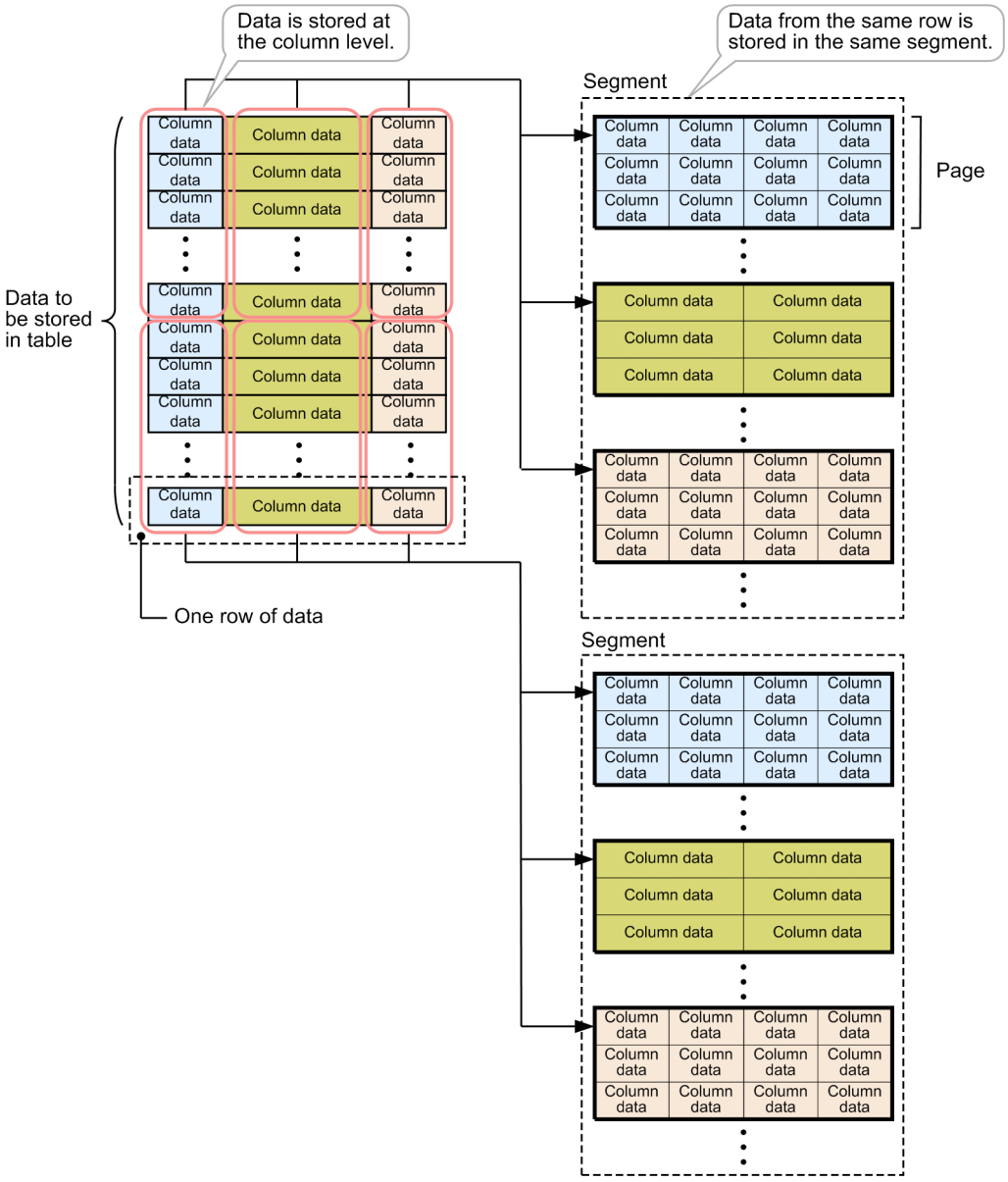
- When retrieving data in a manner that narrows down the search range using B-tree indexes

  When a table search is executed, the HADB server only reads the pages that store rows that are in the search range. Therefore, using a B-tree index to narrow down the search range can reduce the number of pages that need to be loaded.

(3) What is column store format?

A format in which data is stored in the database at the column level is called *column store format*. In column store format, the data in each column of a table is stored together in the database. The following figure shows how data is stored in column store format:

Figure 2-5: Data storage in column store format



Explanation

- The data in each column is stored in the database at the column level.
- Data from the same row is stored in the same segment.

  A maximum of 262,144 rows of data for a given column can be stored in one segment.

> **Note**
>
> For details about pages and segments, see 2.4.3 DB area structure (segments and pages).

■ Retrieval methods that benefit from column store tables

Because data is stored at the column level, column store tables are suited to the following manners of data retrieval:

- Searching the data in a specific column in its entirety without significantly narrowing the search range
- Accessing the data in a specific column within a specific range (such as a specific month or year)
- Performing operations that involve frequent calculations (such as computing averages and totals) in relation to the value data in a specific column
- Grouping data in specific columns on a frequent basis

When a table search is executed, the HADB server first finds the segment that stores the column data targeted by the search. The HADB server then reads the pages within that segment that store the column data targeted by the search. When performing operations that involve calculations in relation to the value data in a specific column, the HADB server only needs to access the pages that store the target column data. This allows the number of pages that need to be read to be reduced.

■ Column-data compression types

When data is imported into a column store table, the data in each column is compressed as it is stored in the table. There are several column-data compression types. For details about each column-data compression type, see (4) Column-data compression types for column store tables in 5.2.2 Criteria for selecting row store tables and column store tables.

> **Note**
>
> When importing data into a column store table, HADB server automatically selects the compression type based on the data being imported. You can also specify the column-data compression type when defining the column store table.

■ Relationship between deletion of data in column store format and invalid row information pages

If the `DELETE` statement is executed for data that is stored in a column store table in column store format, the data is invalidated. The data is not deleted from the disk.

The HADB server uses *invalid row information pages* to manage the information that indicates specific data has been invalidated. Each time the `DELETE` statement is executed for a column store table, an invalid row information page is allocated and free space in the data DB area decreases by the size of that page.

Therefore, if the `DELETE` statement is executed for data in column store format when free space in the data DB area is insufficient, an error might occur.

■ If the INSERT or UPDATE statement is executed for a column store table

If the `INSERT` or `UPDATE` statement is executed for a column store table, the added or updated data is stored in the column store table in row store format. If data is stored in the column store table in row store format, characteristics of the column store table might be adversely affected. For example, the retrieval performance might be degraded at the time of access to specific column data, or the data compression rate might be lowered. Therefore, in situations where the `INSERT` or `UPDATE` statement might be executed for a column store table, we recommend that you enable the *updated-row columnizing facility*. If the updated-row columnizing facility is enabled and data is stored in row store format in a column store table, HADB automatically converts the data into column store format. For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).

2.2.2 Single-chunk tables and multi-chunk tables

Base table can be classified into single-chunk tables and multi-chunk tables, with each type allowing the use of different functionality. This section explains single-chunk tables and multi-chunk tables, and describes the functionality that can be used with each type of table.

(1) What is a single-chunk table?

A single-chunk table is a base table in which only one chunk can be created. The background-import facility cannot be used with single-chunk tables.

For details about chunks and the background-import facility, see 2.14 Background-import facility.

(2) What is a multi-chunk table?

A multi-chunk table is a base table in which multiple chunks can be created. The background-import facility can be used with multi-chunk tables.

Multi-chunk tables can be categorized into two types: regular multi-chunk tables and archivable multi-chunk tables.

- **Regular multi-chunk table**
  For regular multi-chunk tables, the background-import facility, which is a basic function for multi-chunk tables, can be used, but the chunk archiving function cannot be used.
- **Archivable multi-chunk table**
  For archivable multi-chunk tables, both the background-import facility and chunk archiving function can be used.

For details about the chunk archiving function, see 2.15 Chunk archiving function (compressing data in a chunk).

Important

The available functions differ between single and multi-chunk tables. The following table shows the relationship between the table type and the available functions:

Table 2-1: Relationship between the base table types and functions that can be used

Base table type		Function that can be used	
		Background-import facility	Chunk archiving function
Single-chunk table		N	N
Multi-chunk table	Regular multi-chunk table	Y	N
	Archivable multi-chunk table	Y	Y

Legend:

Y: Can be used.

N: Cannot be used.

(3) Relationship between row store tables and column store tables

When designing a table, the first step is to decide whether to define it as a row store table or column store table. You then decide whether to define it as a single-chunk table or multi-chunk table (regular multi-chunk table or archivable multi-chunk table). Note that you cannot define a column store table as an archivable multi-chunk table.

The following table shows the relationship between row and column store tables and single and multi-chunk tables.

Table 2-2: Relationship between row and column store tables and single and multi-chunk tables

Base table type	Base table type		
	Single-chunk table	Multi-chunk table	
		Regular multi-chunk table	Archivable multi-chunk table
Row store table	Y	Y	Y
Column store table	Y	Y	N

Legend:

Y: You can define a table of this type.

N: You cannot define a table of this type.

2.3 Index types

There are three types of indexes: B-tree indexes, text indexes, and range indexes. An explanation of each type of index follows.

2.3.1 B-tree indexes

In a B-tree index, the key values are managed in a B-tree structure. Use of a B-tree index for retrievals from a table enables you to directly access the rows that satisfy the search condition. This improves table retrieval performance.

2.3.2 Text indexes

Text indexes are used for retrieving character string data that contains character strings specified as the retrieval criteria in SQL statements. A text index manages, in pages, information about the positions of data items (character strings) that are stored in columns in the base table for which the text index has been defined.

If one of the following items is specified in the retrieval criteria in an SQL statement that retrieves data from a table, use of a text index reduces the number of pages to be loaded. This improves table retrieval performance.

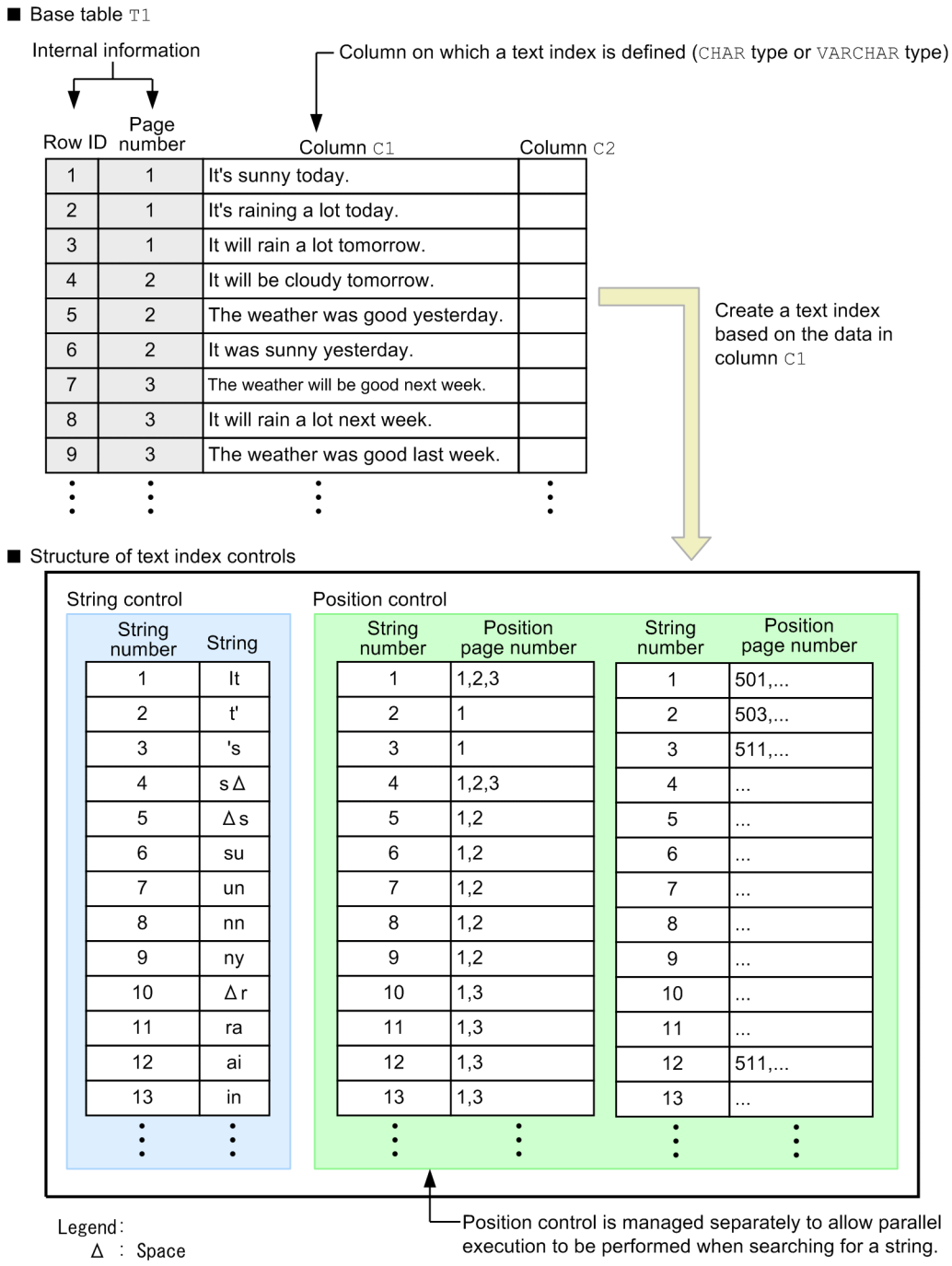
- `LIKE` predicate
- `LIKE_REGEX` predicate
- Scalar function `CONTAINS`

A text index consists of the following two types of controls:

- **String control**
  String control manages text-indexed character strings.
- **Position control**
  Position control manages the positions of text-indexed character strings.

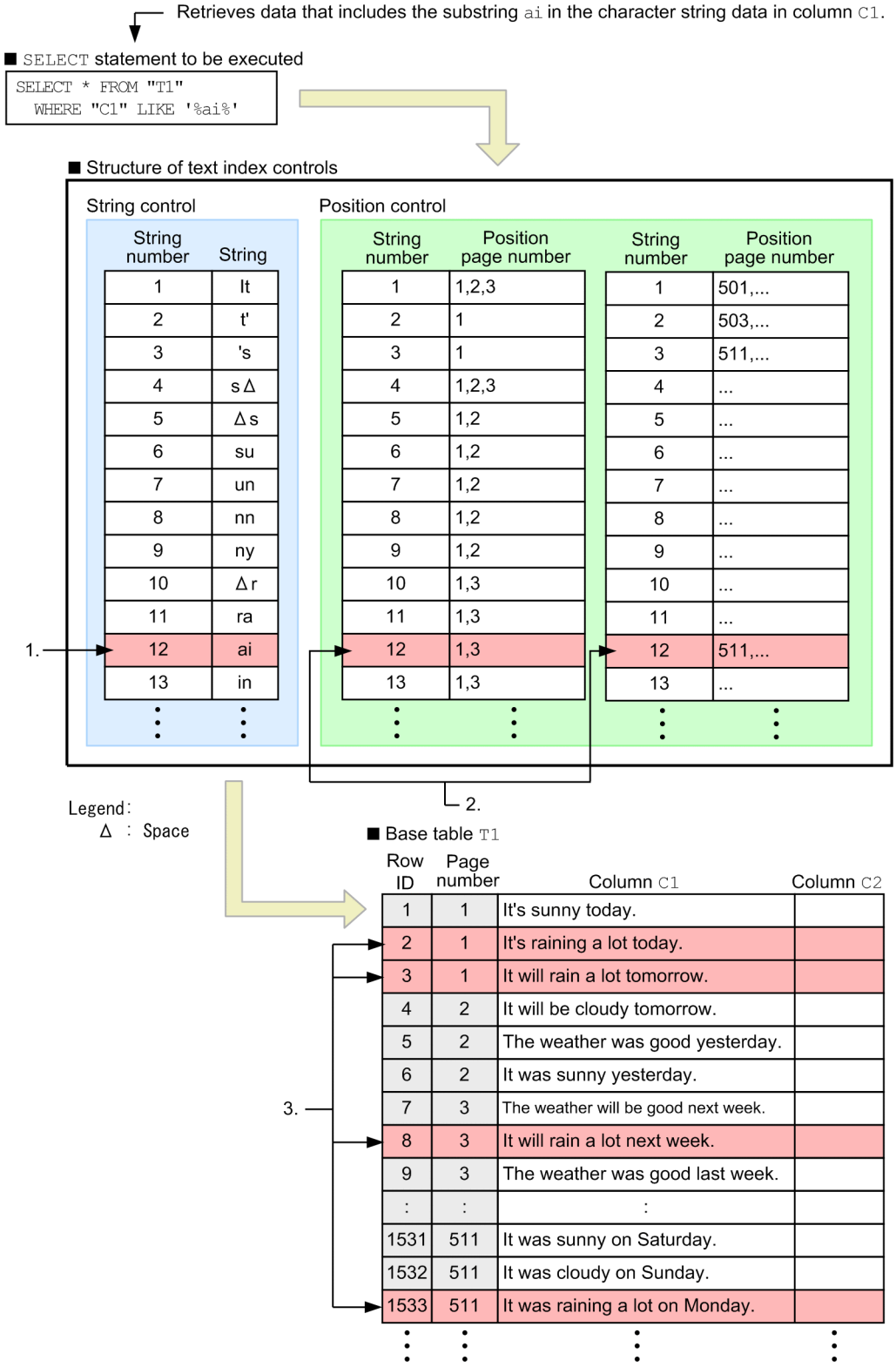
The following figure shows the structure of the text index controls.

Figure 2-6: Structure of text index controls



The following figure shows the general procedure for using a text index to retrieve data from a base table.

Figure 2-7: Overview of using a text index for retrieval processing



Explanation

A base table is retrieved using a text index with the following procedure:

1. Searches for the search string specified in the `LIKE` predicate (`ai`) from the string control. Then obtains the corresponding string number (`12`).

2. Retrieves the corresponding position page numbers (`1`, `3` and `511`) from the position control by using the character string number (`12`) retrieved from the string control as the key value.
3. Imports to base table `T1` the data pages that correspond to the position page numbers (`1`, `3` and `511`) in the position control. Then retrieves the corresponding row IDs (`2`, `3`, `8` and `1533`) to obtain the table data. Data pages that do not correspond to the position page numbers are not imported.

> **Note**
>
> If a B-tree index and a text index are both defined for a column specified as a retrieval criterion, only the B-tree index or only the text index is used. However, depending on the specified retrieval criteria, neither of the indexes might be used.

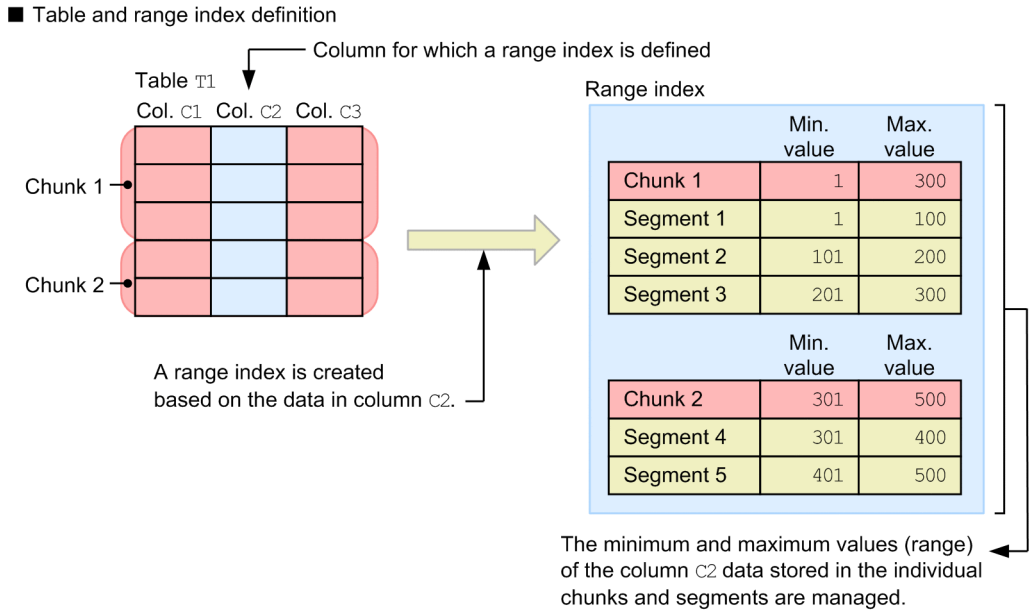
2.3.3 Range indexes

A range index manages the following two ranges (the minimum and maximum values of a column's data) for the data stored in the base table for which the range index is defined:

- **Range of data stored in a chunk**
- **Range of data stored in a segment**

The following figure provides an overview of the ranges that are managed by a range index.

Figure 2-8: Overview of the ranges managed by a range index



When a range index is used, retrieval processing of chunks that store ranges of data that do not satisfy the search condition is skipped (chunks can be skipped).

After those chunks are skipped, retrieval processing of segments that store ranges of data that do not satisfy the search condition, within the chunks that store ranges of data that satisfy the search condition, is also skipped (segments can be skipped).

Skipping retrieval processing of unnecessary chunks and segments improves overall retrieval performance.

- **Chunk skipping**
  This refers to skipping retrieval processing of chunks storing ranges of data that do not satisfy the search condition.
  For an overview of chunks, see 2.14.2 Managing data in data-import units (chunks). For details about chunk skipping, see 2.14.3 Relationship between chunks and range indexes.
- **Segment skipping**
  This refers to skipping retrieval processing of segments storing ranges of data that do not satisfy the search condition.
  For an overview of segments, see (1) Segment in 2.4.3 DB area structure (segments and pages). The details about segment skipping are described below.

Note

■ Relationship between a range index and out-of-order execution

An SQL statement that is executed by out-of-order execution first reads the range of data stored in the chunks managed with a range index at the start of retrieval processing (when the cursor opens). Then, the statement stores the range in memory. During retrieval, the statement determines whether to perform chunk skipping based on the information stored in memory.

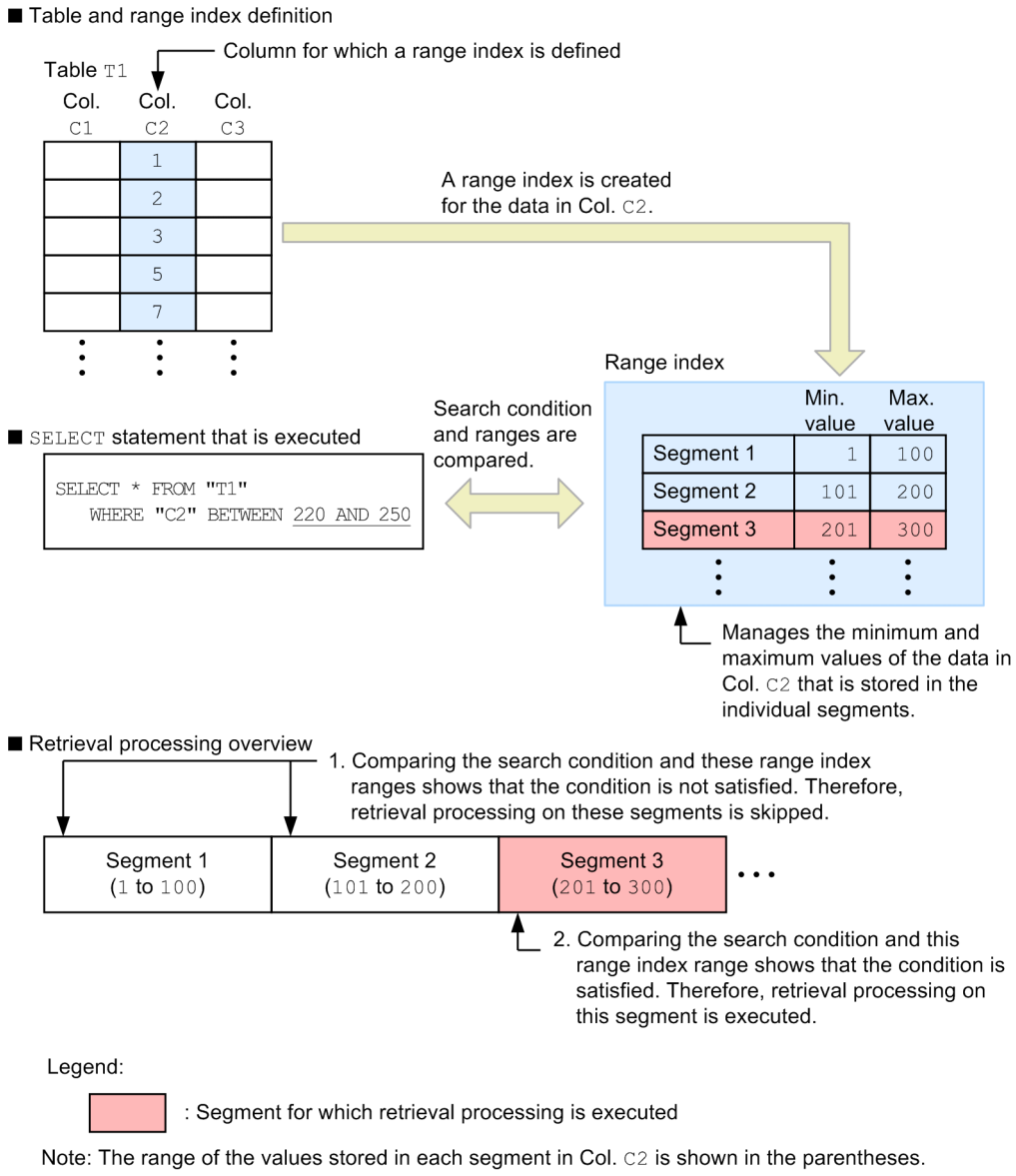
Note that this processing occurs only if the range index meets all of the following conditions:

- The range index is defined in the table subject to retrieval performed by using a B-tree index.
- The range index is used for chunk skipping.

(1) Range index overview

Using segment skipping as an example, the following figure provides an overview of range indexes.

Figure 2-9: Range index overview (segment skipping)



Note: The range of the values stored in each segment in Col. C2 is shown in the parentheses.

(2) Range index characteristics

A range index has the four major characteristics described below. For the characteristics of range indexes that can be used for skipping chunks, see (2) Range index characteristics (skipping of chunks) in 2.14.3 Relationship between chunks and range indexes.

(a) Using B-tree indexes or text indexes together with range indexes

After chunk skipping is performed by using a range index, a B-tree index or text index can be used. For details about retrieval using a range index together with a B-tree index or text index, see (a) Using B-tree indexes or text indexes together with range indexes in (2) Range index characteristics (skipping of chunks) in 2.14.3 Relationship between chunks and range indexes.

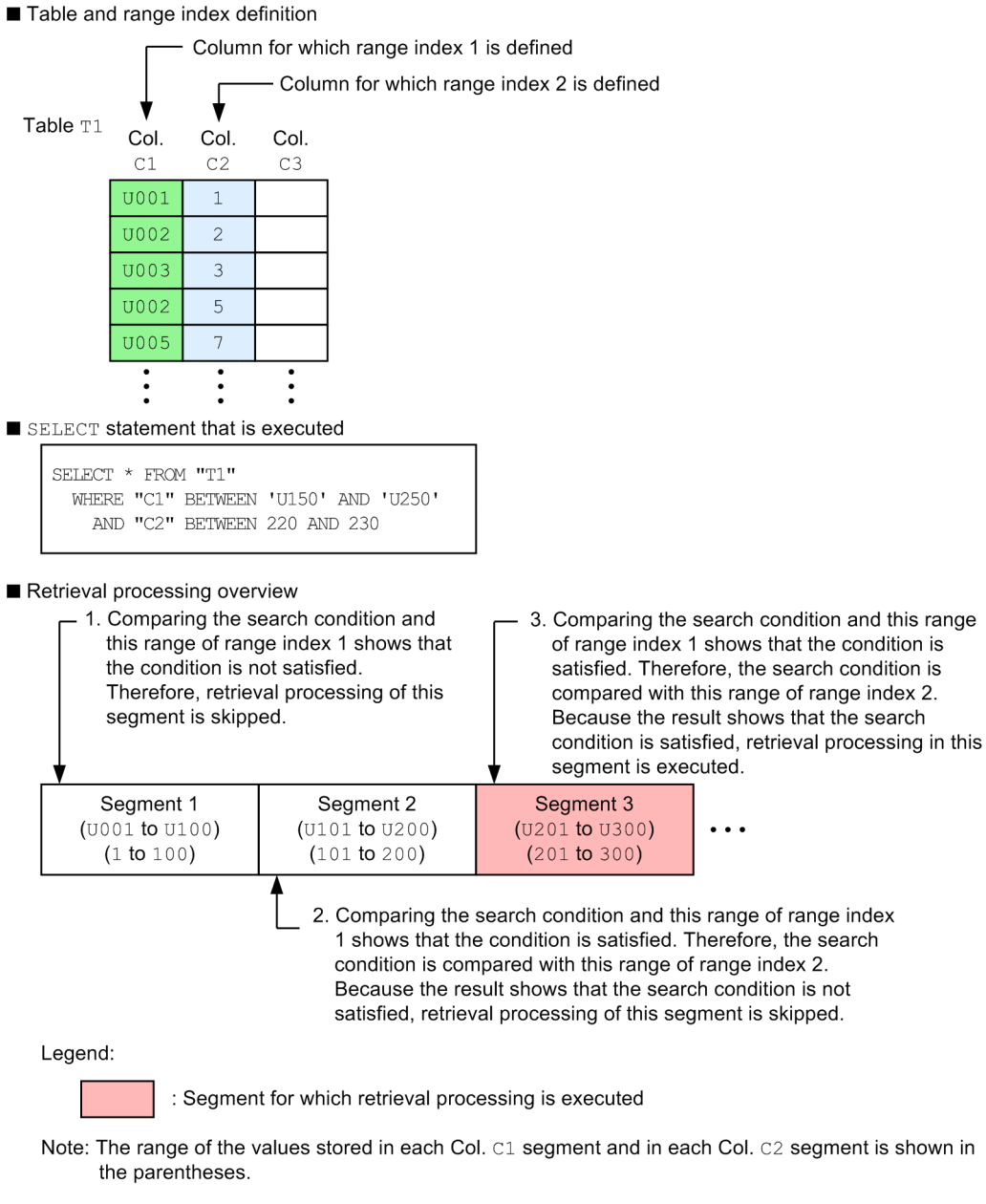
Note

If a range index is used together with a B-tree index or text index, retrieval using a range index (segment skipping) is not performed.

(b) Multiple range indexes can be used

The following figure provides an example of a retrieval that uses multiple range indexes.

Figure 2-10: Example of a retrieval that uses multiple range indexes



(c) Smaller data size than a B-tree index or a text index

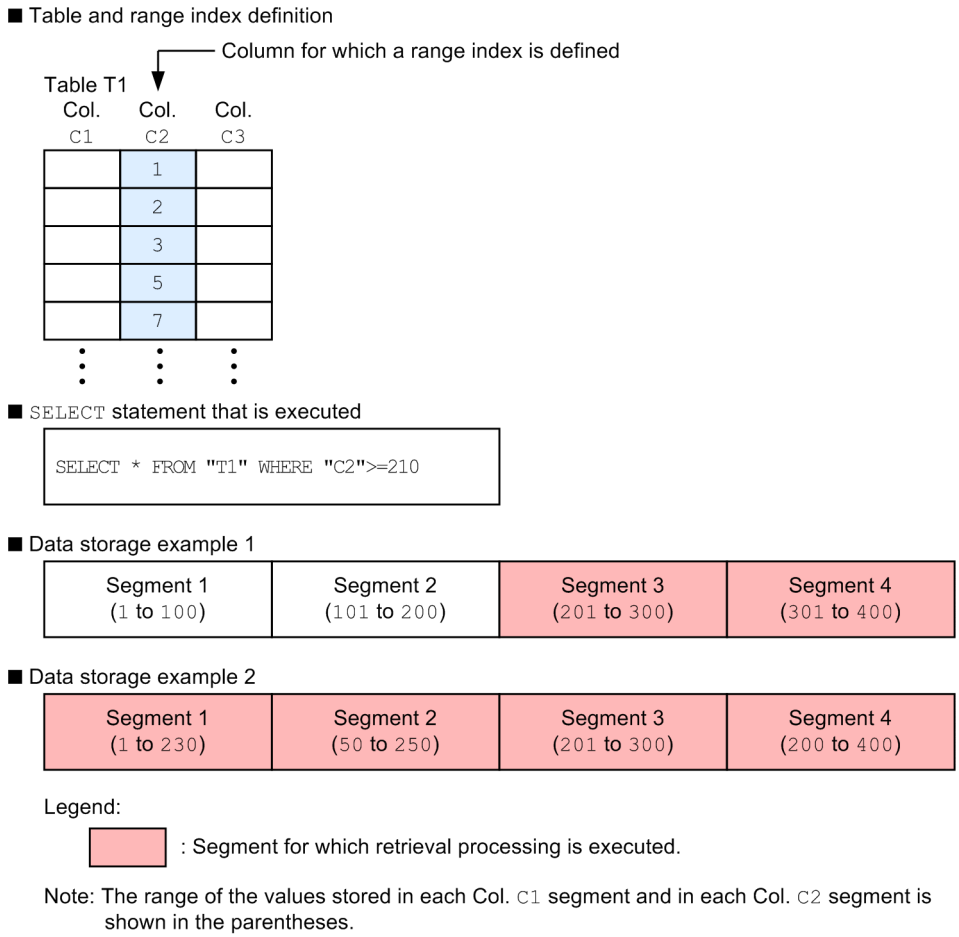
In a B-tree index, a single index key is created for an entire row in the table. In a text index, each table row is managed by assigning a number to each character string contained in that row.

In a range index, by contrast, one range is created for each segment. Therefore, the data size is smaller in range indexes than in B-tree indexes or text indexes.

(d) Retrieval performance varies depending on the range of data stored in the segment

The performance of a retrieval that uses a range index depends on the range of data stored in the segment. The following figure provides an example.

Figure 2-11: Example in which retrieval performance varies depending on the range of data stored in the segment



Explanation

In data storage example 1, retrieval processing is performed in two segments. In contrast, in data storage example 2, there are duplicate ranges and the range in each segment is wider than that in data storage example 1. Consequently, retrieval processing is performed in four segments. In this way, retrieval performance varies depending on the ranges of the data stored in the segments. Narrowing the ranges improves retrieval performance.

(3) Updating the range in a range index (segment)

When addition and update processing are performed repeatedly on rows, the ranges in the range index might expand, and as a result the benefits of using the range index might begin to diminish (the ranges never become narrower). When deletion processing is performed repeatedly on rows, the range index's ranges might become too wide for the ranges of data stored in the segments, and as a result the benefits of using the range index will diminish.

When the benefits of using a range index diminish, re-create the range index by performing index rebuilding.

The following table shows the conditions that result in the expansion of the range index's ranges.

Table 2-3: Conditions that expand the ranges in a range index (segment)

No.	Operation	Condition that expands ranges in a range index
1	Row addition	When a value that is outside a range is added, the range of the segment that stores the added row expands to accommodate the added value.
2	Row update	When a value is updated so that it is no longer within a range, the range of the segment that stores the updated row expands to accommodate the updated value.
3	Row deletion	No range expansion occurs.

2.4 DB areas (areas for storing tables and indexes)

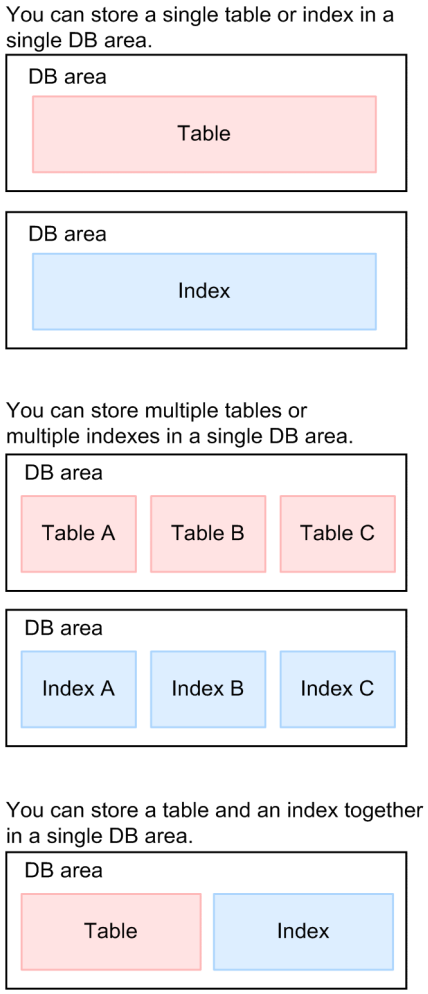
This section explains DB areas, which are logical areas in which tables and indexes are stored.

2.4.1 DB areas

HADB stores tables and indexes in a logical area called a *DB area*.

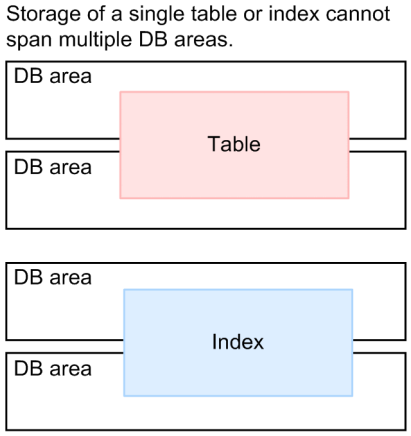
Each DB area can store multiple tables or indexes. The following figure shows cases in which tables and indexes can be stored in a DB area.

Figure 2-12: Cases in which tables and indexes can be stored in a DB area



Note that a single table or index cannot span multiple DB areas. The following figure shows cases in which a table or index cannot be stored in DB areas.

Figure 2-13: Cases in which a table or index cannot be stored in DB areas



2.4.2 DB area types

DB areas can be classified according to application. The following table shows the available types of DB areas.

Table 2-4: DB area types

No.	DB area type	Description
1	Data DB area	This DB area stores tables and indexes.
2	Work table DB area	This DB area stores the work tables created by HADB while executing SQL commands. For details about work tables, see *Types of work tables* under *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.
3	Master directory DB area	This DB area stores the internal information for the entire system.
4	Dictionary DB area	This DB area stores dictionary tables (base tables) and B-tree indexes of dictionary tables (base tables).
5	System-table DB area	This DB area stores system tables (base tables) and B-tree indexes of system tables (base tables).

Note

All DB areas, except for the data DB area, are created automatically by HADB.

■ Dictionary table

Dictionary tables store definition information for base tables, viewed tables, B-tree indexes, text indexes, and range indexes, as well as DB area information.

Dictionary tables can be classified into base tables and viewed tables. For details, see B.1 Dictionary table overview.

■ System table

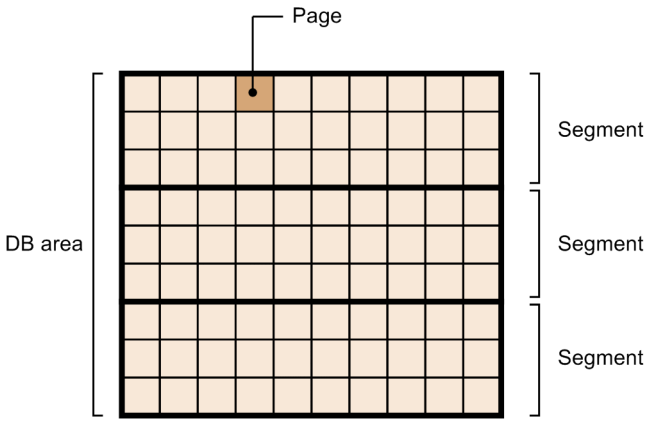
System tables store cost information related to base tables, B-tree indexes, and text indexes, as well as the chunk information for multi-chunk tables.

System tables can be classified into base tables and viewed tables. For details, see C.1 System table overview.

2.4.3 DB area structure (segments and pages)

A DB area consists of units called *segments* and *pages*. Each DB area consists of multiple segments. Furthermore, each segment consists of multiple pages. The following figure shows the structure of a DB area.

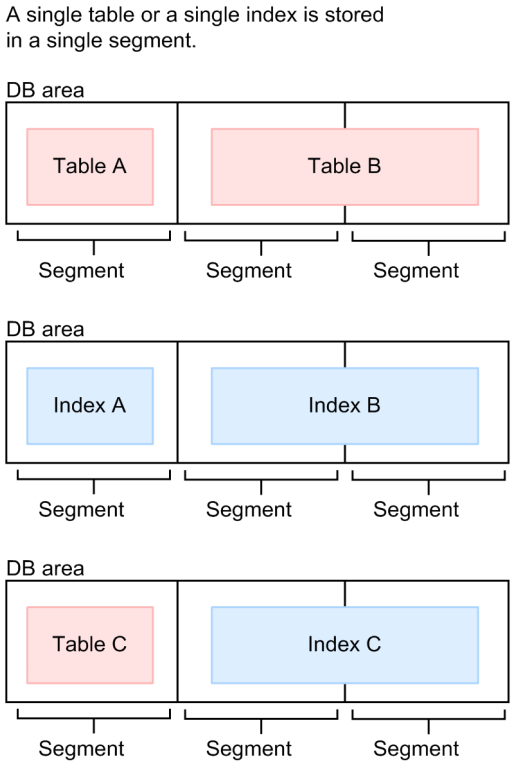
Figure 2-14: DB area structure



(1) Segment

A segment is the smallest unit that stores a table or index. Each segment can store a single table or index. The following figure shows the relationships that segments can have with tables and indexes.

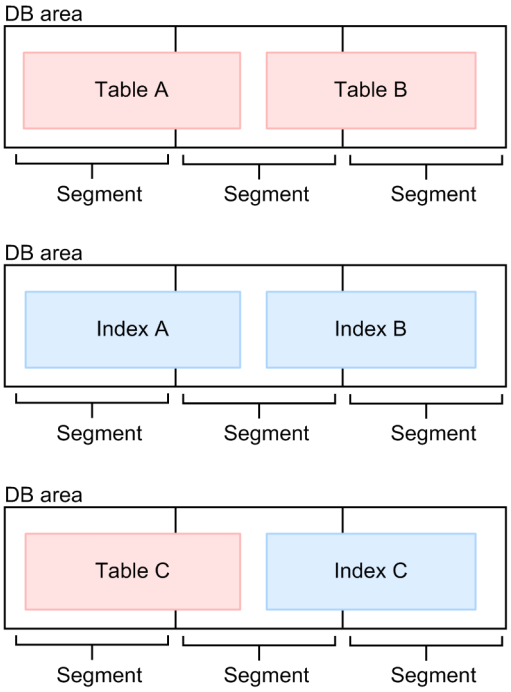
Figure 2-15: Relationships that segments can have with tables and indexes



Note that a single segment cannot store multiple tables or multiple indexes. The following figure shows examples of ways in which tables and indexes cannot be stored in segments.

Figure 2-16: Examples of ways in which tables and indexes cannot be stored in segments

Multiple tables or multiple indexes cannot be stored in a single segment.



(a) Segment types

Segments are classified according to the page types assigned to them as shown in the table below. For details about page types, see (a) Page types in (2) Pages under 2.4.3 DB area structure (segments and pages).

Table 2-5: Segment types

No.	Segment type		Description
1	Table (row store format)	Base row segment	Segment in which pages that store base rows (base row pages) are allocated
2		Branch row segment	Segment in which pages that store branch rows (branch row pages) are allocated
3	Table (column store format)	Column-data segment	Segment that stores data in column store format when importing data into a column store table. The column-data segment is also used to store data that is converted from row store format to column store format by the updated-row columnizing facility. For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).
4		Row-data segment	Segment that stores branched data (branch rows) when importing data into a column store table. The row-data segment is also used to store the following types of data: • Data added to a column store table by using the `INSERT` statement • Data updated in a column store table by using the `UPDATE` statement

No.	Segment type		Description
			• Data indicating that specific data was invalidated by running the `UPDATE` or `DELETE` statement
5	B-tree index	Upper page segment	Segment in which root pages and middle pages of a B-tree index are allocated
6		Lower page segment	Segment in which leaf pages, row ID directory pages, and row ID list pages of a B-tree index are allocated
7	Text index	String control segment	Segment in which string control pages of a text index are allocated
8		Position control segment	Segment in which position control pages of a text index are allocated
9	Range index	DB area file control segment	Segment in which DB area file control pages of a range index are allocated
10		Range control segment	Segment in which segment control pages and range control pages of a range index are allocated

(b) Timing at which segments are allocated or released

Timing at which segments are allocated

A segment is allocated when a new page is allocated. If there are no more free pages in a segment that has already been allocated, a new segment is allocated and assigned when a new page is allocated.

In the following cases, however, a new segment is allocated and assigned, even if there are free pages in a segment that has already been allocated:

- When background import is executed
- When multiple chunks are merged
- When a chunk status change swaps the current chunk and creates a new chunk
- When a system table is reorganized
- When HADB uses the updated-row columnizing facility to convert data from row store format to column store format

Timing at which segments are released

Allocated segments are released at the following times:

- When the base table and index are deleted
- When all row data is deleted from the base table (when the `TRUNCATE TABLE` statement is executed)
- When data is imported into the base table and deletion of all existing data is specified
- When the chunk is deleted
- When the index is rebuilt
- When the index data for the merge-source chunks is deleted after the chunks are merged
- When a chunk is archived
- When an archivable multi-chunk table is changed to a regular multi-chunk table (when the `ALTER TABLE` statement is executed)
- When a system table is reorganized

(2) Pages

A page is the smallest unit for disk I/O operations. Data is read from or loaded to a disk in page units.

(a) Page types

Pages are categorized into some types based on the type of data to be stored. The following table shows the page types.

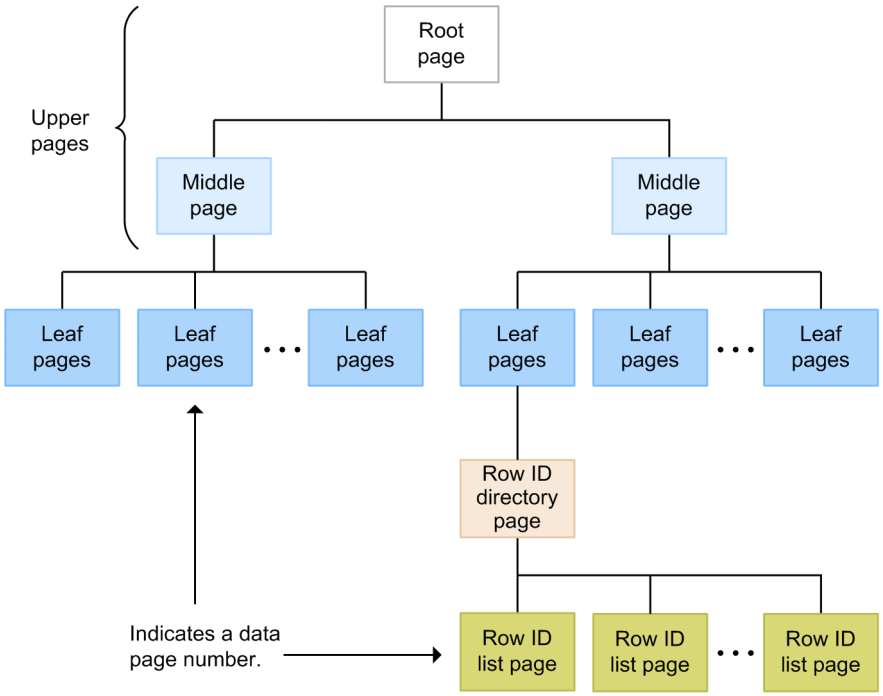
Table 2-6: Page types

No.	Page type			Description
1	Data page (row store format)	Base row page		Page for storing the base rows of a table
2		Branch row page		Page for storing the branch rows of a table
3	Data page (column store format)	Entry page		Page for storing the management information related to table data stored in column store format. This page is allocated in a column-data segment.
4		Dictionary page		Page for storing dictionary data when dictionary encoding (`DICTIONARY`) is selected as the compression type. This page is allocated in a column-data segment.
5		Column-data page		Page for storing column data (excluding branch row data) in column store format. This page is allocated in a column-data segment.
6		Basic row page		When the `INSERT` statement is used to add data to a column store table, the data is stored on this page as base rows. Also, when the `UPDATE` statement is executed for data that is stored in column store format, the updated data is stored on this page as base rows. This page is allocated in a row-data segment.
7		Branch row page		Page for storing the branch rows of row data stored in column store format. This page is allocated in a row-data segment. Column data whose actual length is 128 bytes or more is stored in this page as branch row data.
8		Invalid row information page		This page is used to manage information about data invalidated when the `UPDATE` or `DELETE` statement was executed for data that is stored in column store format. This page is allocated in a row-data segment.
9		Allocation control page		A page that is present at the beginning of a row-data segment and controls the allocation of pages within the segment.
10	B-tree index index page[#1]	Upper page	Root page	The root page is the highest level index page in the B-tree structure.
11			Middle page	Middle level B-tree index page
12		Leaf page		Lower level B-tree index page
13		Row ID directory page		Page for managing row ID lists
14		Row ID list page		Leaf page used exclusively for a key value if that key value is used 256 times or more

No.	Page type		Description
15	Text index index page	String control page	Page for managing the types of text-indexed character strings
16		Position control page	Page for managing the positions of text-indexed character strings in pages
17	Range index index page	DB area file control page	Page that stores pointers to relate DB area files to data ranges in the DB area files in a table for which a range index is defined This page also stores the ranges of data stored in a chunk of a table for which a range index is defined.
18		Segment control page	Page that stores pointers to relate segments to data ranges in the segments in a table for which a range index is defined
19		Range control page	Page that stores the ranges of data stored in a segment of a table for which a range index is defined
20	Work table page#2		Page for work table DB areas The page size for work table DB areas can exceed 32 kilobytes. There are no pages for storing branch rows.
21	Directory page#3		Page for storing the management information related to DB areas

#1

The following figure shows the relationship between a B-tree index and index pages:



#2

A work table page is allocated when you execute an SQL statement that creates a work table.

#3

A predetermined number of directory pages are allocated whenever DB areas are created. After that, new pages are allocated any time the number of data pages or index pages reaches a preset value.

■ **Base rows and branch rows**

In row store format, an entire row of data is stored on the same page. However, if there is a variable-length column whose definition length exceeds 255 bytes, one row of data might be broken up and stored on separate pages. Rows that are broken up and stored in separate pages are called *branch rows*. The row containing the information on the branched destination is called the *base row*. This applies to both the data of a row store table and the data added to a column store table by using an update SQL statement.

When you define a row store table (by executing the `CREATE TABLE` statement), you can specify whether a variable-length column's data is to be broken up and stored on separate pages.

■ **Existence of directory page groups and locations of directory pages**

Each DB area file begins with a directory page group, which manages the following types of information:

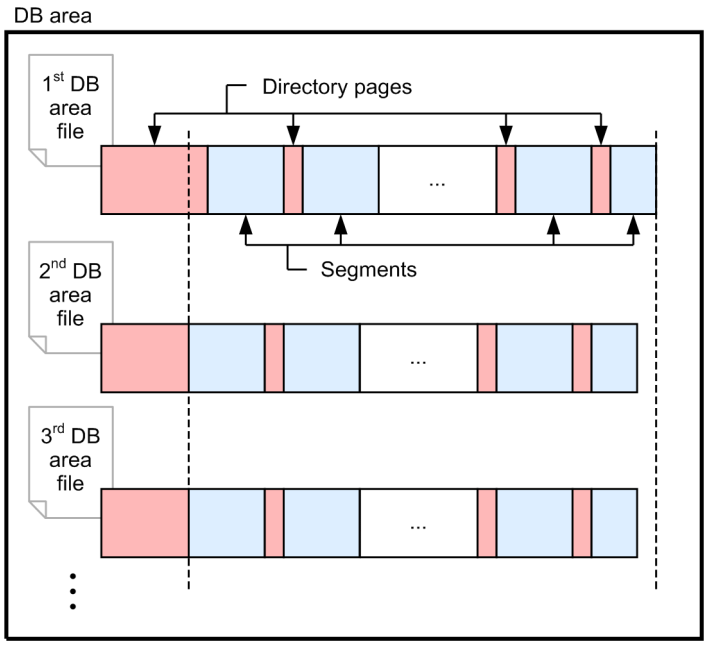
- DB area definition information
- Information about the defined tables and indexes
- Chunk information

In the first one of the DB area files that make up a DB area, the directory page group includes the directory page that manages the DB area information. Therefore, the locations of directory pages in the first DB area file are different from the locations of directory pages in other DB area files.

A directory page group is located before segments that store tables and indexes. A directory page is located at regular intervals to manage the information about each segment. Therefore, the entire space in the DB area file is not used for segments that store tables and indexes.

The following figure shows an example of the locations of directory pages in DB area files.

Figure 2-17: Example of directory page placement in DB area files



(b) Timing at which pages are allocated or released

Timing at which pages are allocated

Data pages and index pages are allocated at the following times:

- **When a row is added to, or updated in, a row store table**

  A base row page or a branch row page are allocated when a row is added to a table or when a row is updated. Page allocation does not take place when a row store table is defined by executing the `CREATE TABLE`

statement. When an already allocated page is filled to its capacity as a result of repeated operations to add rows, a new page will be reserved and allocated.

An index page of a range index is allocated when a row is added to the table or when a row is updated.

- **When a row is added to, or updated in, a column store table**
  - An entry page, dictionary page, or column-data page is allocated when a row is added to a column store table by executing the `adbimport` command.

    A basic row page is allocated when a row is added to, or updated in, a column store table by executing an update SQL statement.

    A branch row page is allocated when a row is added to, or updated in, a column store table by executing the `adbimport` command or an update SQL statement.

    An entry page, dictionary page, column-data page, basic row page, or branch row page is not allocated when a column store table is defined by using the `CREATE TABLE` statement. A new page is reserved and allocated when an already allocated page is filled to its capacity as a result of repeated operations to add rows.
  - An invalid row information page is allocated when the data in column store format is deleted or updated. The invalid row information page is not allocated when a column store table is defined by using the `CREATE TABLE` statement. A new page is reserved and allocated when an already allocated page is filled to its capacity as a result of repeated operations to delete or update rows.

    Note that an invalid row information page is not allocated when the data added or updated in row store format by using an update SQL statement is deleted or updated.
  - An index page of a range index is allocated when a row is added to the table or when a row is updated.
- **When HADB uses the updated-row columnizing facility to convert data from row store format to column store format**

  When data in a column store table is converted from row store format to column store format, a new page is reserved and allocated.
- **When a B-tree index is defined**

  When a B-tree index is defined by executing the `CREATE INDEX` statement, one upper page and one leaf page are allocated.
- **When index page split occurs in a B-tree index**

  A new index page is allocated and assigned every time a B-tree index page split occurs. For details about B-tree index page splits, see (2) B-tree index page splits in 5.3.4 Allocating an unused area inside a B-tree index page (PCTFREE).
- **When a text index is defined**

  When a text index is defined by executing the `CREATE INDEX` statement, two string control pages and two position control pages are allocated.
- **When index page split occurs in a text index**

  A new index page is allocated and assigned every time an index page split occurs in a text index. For details about index page splits of text indexes, see (2) Text index page splits in 5.4.2 Allocating an unused area inside a text index page (PCTFREE).
- **When a range index is defined**

  When a range index is defined by executing the `CREATE INDEX` statement, one segment's worth of the DB area file control page is allocated.

Timing at which pages are released

When a segment is released, all pages inside that segment are released.

(c) Page group

When pages are allocated from the range control segment of a range index, HADB allocates multiple pages at once. The pages that are allocated together form a unit called a *page group*. The number of pages comprising a page group is called the *page group size*.

To determine the page group size, see the description of the variable *PGGRPSIZE* in (1) Determining the SGRI variable under 5.8.6 Determining the number of segments for storing each range index.

To check the page group size of a range index, see 10.9.5 Checking the status and usage of range indexes.

2.4.4 DB area file configuration

All data, including tables and indexes, that is stored in a DB area is physically stored in files. Files that comprise a DB area are called *DB area files*.

(1) Files that can be used as DB area files

In HADB, the files types listed in the following subsections can be used as DB area files.

(a) Regular files (files in a file system)

Regular files that are normally supported by the OS can be used as DB area files.

When a regular file is updated or referenced, the file system's control area is updated or referenced. Therefore, especially when you are adding to a file, the access performance will be lower than when a block special file is used.

(b) Block special files

Block special files can be used as DB area files. A block special file functions as a single block device. It allows an entire disk or a partitioned area of a disk to be treated and accessed as a single file.

In block special files, data is manipulated in blocks. Consequently, block special files can be accessed faster than regular files.

Before the user creates a database, a block special file must be allocated taking the expected size of the database area into consideration.

Note that you must grant read-write privileges to the block special file so that it can be read and written by the HADB server. When changing the settings of a block special file, you must use the `udev` rules. For details about the `udev` rules, see the documentation for your operating system.

Important

If a block special file is opened in the write mode, an event is issued to `udev` when that file is closed, returning it to the state specified in the `udev` rules. Consequently, if the settings of the block special file are changed using the `chmod` or `chown` command, when database initialization is complete or the HADB server terminates, the owner and privileges are returned to the states specified in the `udev` rules.

(2) Types of DB area files

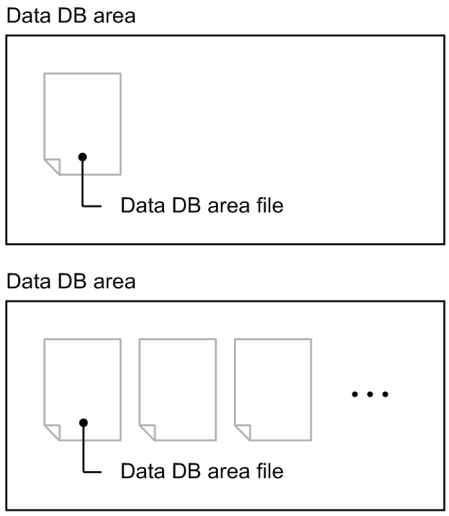
The following sections describe the five types of DB area files that are available, corresponding to the DB area types.

(a) Data DB area files

A data DB area is comprised of data DB area files.

Each data DB area consists of one or more data DB area files. A Data DB area can consist of a maximum of 1,024 data DB area files. The following figure shows the relationship between a data DB area and data DB area files.

Figure 2-18: Relationship between a data DB area and data DB area files



Data DB area files are normally created using a regular file under the DB directory, but can also be created using a block special file that is not under the DB directory.

The location for storing data DB area files is specified during database initialization. It is also specified during data DB area addition following database initialization.

(b) Work table DB area files

A work table DB area is comprised of a work table DB area file.

Normally, a single work table DB area file is created using a regular file under the DB directory, but it can also be created using a block special file that is not under the DB directory.

The location for storing work table DB area files is specified during database initialization.

We recommend that you use a block special file to create the work table DB area file. If you use a regular file and an SQL statement is executed that creates multiple work tables, I/O operations might become concentrated on the file system that stores the regular file, lengthening the processing time of the SQL statement.

(c) Master directory DB area file

A master directory DB area is comprised of a master directory DB area file.

Normally, a single master directory DB area file is created using a regular file under the DB directory, but it can also be created using a block special file that is not under the DB directory.

The location for storing the master directory DB area file is specified during database initialization.

(d) Dictionary DB area file

A dictionary DB area is comprised of a dictionary DB area file.

Normally, a single dictionary DB area file is created using a regular file under the DB directory, but it can also be created using a block special file that is not under the DB directory.

The location for storing the dictionary DB area file is specified during database initialization.

(e) System-table DB area file

A system-table DB area is comprised of a system-table DB area file.

Normally, a single system-table DB area file is created using a regular file under the DB directory, but it can also be created using a block special file that is not under the DB directory.

The location for storing the system-table DB area file is specified during database initialization.

We recommend that you use a block special file to create the system-table DB area file. If you use a block special file, the HADB server outputs a warning message when the free space in the system-table DB area becomes insufficient and the system table needs to be reorganized.

Note that if you use a regular file to create the system-table DB area file, the HADB server does not output a warning message even when the free space in the system-table DB area becomes insufficient. If you choose to create a system-table DB area file by using a regular file, consider the time at which the system table is to be reorganized before you start operation. Then, reorganize the system table periodically based on the result of consideration. For details about reorganizing the system table, see 11.17 Reorganizing system tables.

(3) DB directory

The directory under which various DB area files, such as data DB area files, are stored is called the *DB directory*. The DB directory is created during database initialization.

The DB directory also stores the following files:

- Database update history information (system log files)
- Information (status files) necessary for restarting the HADB server in case of an abnormal termination

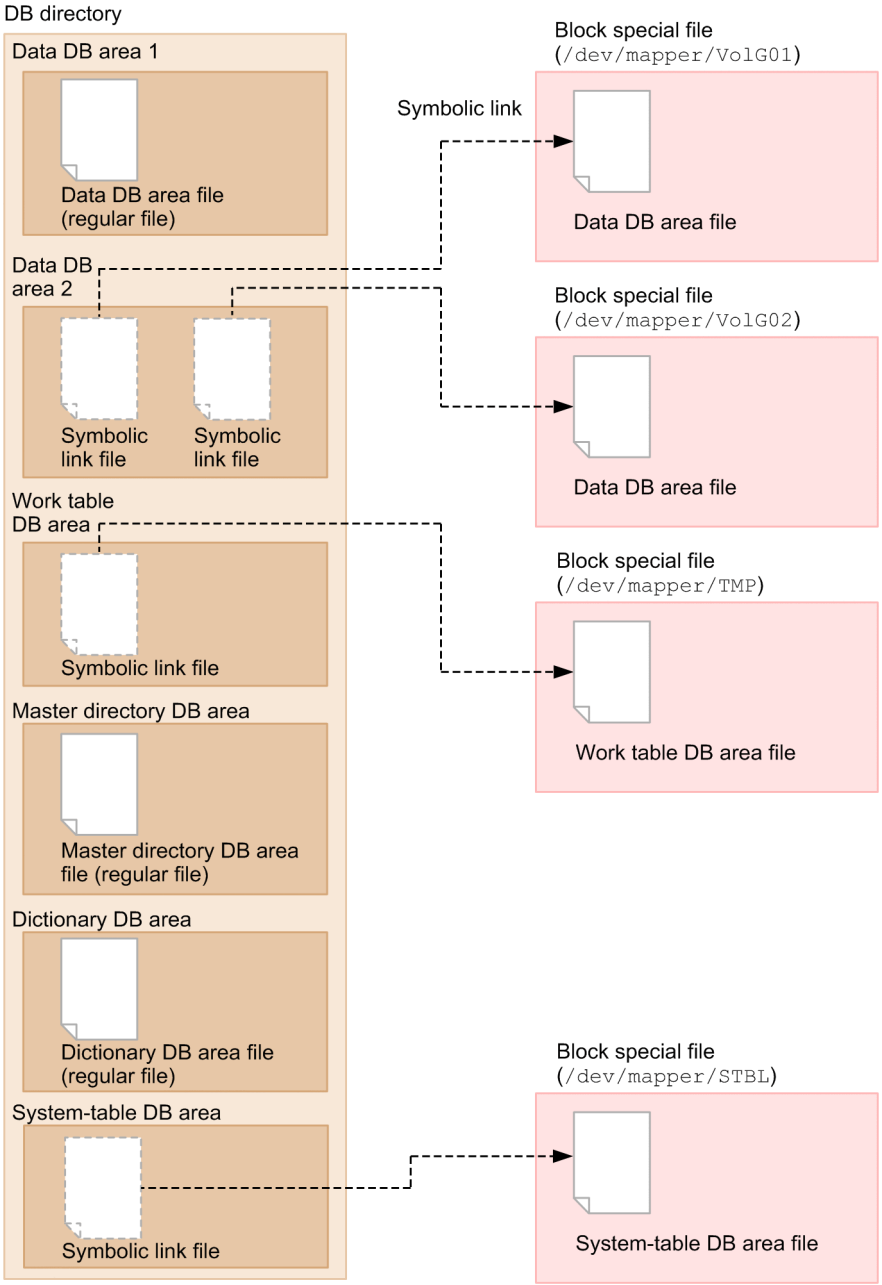
For details about the structure of the DB directory, see A.3 DB directory configuration.

(4) DB area file configuration example

When a DB area file is created using a block special file that is not under the DB directory, a symbolic link file pointing to the actual file is created under the DB directory.

The following figure shows an example of a DB area file configuration.

Figure 2-19: Example of a DB area file configuration



(5) Notes about file systems

- Create the files that make up the database (DB area files and files in the DB directory) in an ext3, ext4, or XFS file system.
- Do not create files that make up a database (DB area files and files in the DB directory) in a file system mounted via an NFS network.

(6) Relationship between DB area files and the multi-node function

When you use the multi-node function, you need to allocate block special files to the following DB area files:

- Data DB area file
- Master directory DB area file

- Dictionary DB area file
- System-table DB area file

You must also make sure that the disk corresponding to the block special files to be allocated can be referenced from all nodes in the multi-node configuration.

2.4.5 DB area automatic extension

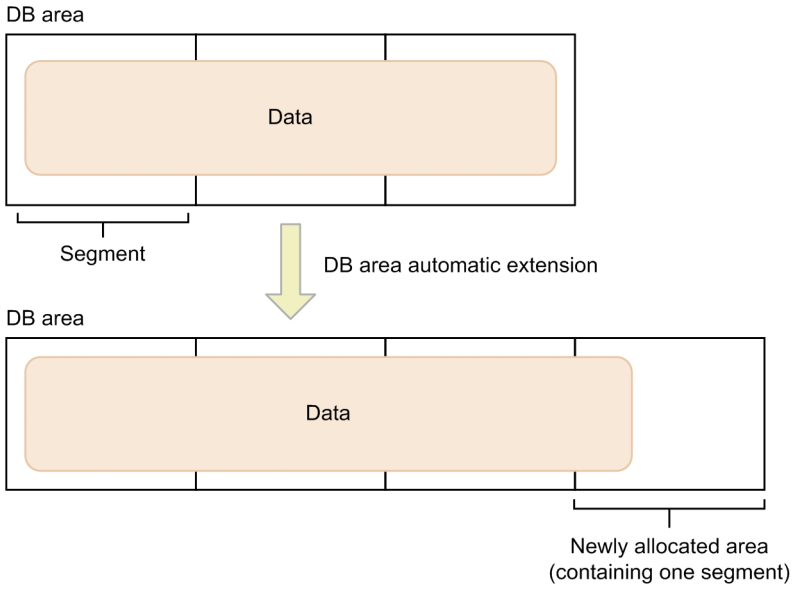
When the addition of data or some other action causes a DB area to run out of space, HADB automatically increases the size of the DB area. This is called *DB area automatic extension*.

(1) Unit for DB area automatic extension

When a DB area is automatically extended, its capacity is extended by one segment at a time.

The following figure shows an example of DB area automatic extension.

Figure 2-20: Example of DB area automatic extension



When a DB area file is configured using a regular file, automatic extensions will reduce the performance of update processing. Therefore, we recommend that you take one of the following steps, and based on the results, initialize the necessary amount of DB area during database initialization or DB area addition/expansion:

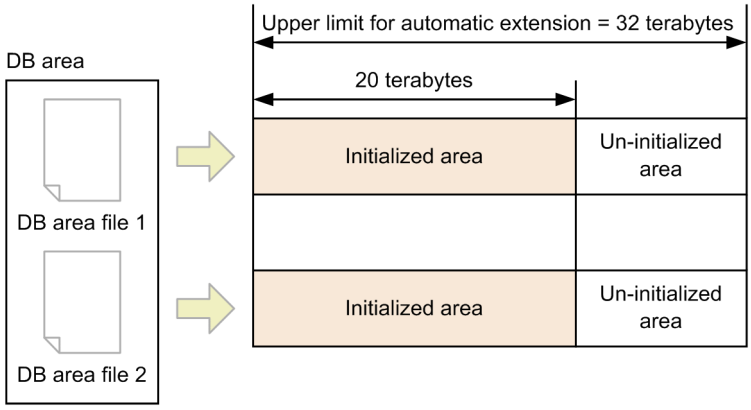
- Make an estimate of the amount of DB area that you will need.
- Execute the `adbdbstatus` command in a small-scale test environment to determine the size of the DB area.

(2) Maximum value for DB area automatic extension

In DB area automatic extension, one segment of storage is added to the DB area every time a space shortage occurs, up to its maximum value.

You can determine the maximum value automatic extension for each DB area file by rounding up the file size that was specified during database initialization or DB area addition/expansion to the nearest multiple of 16 terabytes. The following figure shows an example of the maximum value for automatic extension.

Figure 2-21: Example of the maximum value for DB area automatic extension



Explanation

When two DB areas are initialized to 20 terabytes each during database initialization or DB area addition/expansion, 32 terabytes becomes the maximum value for automatic extension for each DB area file. Therefore, the maximum value for the two DB areas combined is 64 terabytes.

For a work table DB area, the maximum value for work table DB area files is the upper limit for automatic extension. It is not the value obtained by rounding up the area initialized during database initialization in 16-terabyte units.

2.5 Users and schemas

This section explains the types of users who are involved in HADB server operation. It also explains a schema, which is a concept that includes tables in indexes.

2.5.1 User types (OS users and HADB users)

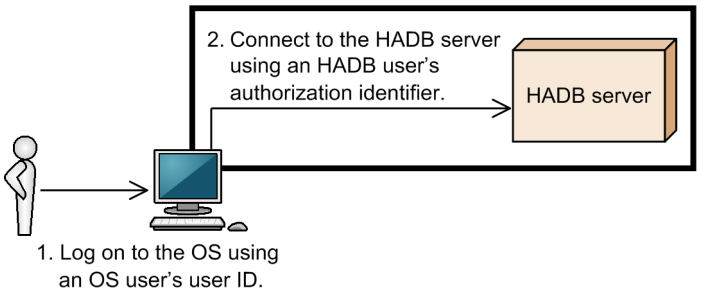
The following types of users are involved in HADB server operation:

- OS user
- HADB user

Since OS users are different from HADB users, you must manage them properly so that they do not get mixed up.

The following figure provides an overview of OS users and HADB users.

Figure 2-22: Overview of OS users and HADB users



(1) OS user

An OS user is a user who logs on to the OS and can utilize the OS functions. To log on to the OS, the user must possess an OS user ID and password.

An OS user ID and password do not permit the OS user to connect to the HADB server. To connect to the HADB server, the user must possess an HADB user ID and password.

There is a concept called *privilege* that affects OS users. To install the HADB server under the OS, an OS user must have administrator privileges.

There is also a concept called *group* that affects OSs. The operations that can be executed differ depending on the group to which an OS user belongs.

The following types of OS users are involved in HADB server operation:

- Superuser

  This is an OS user who has administrator privileges.

  A superuser takes actions such as setting up the OS environment, managing OS users, and installing the HADB server.
- HADB administrator

  This is a dedicated OS user who manages the HADB server.

  The HADB administrator is created when the superuser adds an OS user who will function as the HADB administrator to the OS of the machine on which the HADB server is to be installed.

The HADB administrator takes actions such as setting up the HADB server environment, starting/terminating the HADB server, and managing/operating the HADB server.

The HADB administrator has the privileges needed to execute various types of HADB commands. The HADB administrator also becomes the owner of the directories for operating the HADB server (server directory and DB directory) and the files created under these directories.

- OS user belonging to the HADB administrators group

  This is an OS user who manages the HADB server separately from the HADB administrator. This user belongs to the same OS group (HADB administrators group) as the HADB administrator.

  This user is created when the superuser adds an OS user belonging to the HADB administrators group to the OS of the machine on which the HADB server is to be installed.

  The OS user belonging to the HADB administrators group has the privileges needed to execute some of the HADB commands. This OS user can also access the server directory and DB directory owned by the HADB administrator.

(2) HADB user

An HADB user is a user who, by connecting to the HADB server, can take such actions as importing data, retrieving or updating data, and managing HADB users.

To connect to the HADB server, an HADB user authorization identifier and password must be entered. A user authorization identifier is a user ID that is used for connecting to the HADB server.

The first HADB user is created during database initialization. The created HADB user is managed within the HADB server. You can also create multiple HADB users.

An HADB user must possess the privileges needed to connect to the HADB server and perform various operations. For details about privileges, see 2.7.1 Privilege types.

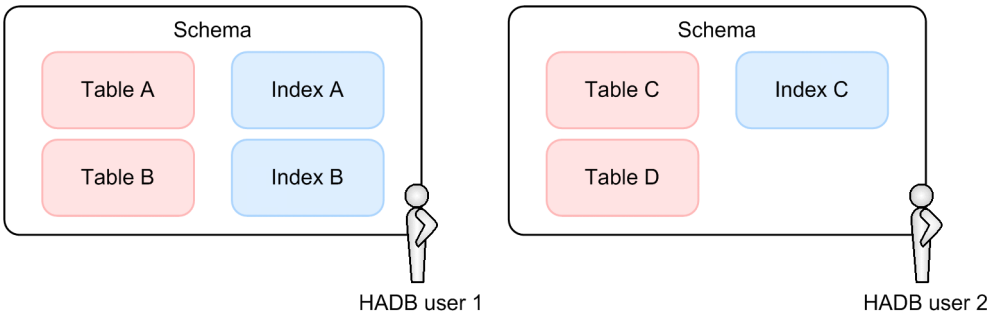
2.5.2 Schemas

A schema is a logical concept that includes tables and indexes. Each HADB user can own a single schema only. Elements that can be defined in a schema are called schema objects.

In HADB, a user (the HADB user with the authorization identifier that was used for the current connection to the HADB server) can define schema objects only in a schema which that user owns. He or she cannot define schema objects in schemas owned by other HADB users.

The following figure shows the concept of schema.

Figure 2-23: Concept of schema



The following lists the schema objects that can be handled in HADB.

▪ **List of schema objects**

- Base table
- Viewed table
- B-tree index
- Text index
- Range index

HADB manages the definition information for tables and indexes in tables that are called *dictionary tables*. It also manages the cost information and chunk information for tables and indexes in table format. These tables are called *system tables*. Once HADB's initial setup has been completed, a schema (called the `MASTER` schema) that owns the dictionary tables and system tables is created automatically.

The name of a schema is called a *schema name*. One schema name is assigned to each HADB user. A schema name uniquely identifies a group of tables and indexes in the HADB database.

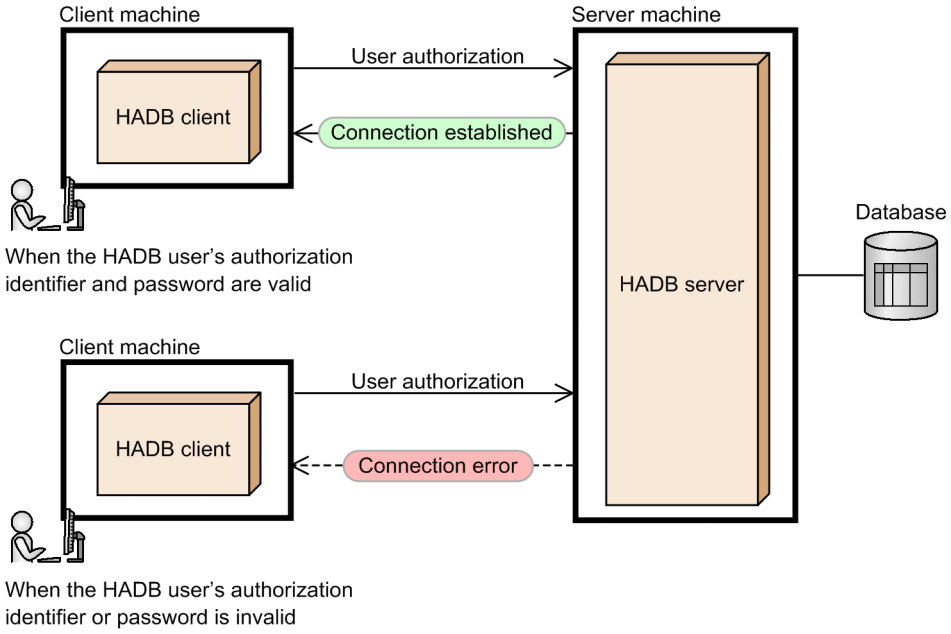
In HADB, the user ID (authorization identifier) that identifies an HADB user who is accessing the HADB server is the same as that user's schema name.

2.6 User authorization

When an HADB user attempts to connect to the HADB server, the server performs authorization using the HADB user's authorization identifier and password. This process is called *user authorization*.

The following figure provides an overview of user authorization.

Figure 2-24: User authorization overview



When an HADB user enters a correct authorization identifier and password, and has the `CONNECT` privilege, that HADB user can connect to the HADB server.

If the entered authorization identifier or password is incorrect, or if the HADB user does not have the `CONNECT` privilege, the attempt to connect to the HADB server ends in an error.

For details about the `CONNECT` privilege, see (2) CONNECT privilege in 2.7.2 User privileges.

2.7 Privileges

This section explains the types of privileges that are needed to operate HADB servers.

2.7.1 Privilege types

To operate an HADB server, an HADB user needs various types of privileges.

When an HADB user tries to perform operations on an HADB server, an error occurs if the HADB user does not have the required privileges. Therefore, by granting only the privileges that are needed for the operations that the applicable HADB user will perform, you can reduce the risk of unauthorized operations.

An HADB user can have multiple privileges. Privileges can be classified roughly into the following four types:

- **User privileges**
  For details, see 2.7.2 User privileges.
- **Schema operation privilege**
  For details, see 2.7.3 Schema operation privilege.
- **Audit privilege**
  For details, see 2.7.4 Audit privilege.
- **Access privilege**
  For details, see 2.7.5 Access privileges.

> **Note**
>
> To grant or revoke a privilege, execute the `GRANT` or `REVOKE` definition SQL statement.

2.7.2 User privileges

User privileges are privileges that are required to manage HADB users and connect to the HADB server.

The following two types of user privileges exist:

- `DBA` privileges
- `CONNECT` privileges

(1) DBA privilege

The user privilege required for managing an HADB user and granting user privileges is called the *DBA privilege*.

An HADB user who has the `DBA` privilege can execute the following operations:

- **HADB user management**
  - Creating an HADB user
  - Changing the user information of a created HADB user#

- Deleting a created HADB user#

- **User privilege and schema operation privilege management**
  - Granting user privileges and the schema operation privilege to HADB users
  - Revoking user privileges and the schema operation privilege granted to HADB users#
- **Granting audit privileges**
  - Granting audit privileges (audit admin privilege and audit viewer privilege) to HADB users

#

You cannot perform the following operations in relation to an HADB user who has the audit admin privilege:

- Changing the user information of the HADB user
- Deleting the HADB user
- Revoking user privileges and the schema operation privilege

An HADB user who has the `DBA` privilege can reference all information stored in dictionary tables (except for `SQL_AUDITS`) and system tables.

(2) CONNECT privilege

The user privilege that an HADB user must have to connect to the HADB server is called the *CONNECT privilege*.

If an HADB user does not have the `CONNECT` privilege, that HADB user cannot connect to the HADB server, even if he or she has the `DBA` privilege, schema operation privilege, audit privilege, or access privilege.

2.7.3 Schema operation privilege

The privilege that an HADB user must have to manage schemas is called the *schema operation privilege*.

The following type of schema operation privilege exists:

- Schema definition privilege

(1) Schema definition privilege

The schema operation privilege that an HADB user must have to define a schema is called the *schema definition privilege*.

Defining or deleting schemas, tables, or indexes requires the schema definition privilege.

2.7.4 Audit privilege

Audit privilege is a collective term for the following two privileges. You must have the audit privilege to use the audit trail facility. For details about the audit trail facility, see 2.18 Audit trail facility.

- Audit admin privilege

  The privilege that an HADB user must have to perform operation of the audit trail facility is called the audit admin privilege. An HADB user who has the audit admin privilege can:

  - Use the `adbauditttrail` command to operate the audit trail facility

- Use `CREATE AUDIT` statements to define audit targets
- Use `DROP AUDIT` statements to delete audit target definitions
- Use `REVOKE` statements to revoke audit privileges
- Reference audit target definition information (`SQL_AUDITS` retrieval)

> **Important**
>
> A given HADB user cannot have both the audit admin privilege and the `DBA` privilege.

- Audit viewer privilege

  The privilege that an HADB user must have to reference audit trails is called the audit viewer privilege. An HADB user who has the audit viewer privilege can:

  - Use the `ADB_AUDITREAD` function to reference an audit trail
  - Use the `adbconvertaudittrailfile` command to convert audit trail files

> **Note**
>
> `ADB_AUDITREAD` is a function that converts the output audit trails to data in tabular format that can be retrieved by the HADB server. This user can reference audit trails by specifying the `ADB_AUDITREAD` function in a `SELECT` statement.

2.7.5 Access privileges

This section describes the privileges (access privileges) that control access to schema objects.

(1) About access privileges

Access privileges are privileges required to access the following schema objects:

- Base tables
- Viewed tables

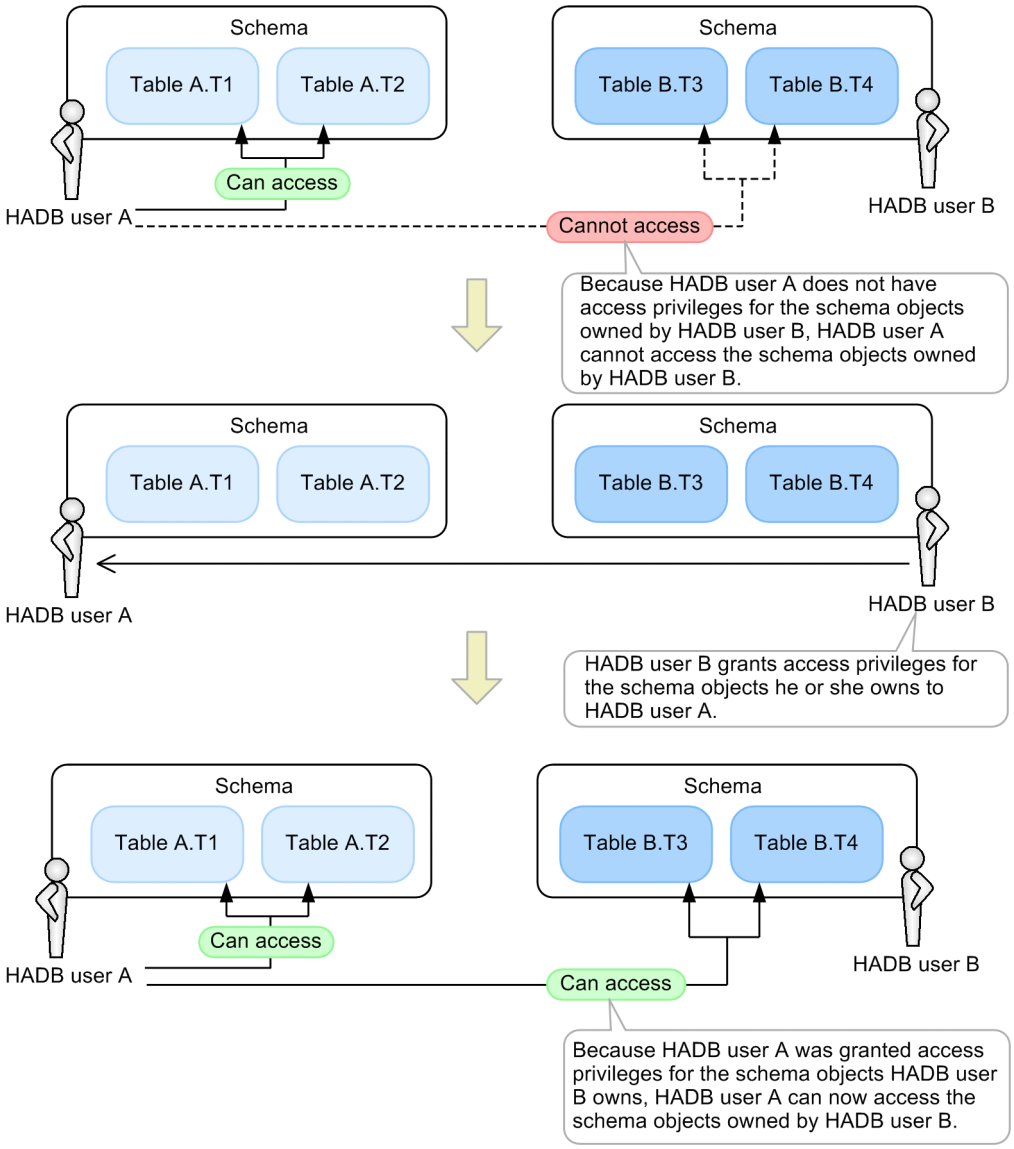
Before you can use SQL statements to search the data in a schema object or use commands to manipulate a schema object, you must have access privileges for that schema object.

You will automatically have access privileges for schema objects that you (the HADB user with the authorization identifier currently connected to the HADB server) own. This means that each user can access the schema objects he or she owns.

Ordinarily, you cannot access schema objects owned by other HADB users because you will not have access privileges for them. To access a schema object owned by another HADB user, you must be granted access privileges by an HADB user who already has access privileges for that schema object.

The following figure shows an example of accessing a schema object owned by another HADB user:

Figure 2-25: Example of accessing schema objects owned by another HADB user



Explanation

HADB user A owns table `A.T1` and table `A.T2`. HADB user B owns table `B.T3` and table `B.T4`.

Because HADB user A as owner of table `A.T1` and table `A.T2` has access privileges for those tables, he or she is able to access table `A.T1` and table `A.T2`. However, HADB user A does not have access privileges for tables `B.T3` and `B.T4` which HADB user B owns. Therefore, HADB user A cannot access table `B.T3` or table `B.T4`. To gain access to tables `B.T3` and `B.T4`, HADB user A must be granted access privileges for tables `B.T3` and `B.T4` by HADB user B.

Note

An HADB user can only access schema objects for which they have access privileges.

(2) Types of access privileges

The following table lists the types of access privileges that are available in HADB. The operations you can perform with respect to a schema object depend on the types of access privileges you have.

Table 2-7: List of access privileges

No.	Access privilege type	Explanation	Schema object to be referenced
1	SELECT privilege	The SELECT privilege is the privilege that an HADB user must have to reference the data of a schema object. To execute the SELECT or CREATE VIEW statement, you need the SELECT privilege for the relevant schema object.	• Base table • Viewed table
2	INSERT privilege	The INSERT privilege is the privilege that an HADB user must have to insert data into a schema object. To execute the INSERT statement, you need the INSERT privilege for the relevant schema object.	• Base table • Viewed table
3	UPDATE privilege	The UPDATE privilege is the privilege that an HADB user must have to modify the data of a schema object. To execute the UPDATE statement, you need the UPDATE privilege for the relevant schema object.	• Base table • Viewed table
4	DELETE privilege	The DELETE privilege is the privilege that an HADB user must have to delete the data of a schema object. To execute the DELETE statement, you need the DELETE privilege for the relevant schema object.	• Base table • Viewed table
5	TRUNCATE privilege	The TRUNCATE privilege is the privilege that an HADB user must have to delete all of a schema object's data in a single operation. To execute the TRUNCATE TABLE or PURGE CHUNK statement, you need the TRUNCATE privilege for the relevant schema object.	Base table
6	REFERENCES privilege	The REFERENCES privilege is the privilege that an HADB user must have to define a referential constraint (foreign key) in the CREATE TABLE statement. To define a foreign key, you need the REFERENCES privilege to the referenced table (the table in which the primary key is defined).	Base table
7	IMPORT TABLE privilege	The IMPORT TABLE privilege is the privilege that an HADB user must have to import data into a schema object. To execute the adbimport command, you need the IMPORT TABLE privilege for the relevant schema object.	Base table
8	REBUILD INDEX privilege	The REBUILD INDEX privilege is the privilege that an HADB user must have to rebuild the index of a schema object. To execute the adbidxrebuild command, you need the REBUILD INDEX privilege for the relevant schema object.	Base table
9	GET COSTINFO privilege	The GET COSTINFO privilege is the privilege that an HADB user must have to collect cost information from a schema object and from the index of that schema object, or to delete the collected cost information. To execute the adbgetcst command, you need the GET COSTINFO privilege for the relevant schema object.	Base table

No.	Access privilege type	Explanation	Schema object to be referenced
10	EXPORT TABLE privilege	The EXPORT TABLE privilege is the privilege that an HADB user must have to export the data of a schema object. To execute the adbexport command, you need the EXPORT TABLE privilege for the relevant schema object.	• Base table • Viewed table
11	MERGE CHUNK privilege	The MERGE CHUNK privilege is the privilege that an HADB user must have to merge multiple chunks that were created for a schema object into a single chunk. To execute the adbmergechunk command, you need the MERGE CHUNK privilege for the relevant schema object.	Base table
12	CHANGE CHUNK COMMENT privilege	The CHANGE CHUNK COMMENT privilege is the privilege that an HADB user must have to enter a comment for a chunk defined in a schema object, or to modify or delete the comment. To execute the adbchgchunkcomment command, you need the CHANGE CHUNK COMMENT privilege for the relevant schema object.	Base table
13	CHANGE CHUNK STATUS privilege	The CHANGE CHUNK STAUS privilege is the privilege that an HADB user must have to change the status of a chunk defined in a schema object. To execute the adbchgchunkstatus command, you need the CHANGE CHUNK STAUS privilege for the relevant schema object.	Base table
14	ARCHIVE CHUNK privilege	ARCHIVE CHUNK privilege is the privilege that an HADB user must have to archive chunks. To execute the adbarchivechunk command, you need the ARCHIVE CHUNK privilege for the relevant schema object.	Base table
15	UNARCHIVE CHUNK privilege	UNARCHIVE CHUNK privilege is the privilege that an HADB user must have to unarchive chunks. To execute the adbunarchivechunk command, you need the UNARCHIVE CHUNK privilege for the relevant schema object.	Base table

Note

- The owner of a base table (the HADB user who defined the base table) has all the access privileges listed in the preceding table for the base table they own.
- The owner of a viewed table (the HADB user who defined the viewed table) does not necessarily have all the access privileges listed in the preceding table for the viewed table they own. Which types of access privileges the owner of a viewed table has is determined by the access privileges they have for the underlying table. For details, see 2.7.6 Access privileges for viewed tables.

(3) Granting access privileges

To grant access privileges for a schema object to another HADB user, you must have a grant option for access privileges for that schema object. A grant option for access privileges is a privilege that allows an HADB user to grant access privileges for a schema object to another HADB user.

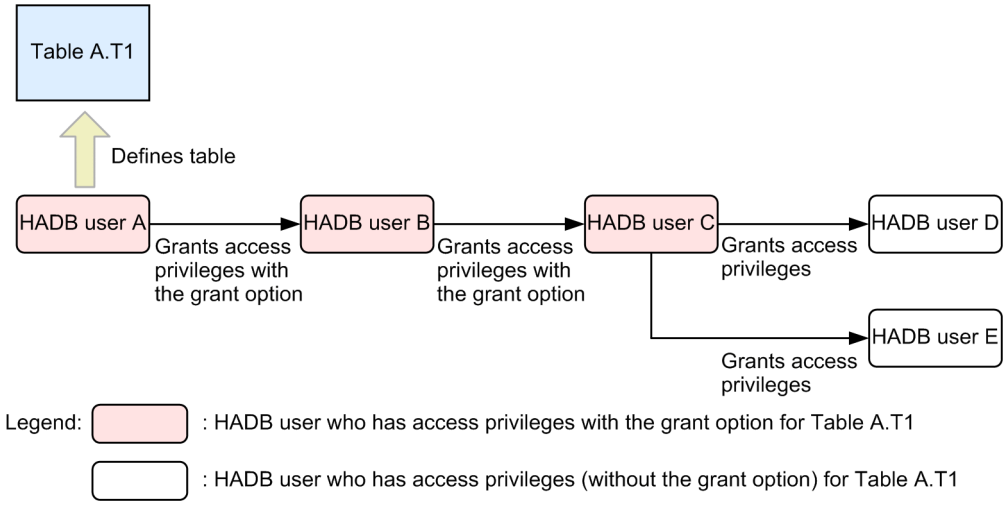
The access privileges owners of base tables have for the base tables they own automatically come with grant options. Although the owners of viewed tables have access privileges for the viewed tables they own, whether they also have grant options depends on whether they have access privileges with the grant options for the underlying table. For details, see 2.7.6 Access privileges for viewed tables.

Important

When granting access privileges, you can select whether to grant access privileges only, or grant access privileges together with the related grant option.

The following figure shows an example of granting access privileges for a schema object:

Figure 2-26: Example of granting access privileges for a schema object



Explanation

- HADB user A

  The owner of the schema object (table `A.T1`). HADB user A has access privileges with a grant option for table `A.T1`.

- HADB user B

  An HADB user who was granted access privileges with a grant option for table `A.T1` by HADB user A. HADB user B can access table `A.T1`, and can also grant access privileges for table `A.T1` to other HADB users.

- HADB user C

  An HADB user who was granted access privileges with a grant option for table `A.T1` by HADB user B. HADB user C can access table `A.T1`, and can also grant access privileges for table `A.T1` to other HADB users.

- HADB user D

  An HADB user who was granted access privileges without a grant option for table `A.T1` by HADB user C. HADB user D can access table `A.T1`. However, because HADB user D does not have a grant option for table `A.T1`, HADB user D cannot grant access privileges for table `A.T1` to other HADB users.

- HADB user E

  An HADB user who was granted access privileges without a grant option for table `A.T1` by HADB user C. HADB user E can access table `A.T1`. However, because HADB user E does not have a grant option for table `A.T1`, HADB user E cannot grant access privileges for table `A.T1` to other HADB users.

To grant access privileges to another HADB user, you execute a `GRANT` statement. To grant access privileges with a grant option, you execute the `GRANT` statement with the `WITH GRANT OPTION` option specified.

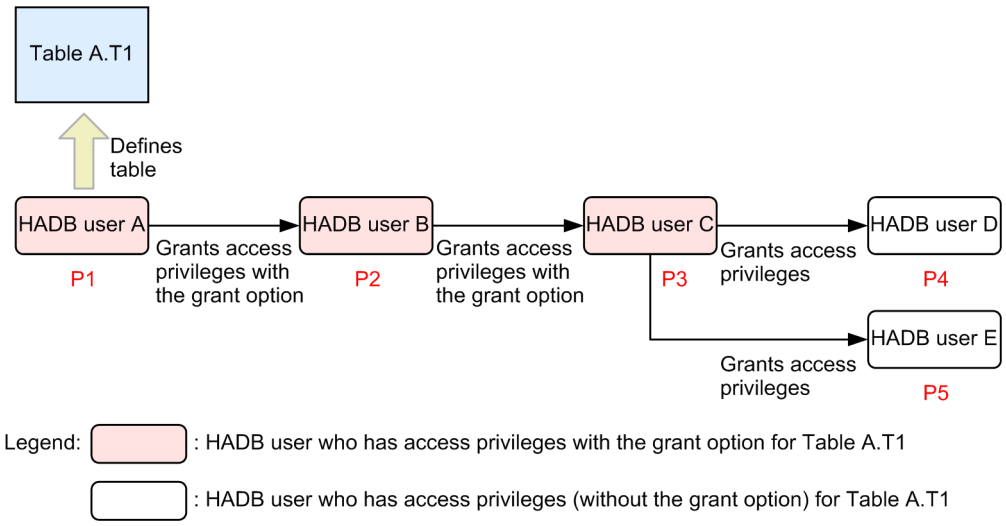
Important

In the preceding example, the access privileges HADB users B to E have for table `A.T1` are dependent privileges.

■ About dependent privileges

Access privileges granted to an HADB user by another HADB user are called dependent privileges.

Example:



- The access privileges HADB user A has for table `A.T1` shall be called `P1`.
- The access privileges HADB user B has for table `A.T1` shall be called `P2`.
- The access privileges HADB user C has for table `A.T1` shall be called `P3`.
- The access privileges HADB user D has for table `A.T1` shall be called `P4`.
- The access privileges HADB user E has for table `A.T1` shall be called `P5`.

In this scenario, the access privileges `P2` to `P5` are dependent privileges of the access privileges `P1`.

The dependent privileges of the access privileges `P2` are `P3` to `P5`.

The dependent privileges of the access privileges `P3` are `P4` and `P5`.

The access privileges `P4` and `P5` do not have any dependent privileges.

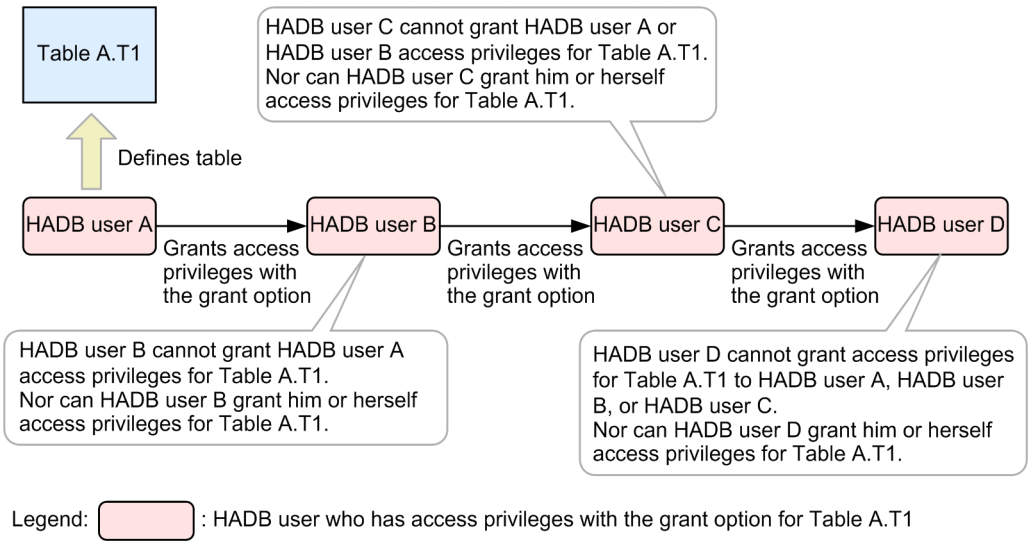
The dependent privileges you have are affected when the access privileges of the HADB user who granted you those privileges are revoked. This includes situations where the access privileges of an HADB user higher in the chain are revoked. For example, suppose that the access privileges `P2` are revoked. In this case, access privileges `P3` to `P5` which are its dependent privileges are also revoked. If the access privileges `P3` are revoked, the access privileges `P4` and `P5` which are its dependent privileges are also revoked.

■ HADB users to whom access privileges cannot be granted

You cannot grant access privileges to the following HADB users, even if you have a grant option for those access privileges:

- The HADB user who granted you the access privileges with the grant option
- An HADB user who is part of the chain that granted the preceding user the applicable access privileges with the grant option
- Yourself (you cannot grant yourself access privileges you have already been granted)

Example:



(4) Revoking access privileges

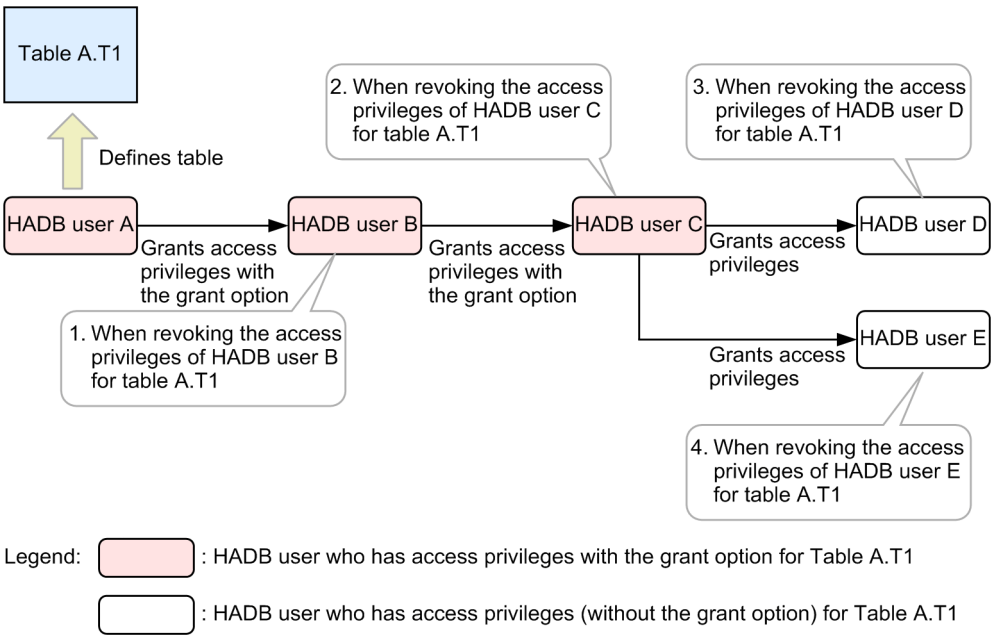
You can revoke the access privileges you have granted. You cannot revoke the access privileges granted by another HADB user.

Important

- If there are any dependent privileges of the access privileges you revoke, those dependent privileges will also be revoked.
- If you revoke access privileges with the grant option, the grant option is revoked along with the access privileges.

The following figure shows an example of revoking access privileges for a schema object:

Figure 2-27: Example of revoking access privileges for a schema object



Explanation

1. Revoking the access privileges of HADB user B for table `A.T1`

   Only HADB user A can revoke these access privileges.

   When you revoke the access privileges of HADB user B for table `A.T1` , the access privileges of HADB users C, D, and E for table `A.T1` are also revoked because they are dependent privileges.

2. Revoking the access privileges of HADB user C for table `A.T1`

   Only HADB user B can revoke these access privileges.

   When you revoke the access privileges of HADB user C for table `A.T1` , the access privileges of HADB users D and E for table `A.T1` are also revoked because they are dependent privileges.

3. Revoking the access privileges of HADB user D for table `A.T1`

   Only HADB user C can revoke these access privileges. The access privileges of HADB user D for table `A.T1` have no dependent privileges.

4. Revoking the access privileges of HADB user E for table `A.T1`

   Only HADB user C can revoke these access privileges. The access privileges of HADB user E for table `A.T1` have no dependent privileges.

To revoke access privileges, you execute a `REVOKE` statement.

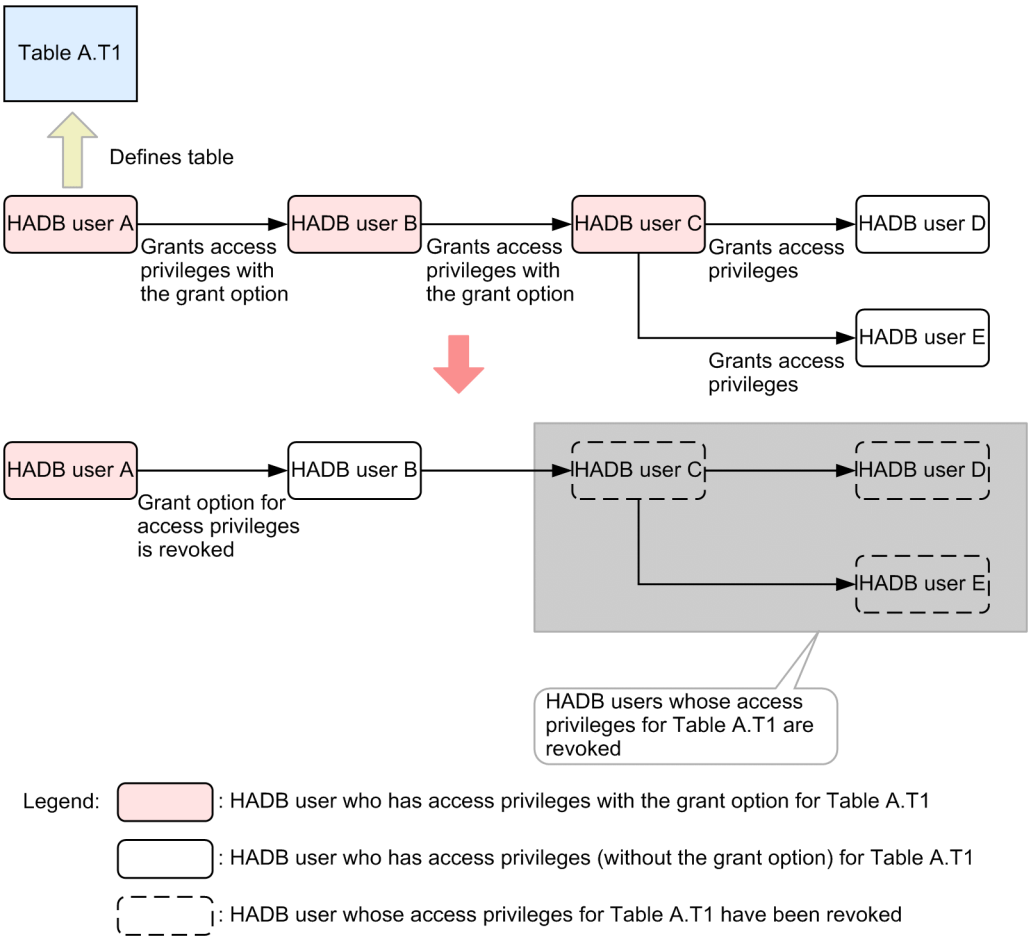
Note

- You cannot revoke the access privileges the owner of a schema object (HADB user A in the preceding example) has for that schema object (table `A.T1` in the preceding example).
- In the preceding example, HADB user A cannot directly revoke the access privileges of HADB user C for table `A.T1` because those access privileges were granted to HADB user C by another user. However, if HADB user A were to revoke the access privileges or grant option granted to HADB user B for table `A.T1`, the access privileges of HADB user C for table `A.T1` would also be revoked. Because the access privileges of HADB user C for table `A.T1` are dependent privileges of the access privileges of HADB user B for table `A.T1`, both can be revoked at once.

■ Revoking only the grant option of access privileges

You can revoke only the grant option of access privileges. When you revoke only the grant option, access privileges granted using that grant option and any dependent privileges are revoked.

Example:



Explanation

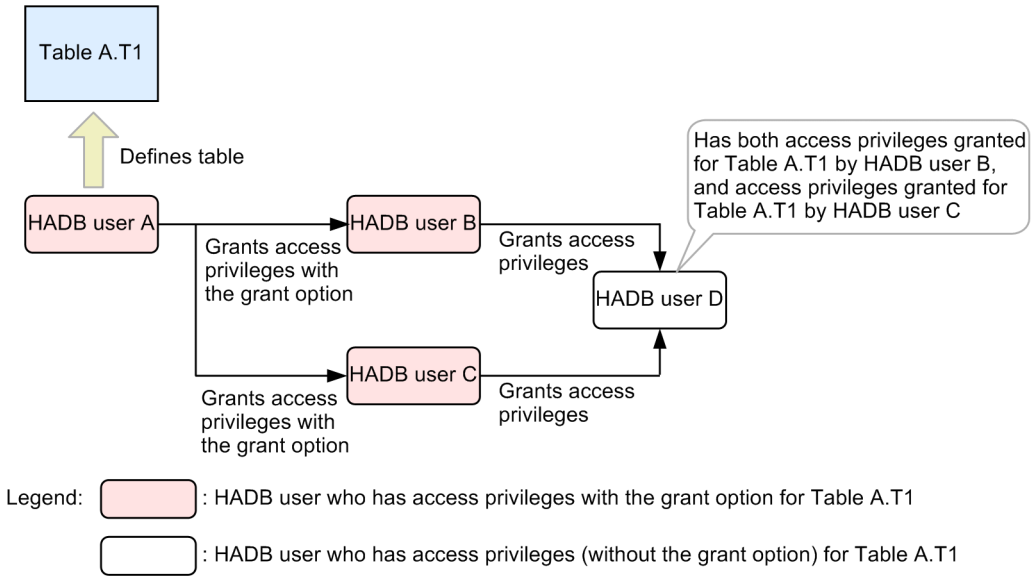
When HADB user A revokes the grant option for the access privileges of HADB user B for table `A.T1`, the access privileges of HADB user C for table `A.T1` are also revoked.

The access privileges of HADB users D and E for table `A.T1` are also revoked because they are dependent privileges of the access privileges HADB user C has for table `A.T1`.

■ When access privileges are granted for the same schema object by multiple HADB users

Suppose that as shown by the following example, HADB user B and HADB user C both grant access privileges for table `A.T1` to HADB user D. In this case, HADB user D will have access privileges for table `A.T1` granted by HADB user B, and access privileges for table `A.T1` granted by HADB user C.

Example:



For example, even if the access privileges for table `A.T1` that HADB user B granted to HADB user D were revoked, HADB user D would still have the access privileges for table `A.T1` granted by HADB user C. This is because an HADB user can only revoke the access privileges he or she has granted. Because HADB user D still has access privileges for table `A.T1`, HADB user D can continue to access table `A.T1`.

(5) Access privileges required to access multiple schema objects

When an HADB user executes an SQL statement, and multiple schema objects need to be accessed to process that SQL statement, the HADB user need access privileges to all of the schema objects in question.

An example follows in which the targeted schema objects are base tables.

Execution example of an SQL statement that accesses multiple base tables

```
SELECT * FROM "ADBUSER02"."T1", "ADBUSER02"."T2", "ADBUSER02"."T3"
```

Explanation

To retrieve base tables `T1`, `T2`, and `T3`, which are owned by another HADB user (`ADBUSER02`), the HADB user (`ADBUSER01`) needs an access privilege (`SELECT` privilege) to all three of those base tables. If the HADB user (`ADBUSER01`) does not have this access privilege (`SELECT` privilege) to base table `T3`, the above `SELECT` statement cannot be executed.

This requirement applies to the `SELECT` privilege as well as to other access privileges, such as the `INSERT`, `UPDATE`, and `DELETE` privilege.

2.7.6 Access privileges for viewed tables

This section describes access privileges for viewed tables.

■ Access privileges for defined viewed tables

Which access privileges an HADB user has for the viewed tables they define depends on the access privileges they have for the underlying table. For example, suppose an HADB user defines a viewed table based on an underlying table for which they have `SELECT` privilege and `INSERT` privilege. In this case, the HADB user who defined the viewed table will have `SELECT` privilege and `INSERT` privilege for that viewed table.

If the viewed table has more than one underlying table, the access privileges the HADB user has for the viewed table are the access privileges he or she has in common for all of the underlying tables.

Example:

Suppose that HADB user A has the following access privileges for the following tables:

- Table X.T1: SELECT privilege, INSERT privilege, and DELETE privilege
- Table X.T2: SELECT privilege, INSERT privilege, and UPDATE privilege
- Table X.T3: SELECT privilege, UPDATE privilege, and DELETE privilege

If HADB user A defines a viewed table A.V1 whose underlying tables are table X.T1 and table X.T2, HADB user A will have SELECT privilege and INSERT privilege for table A.V1.

If HADB user A defines a viewed table A.V2 whose underlying tables are tables X.T1, X.T2, and X.T3, HADB user A will only have SELECT privilege for table A.V2.

> **Note**
>
> If you want to have, for example, UPDATE privilege for a viewed table you define, you must have UPDATE privilege for all of its underlying tables.

For details about the rules that determine the access privileges for defined viewed tables, see *Rules* in *Specification format and rules for the CREATE VIEW statement* in the manual *HADB SQL Reference*.

■ Granting access privileges for viewed tables

To grant access privileges for a viewed table to another HADB user, you must have access privileges with a grant option for the underlying tables.

Example:

Suppose that HADB user A has the following access privileges for the following tables:

- Table X.T1: SELECT privilege (with grant option), INSERT privilege (with grant option), and UPDATE privilege
- Table X.T2: SELECT privilege (with grant option), INSERT privilege, and DELETE privilege (with grant option)

If HADB user A defines a viewed table A.V1 whose underlying table is table X.T1, HADB user A will have SELECT privilege (with grant option), INSERT privilege (with grant option), and UPDATE privilege for table A.V1. Because HADB user A has grant options for the SELECT privilege and INSERT privilege, he or she can grant SELECT privilege and INSERT privilege for viewed table A.V1 to other HADB users.

If HADB user A defines a viewed table A.V2 whose underlying tables are table X.T1 and table X.T2, HADB user A will have SELECT privilege (with grant option) and INSERT privilege for table A.V2. Because HADB user A has a grant option for the SELECT privilege, he or she can grant SELECT privilege for viewed table A.V2 to other HADB users.

■ Propagation of access privileges for viewed tables

When new access privileges are granted in relation to an underlying table, those access privileges are also newly granted for the viewed tables that depend on that underlying table. This is called the *propagation of access privileges*.

Example:

Suppose that HADB user A has defined viewed table A.V1 whose underlying table is X.T1, and viewed table A.V2 whose underlying table is A.V1. In this case, if HADB user A is then granted INSERT privilege for table X.T1, HADB user A will be automatically granted INSERT privileges for viewed tables A.V1 and A.V2.

Propagation of access privileges also takes place when access privileges for the underlying table are revoked.

Example:

Suppose that HADB user A has defined viewed table A.V1 whose underlying table is X.T1, and viewed table A.V2 whose underlying table is A.V1. In this case, if the INSERT privilege HADB user A has for table X.T1 is revoked, the INSERT privileges HADB user A has for viewed tables A.V1 and A.V2 are also revoked.

For details about the rules that govern the propagation of access privileges, see *Rules* in *Granting access privileges* in the manual *HADB SQL Reference*.

■ **Effect on viewed tables when SELECT privilege is revoked for tables**

If the SELECT privilege is revoked for the underlying table of a viewed table, all viewed tables that depend on that underlying table are invalidated.

Example:

Suppose that HADB user A has defined viewed table A.V1 whose underlying table is X.T1, and viewed table A.V2 whose underlying table is A.V1. In this case, if the SELECT privilege HADB user A has for table X.T1 is revoked, the viewed tables A.V1 and A.V2 that depend on table X.T1 are invalidated.

If the SELECT privilege with the grant option is revoked, or the grant option of the SELECT privilege is revoked, any SELECT privileges granted using that grant option are also revoked. This might also result in viewed tables being invalidated.

Example:

Suppose that the grant option HADB user A has for the SELECT privilege for table X.T1 is revoked. If HADB user A has used this grant option to grant SELECT privilege for table X.T1 to HADB user B, the SELECT privilege granted to HADB user B for table X.T1 is also revoked. If HADB user B has defined a viewed table B.V1 with table X.T1 as its underlying table, the viewed table B.V1 will be invalidated.

2.7.7 Scope of information in dictionary tables and system tables that can be referenced by HADB users

An HADB user who has the CONNECT privilege can reference information about the schema objects stored in dictionary tables and system tables (but not update or otherwise manipulate the data). However, the scope of information the HADB user can reference in the dictionary tables and system tables depends on the privileges granted to the HADB user. The following explains the key information HADB users can reference according to their privileges:

HADB users who have the DBA privilege

These users can reference all information stored in dictionary tables (except for SQL_AUDITS) and system tables.

HADB users who do not have the DBA privilege

These users can reference the definition information they themselves have defined (based on the authorization identifier that was used for the current connection to the HADB server). These users can also reference the following definition information even if it was defined by another HADB user:

- Information about schema objects such as tables and indexes for which they have access privileges, and information about the schema of these schema objects
- Information about schema objects for which access privileges are permitted by PUBLIC specification, and information about the schema of these schema objects

HADB users who have the audit admin privilege

These users can reference information subject to auditing by the audit trail facility (audit target definition information stored in the SQL_AUDITS dictionary table). Even without access privileges, these users can also reference some

information about schema objects defined by other HADB users and information about the schema of these schema objects.

For details about the scope of information in dictionary tables and system tables that can be referenced by HADB users, see the following subsections:

- (3) Scope of information in dictionary tables that can be referenced by HADB users in B.1 Dictionary table overview.
- (3) Scope of information in system tables that can be referenced by HADB users in C.1 System table overview.

2.7.8 Example of granting privileges to HADB users

This subsection uses an example to explain the flow of granting privileges to HADB users.

(1) Roles of HADB users and operation example

The figure below shows an operation example in which privileges are granted to HADB users. In this example, four types of HADB users are created.

Figure 2-28: Role of each HADB user and operation example

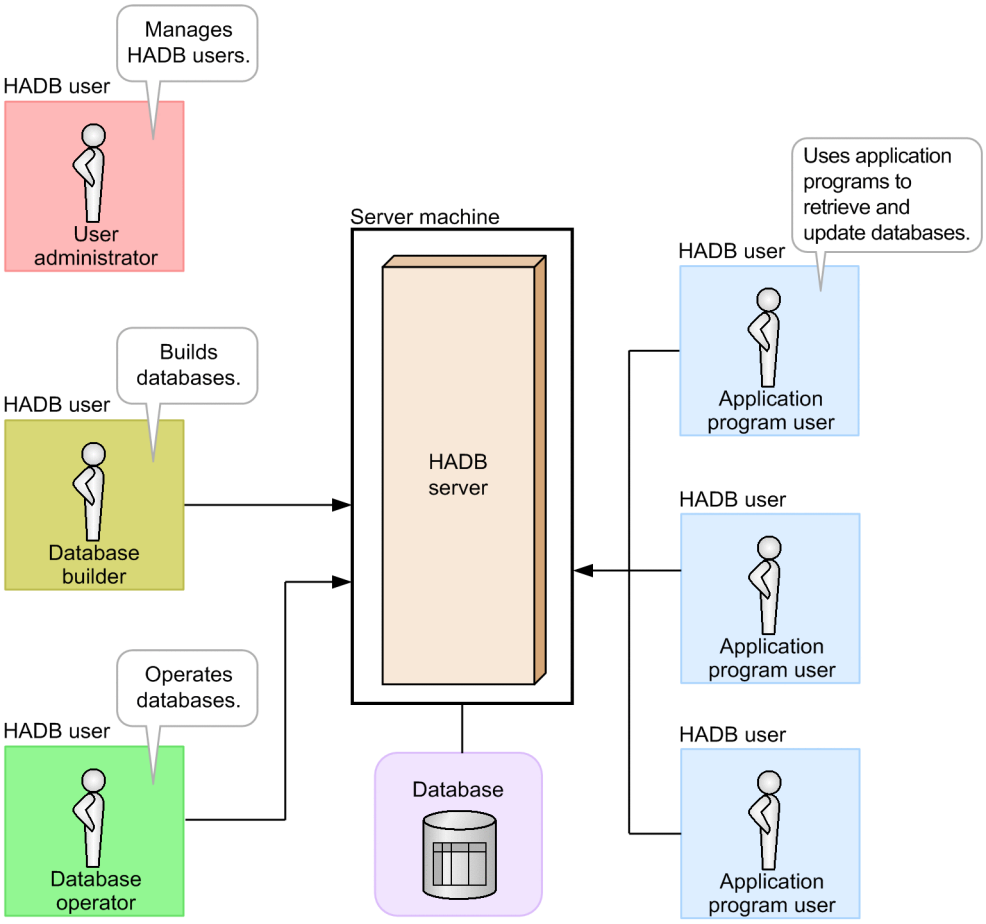


Table 2-8: Role assignment for HADB users to be created

No.	HADB user role	Explanation
1	User administrator	This is the HADB user who manages other HADB users. The user administrator creates HADB users who have the following roles: • Database builder • Database operator • Application program user The user administrator also grants the CONNECT privilege and schema definition privilege to the created HADB users. In this example, the first HADB user who is created during database initialization is made the user administrator.
2	Database builder	This is an HADB user who is in charge of building databases. The database builder performs the following tasks: • Defining schemas, tables, and indexes • Importing data • Granting access privileges to the database operator and application program user for the created tables
3	Database operator	This is the HADB user who is in charge of operating the database. The database operator performs the following tasks on tables created by the database builder: • Importing data • Re-creating indexes • Collecting cost information • Retrieving data using the SELECT statement
4	Application program user	This is the HADB user who uses an application program to retrieve or update the database. The application program user performs the following tasks on the tables created by the database builder: • Using the SELECT statement to retrieve data • Using the INSERT, UPDATE, or DELETE statement to add, update, or delete data

Note

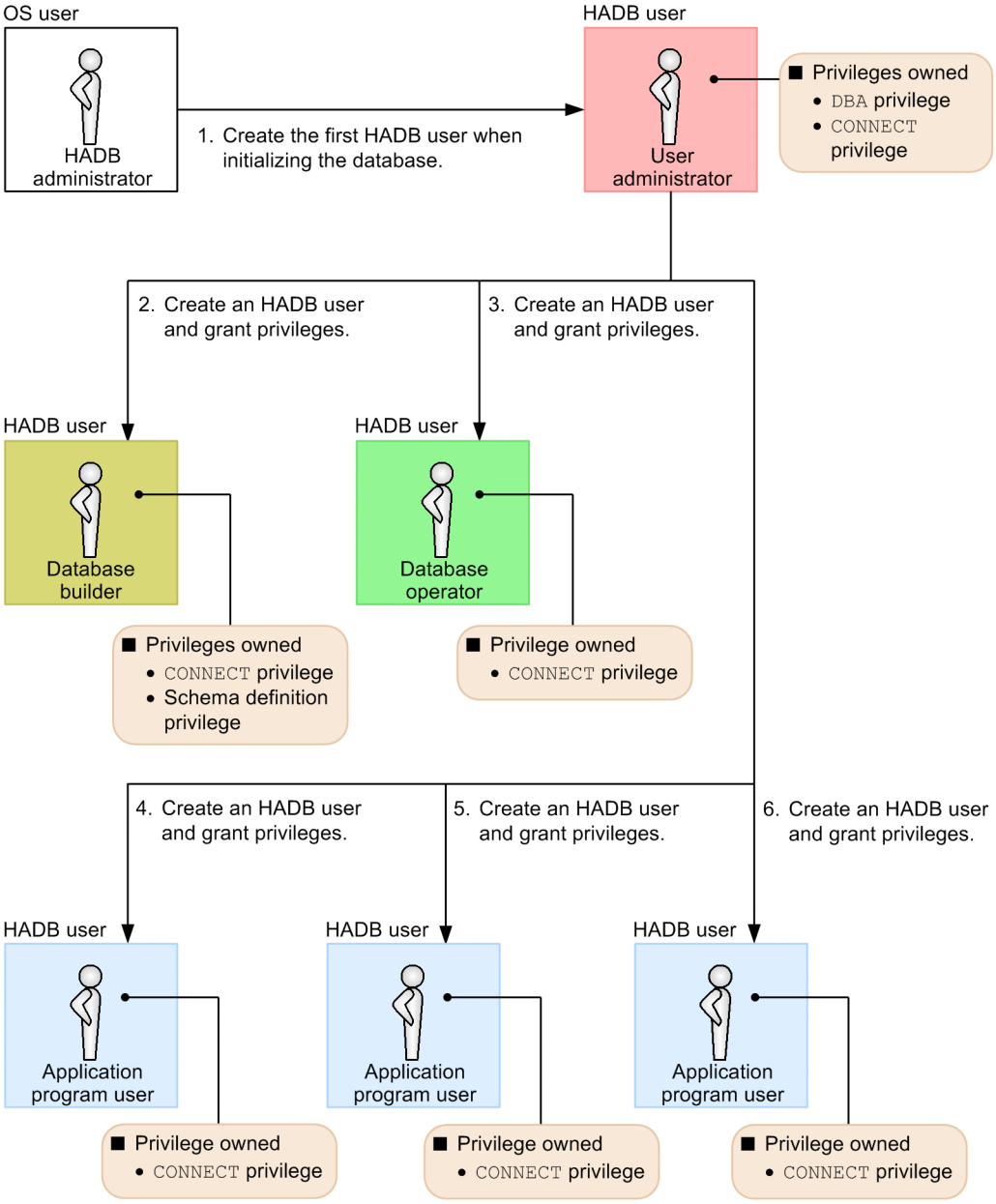
This example does not involve an auditor, which is a role required when using the audit trail facility. For details about the audit trail facility and the role of the auditor, see 2.18 Audit trail facility.

(2) Process of granting privileges to HADB users

This subsection explains the general procedure for granting privileges to HADB users based on the HADB user roles described in (1) Roles of HADB users and operation example.

First, the following figure shows the general procedure that the user administrator follows to create HADB users and grant privileges.

Figure 2-29: General procedure for granting privileges (part 1)



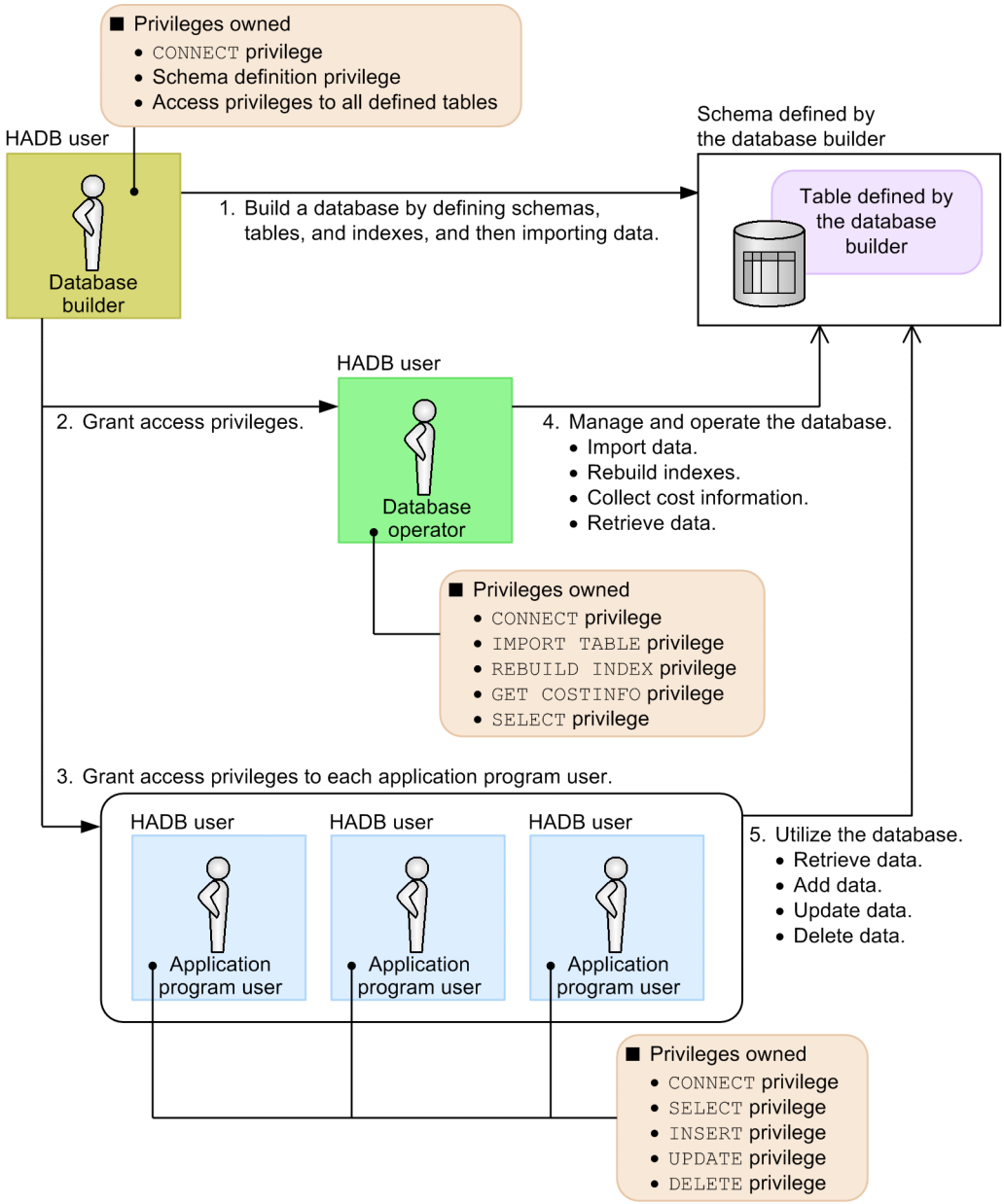
Explanation

The HADB administrator, who is an OS user, performs database initialization. During this process, the user administrator is created as the first HADB user.

Then, the user administrator creates other HADB users and grants the CONNECT privilege and schema definition privilege.

Next, the following figure shows the general procedure for granting access privileges to the database operator and the application program users for access to the tables defined by the database builder.

Figure 2-30: General procedure for granting privileges (part 2)



Explanation

- The database builder defines the tables. Then, the access privileges for the tables are granted to the database operator and application program user. This enables the database operator and the application program users to access the tables defined by the database builder, and to perform tasks allowed by the access privileges they now have.
- When there are multiple database operators, you can grant access privileges with the grant options to a representative database operator in *2. Grant access privileges*. This representative database operator can then grant access privileges to the other database operators.
- You can also grant access privileges with the grant option to a representative application program user in *3. Grant access privileges to each application program user*. This representative application program user can then grant access privileges to other application program users.

Note

- To grant a privilege to an HADB user or to revoke a granted privilege, execute the `GRANT` or `REVOKE` definition SQL statement.
- The first HADB user is created during database initialization. The `DBA` privilege and `CONNECT` privilege are automatically granted to this first HADB user.

2.8 Database access method

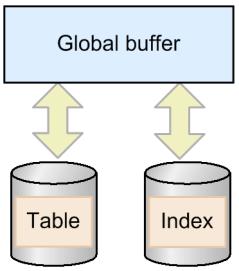
This section explains the database access method, and describes the global buffers used for a database's I/O processing, the database updating method, and the database retrieval method (out-of-order execution).

2.8.1 Global buffers

HADB uses global buffers for a database's I/O processing.

A global buffer is an area that is used for the input and output of data stored in a DB area on a disk. Global buffers are allocated in the shared memory. The following figure shows the global buffer concept.

Figure 2-31: Global buffer concept



A global buffer must always be allocated for any data DB area that stores tables and indexes. You can allocate any single global buffer to one or more DB areas.

2.8.2 Database updating method

HADB uses the write-once updating method as the database updating method.

With the write-once updating method, when a row is deleted it becomes invalid but is not physically deleted from the disk. Likewise, when a row is updated it becomes invalid and a new, updated, row is added. The original row that became invalid is not physically deleted from the disk.

If you delete a row by executing the `TRUNCATE TABLE` data manipulation SQL statement, the segment is released and the row is deleted from the disk.

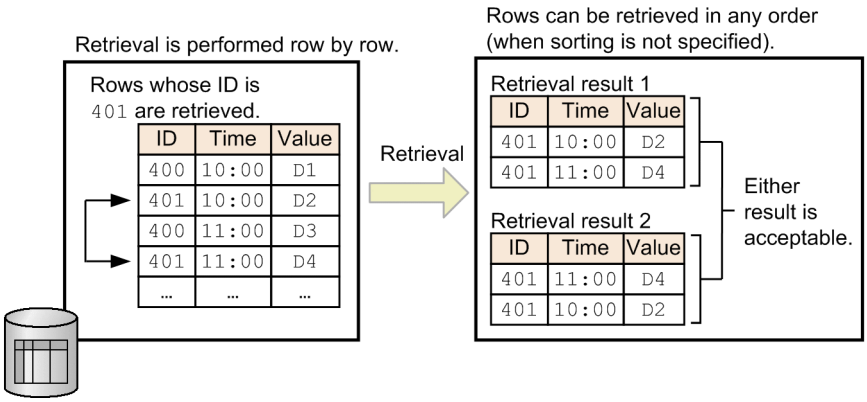
2.8.3 Database retrieval method (out-of-order execution)

HADB utilizes the out-of-order execution method as the database retrieval method. This method is a way of processing that was devised in order to search through a large-scale database at high speeds.

(1) Concept of out-of-order execution

With relational databases, retrieval processing consists of a set operation executed against rows. There are no rules governing the sequence in which retrieval results are returned by a DBMS, except for the final sorting process. The out-of-order execution method was devised to take advantage of this. The following figure shows the concept of out-of-order execution.

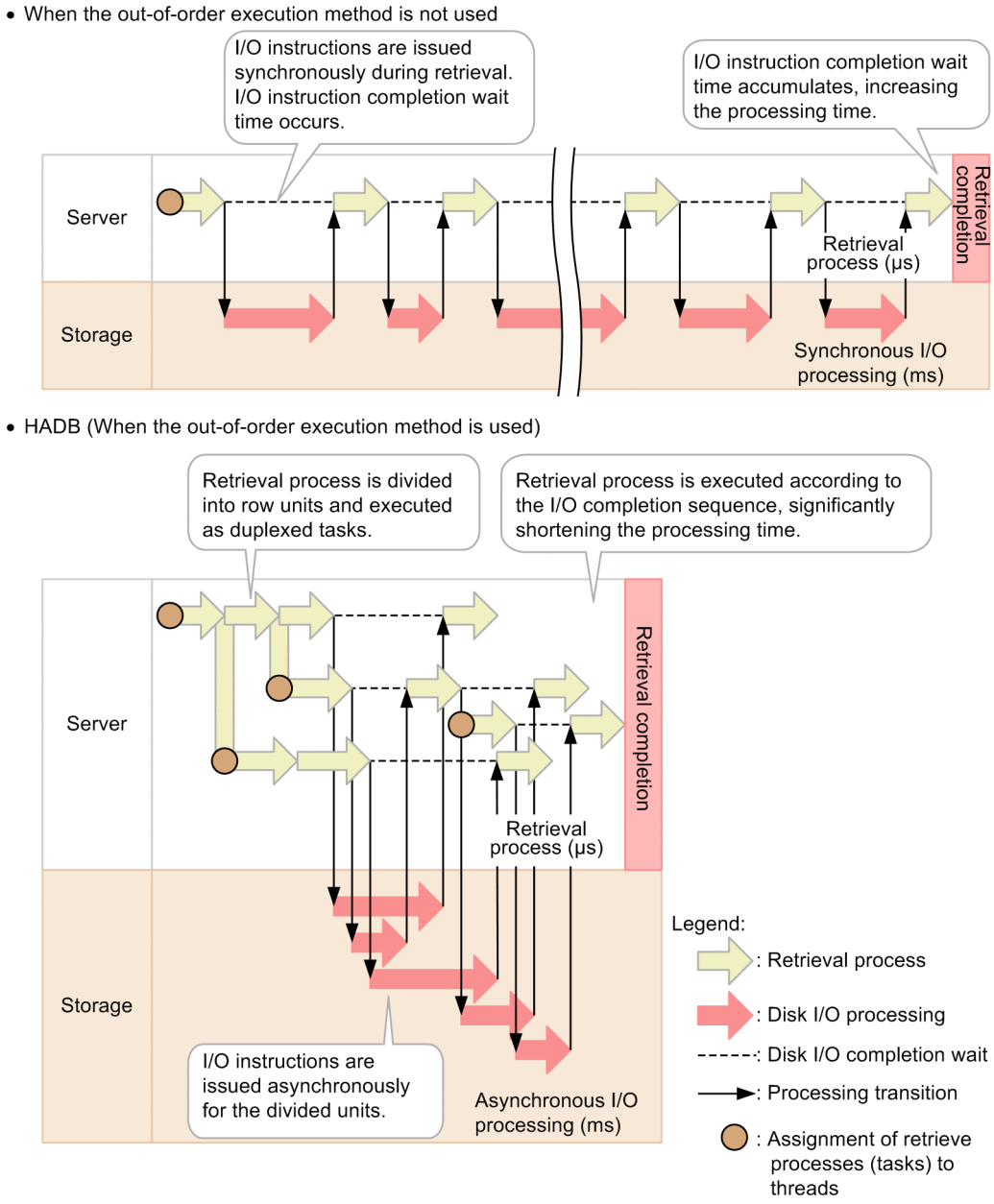
Figure 2-32: Concept of out-of-order execution



(2) Retrieval using the out-of-order execution method

By effectively utilizing hardware resources, the out-of-order execution method can search through a large-scale database at high speeds. The following figure compares retrieval using the out-of-order execution method and other retrieval methods.

Figure 2-33: Retrieval using the out-of-order execution method (comparison with retrieval not using the out-of-order execution method)



When the out-of-order execution method is not used, only one retrieval process is assigned to a thread. Furthermore, a new I/O instruction is issued only after the previously issued I/O instruction is completed. Therefore, you can only issue the number of I/O instructions that does not exceed the upper limit of the I/O performance of the storage device. As a result, as the volume of data in the database becomes larger, the amount of time the system must wait for I/O completion becomes longer and the retrieval takes longer.

With the out-of-order execution method, retrieval processes are divided by rows and assigned to individual threads. Each assigned retrieval process is executed independently in each thread. Additionally, the individual threads can be executed out-of-order based on the progress of I/O processing. That is, after the threads issue I/O instructions, they are processed in parallel and processing can be switched to another thread without having to wait for the completion of other I/O instructions (asynchronous I/O processing). The result is high-speed data retrieval.

(3) Threads with which the out-of-order execution method is used

HADB executes I/O instructions and other processing in parallel by assigning processes not having dependent relationships to multiple threads of the thread types listed below. The out-of-order execution method uses the following types of threads:

- Real threads

  Real threads are managed by the OS (kernel and library). When the CPU has two or more cores, threads allow multiple processes to be executed in parallel.

- Pseudo-threads

  Pseudo-threads are managed by HADB. Although multiple pseudo-threads can be generated in each real thread, two or more pseudo-threads cannot be active simultaneously. By using multiple pseudo-threads, HADB executes processes and I/O instructions in parallel.

(4) Cases in which the out-of-order execution method is not applied

In the following cases, the out-of-order execution method is not applied; standard SQL execution order is used instead:

- If an SQL statement other than retrieval (`SELECT` statement) is being executed
- If `0` is specified as the number of real threads to be used when executing an SQL statement (`adb_sql_exe_max_rthd_num` operand)
- If pseudo-threads cannot be generated (when `0` is specified as the number of pseudo-threads that can be generated in each real thread (`adb_sys_uthd_num` operand))

(5) Relationship between the out-of-order execution method and wait status

This subsection explains the relationship between the out-of-order execution method and the wait status that occurs if the number of processing real threads required for executing an SQL statement cannot be allocated.

(a) When executing SQL statements

If the required number of processing real threads cannot be allocated when an SQL statement is to be executed, the SQL statement is placed in wait status. The SQL statement that is in wait status is executed after the required number of processing real threads are allocated.

If an SQL statement or command is placed in wait status, all the SQL statements that follow that SQL statement or command will be placed in wait status. (This also applies even if the number of processing real threads required for any of the subsequent SQL statements can be allocated.) If this occurs, the subsequent SQL statements will be released from wait status only after the prior SQL statement or command has been released from wait status.

■ When using the client-group facility

If an SQL statement that is executed by an HADB client belonging to a group is placed in wait status, all SQL statements that are executed by other HADB clients belonging to the same group are also placed in wait status. If this occurs, the subsequent SQL statements will be released from wait status only after the prior SQL statement has been released from wait status.

Note that the wait status for an SQL statement that is executed by an HADB client belonging to one group has no effect on SQL statements that are executed by HADB clients belonging to other groups.

For details about the processing real threads used when executing SQL statements, see the description of the adb_sql_exe_max_rthd_num operand in 7.2.2 Operands related to performance (set format).

Note

For details about how to check whether a target SQL statement is in wait status, see 10.8.2 Checking whether the process of allocating processing real threads has gone into wait status.

(b) When executing commands

Processing real threads are used to execute some commands, such as the `adbimport` command. If the required number of processing real threads cannot be allocated, the corresponding command is placed in wait status. For details about the commands to which this applies, see *Targeted commands* in the description of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

If a command or SQL statement is placed in wait status, all the commands that follow that command or SQL statement are also placed in wait status. (This also applies even if the number of processing real threads required for any subsequent command can be allocated). If this occurs, the subsequent commands will be released from wait status only after the prior command or SQL statement has been released from wait status.

■ When using the client-group facility

If a command belonging to a group is placed in wait status, all other commands belonging to the same group are also placed in wait status. If this occurs, the subsequent commands will be released from wait status only after the prior command has been released from wait status.

For details about the processing real threads used during command execution, see the description of each command in the manual *HADB Command Reference*.

Note

For details about how to check whether a target command is in wait status, see 10.8.2 Checking whether the process of allocating processing real threads has gone into wait status.

2.9 Transaction control

This section explains how transactions are controlled by HADB.

2.9.1 Start and end of a transaction

An HADB transaction starts when the pre-processing of an SQL statement is executed or a statement handle to an SQL statement is allocated. An HADB transaction ends when a synchronization point (commit or rollback) is set or when the application is disconnected from the HADB server.

2.9.2 Transaction isolation levels supported by HADB

HADB supports the following transaction isolation levels:

- `READ COMMITTED`
- `REPEATABLE READ`

(1) READ COMMITTED

Data that has already been committed when an SQL statement is executed (when a cursor is opened for retrieval) becomes the processing target of the SQL statement. Therefore, if the same `SELECT` statement is executed multiple times within the same transaction, data updating by another transaction might cause the execution result of the `SELECT` statement to vary.

(2) REPEATABLE READ

If the same SQL statement with the same search condition is executed repeatedly within a transaction for which the `REPEATABLE READ` isolation level is set, the data that becomes the processing target of that SQL statement is always the same regardless of any processing by other transactions.

The data to be processed by the SQL statement changes according to the conditions shown in the following tables.

- Table 2-9: Relationship between chunks that store data and the data to be processed by an SQL statement
- Table 2-10: Relationship between a transaction that updated the data within a chunk and the data to be processed by an SQL statement

Table 2-9: Relationship between chunks that store data and the data to be processed by an SQL statement

No.	Chunks that store data	Data to be processed by an SQL statement
1	• Chunk that was created during base table definition • Chunk that was created during execution of the `TRUNCATE TABLE` statement • Chunk that was created during execution of the `adbimport` command with the `-d` option specified • Chunk that was created during execution of the `adbimport` command, which uses the background-import facility before executing a local transaction • Chunk that was created during execution of the `adbmergechunk` command (merge-target chunk)	For details, see Table 2-10: Relationship between a transaction that updated the data within a chunk and the data to be processed by an SQL statement.

No.	Chunks that store data	Data to be processed by an SQL statement
	• Chunk whose status was changed by the `adbchgchunkstatus` command • Archived chunk that was created when the `adbimport` command, which uses the background-import facility, was executed in a local transaction	
2	Non-archived chunk that was created when the `adbimport` command, which uses the background-import facility, was executed in a local transaction	None#

#

If the following two commands are executed in a local transaction, the data to be processed by SQL statements changes as shown in Table 2-9: Relationship between chunks that store data and the data to be processed by an SQL statement and Table 2-10: Relationship between a transaction that updated the data within a chunk and the data to be processed by an SQL statement.

- The `adbimport` command, which uses the background-import facility
- The `adbmergechunk` command, which merges multiple chunks, including the chunk created during execution of the `adbimport` command

The following figure shows an example.

Figure 2-34: Changes in the data to be processed by the local transaction in connection with command execution

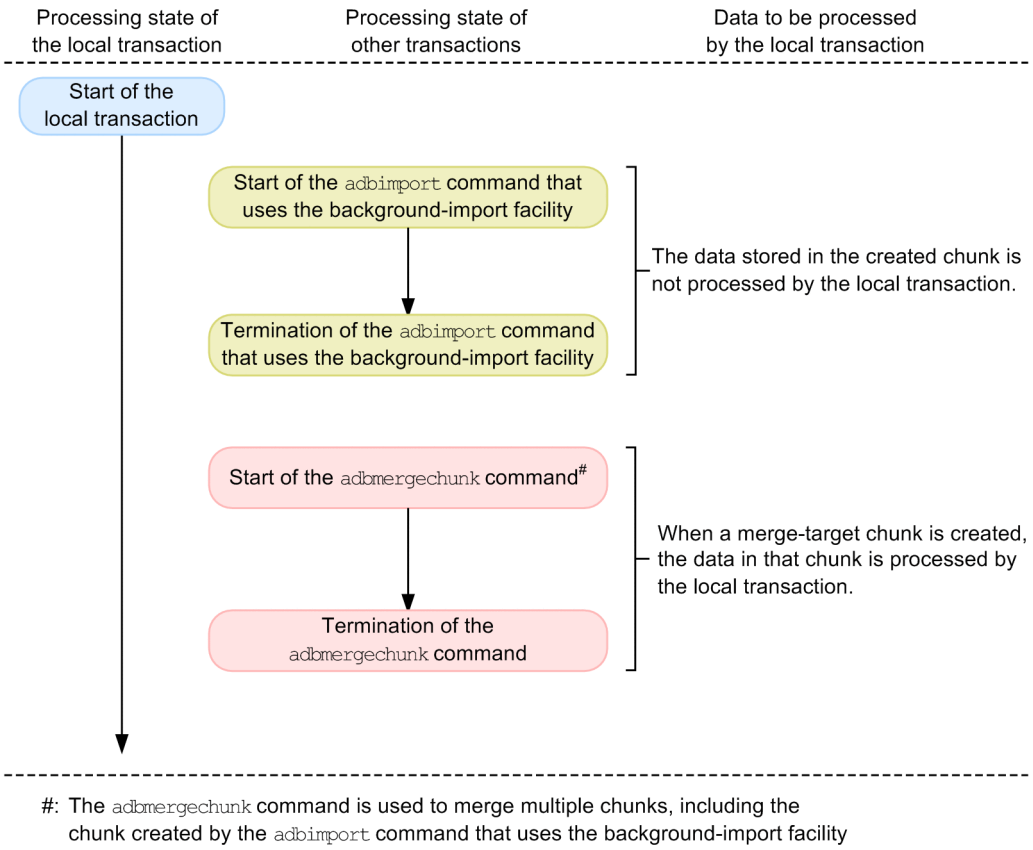


Table 2-10: Relationship between a transaction that updated the data within a chunk and the data to be processed by an SQL statement

No.	Transaction that updated the data within a chunk		Data that becomes the processing target of the SQL statement
1	Local transaction		All data
2	Other transaction	Updating with an SQL statement: • `INSERT` statement • `UPDATE` statement • `DELETE` statement	Data that had already been committed when the local transaction started
3		Adding data with the `adbimport` command	Data when the `adbimport` command terminates normally
4		Updating with the `PURGE CHUNK` statement	Data when the `PURGE CHUNK` statement terminates normally
5		Updating with the `TRUNCATE TABLE` statement	Data when the `TRUNCATE TABLE` statement terminates normally
6		Changing the chunk status with the `adbchgchunkstatus` command	Data when the `adbchgchunkstatus` command terminates normally

Important

If a retrieval with a chunk ID specified (for example, the `#GETDATA` subcommand of the `adbsql` command) is executed, the transaction isolation level might not become `REPEATABLE READ`. For details, see (4) Chunks that are retrieved during execution of the adbmergechunk command in 11.4.9 Merging chunks (to reduce the number of chunks).

2.9.3 Concurrent execution control of transactions

This subsection explains the processing method used when multiple transactions access the same row concurrently.

(1) When transactions A and B reference the same row

Transactions A and B are not mutually affected.

(2) When transaction B references a row that has been updated by transaction A

The contents of the row to be referenced by transaction B are determined by the status of transaction A and the transaction isolation level of transaction B. The following table shows the row contents referenced by transaction B.

Table 2-11: Contents of the row referenced by transaction B

No.	Status of transaction A	Transaction isolation level of transaction B	
		READ COMMITTED	REPEATABLE READ
1	Being executed	Row's contents before being updated by transaction A	Row's contents before being updated by transaction A

No.	Status of transaction A	Transaction isolation level of transaction B	
		READ COMMITTED	REPEATABLE READ
2	Already committed	Row's contents after being updated by transaction A	

(3) When transaction B attempts to update a row that has been updated by transaction A

If transaction B attempts to update a row that has been updated by transaction A which is still executing, transaction B is rolled back.

2.9.4 Synchronization point processing of a transaction

When a transaction is committed, the contents updated by that transaction are committed.

If a transaction is rolled back, the contents updated by that transaction become invalid, and the database returns to its status before the transaction started.

2.9.5 Transaction access modes

Transaction access modes are used to control the SQL statements that are executed in a transaction. HADB supports the following transaction access modes:

- Read/write mode
- Read-only mode

> **Note**
>
> You can use either the `adb_clt_trn_access_mode` operand in the client definition or the connection attribute to specify a transaction access mode. For details about the client definition and the connection attribute, see the *HADB Application Development Guide*.

(1) Read/write mode

When a transaction is started in the read/write transaction access mode, that transaction becomes a *read/write transaction*.

A read/write transaction can execute all SQL statements.

(2) Read-only mode

When a transaction is started in read-only transaction access mode, that transaction becomes a *read-only transaction*.

A read-only transaction cannot execute definition SQL statements or update SQL statements.

A read-only transaction can execute the SQL statement listed in the following.

■ SQL statement that can be executed in a read-only transaction:

- `SELECT` statement

2.10 Locking

This section explains how HADB applies locks.

2.10.1 Lock modes

The following describes the lock modes provided by HADB. The following also describes whether two competing transactions can run concurrently depending on the combination of lock modes.

(1) Lock modes

Several lock modes are available on the HADB server.

- *Exclusive mode* **(***EXCLUSIVE***) (***EX***)**
  In exclusive mode, a transaction that secured locked resources can exclusively use them. While locked resources are secured in exclusive mode by a transaction, other transactions cannot reference or update them.
- *Protected update mode* **(***PROTECTED UPDATE***) (***PU***)**
  In protected update mode, a transaction that secured locked resources can update them. While locked resources are secured in protected update mode by a transaction, other transactions can only reference them.
- *Shared update mode* **(***SHARED UPDATE***) (***SU***)**
  In shared update mode, a transaction that secured locked resources can update them. While locked resources are secured in shared update mode by a transaction, other transactions can reference and update them.
- *Shared retrieval mode* **(***SHARED RETRIEVE***) (***SR***)**
  In shared retrieval mode, a transaction that secured locked resources can reference them. While locked resources are secured in shared retrieval mode by a transaction, other transactions can reference and update them.

(2) Relationship of concurrent execution between lock modes (behavior of transactions competing for locked resources)

When multiple transactions attempt to reserve the same locked resource, they compete against each other for the locked resource. When competition of two transactions occurs, the combination of the lock modes between the transactions that earlier and later attempted to reserve the resource will determine whether the transactions can run concurrently.

The following table shows whether two transactions that compete for the same locked resource can run concurrently depending on the combination of lock modes.

Table 2-12: Relationship between the lock modes of competing transaction

Lock mode of the earlier transaction (T1)	Lock mode of the later transaction (T2)			
	SR	SU	PU	EX
Shared retrieval mode (SR)	T1/T2	T1/T2	T1/T2	T1
Shared update mode (SU)	T1/T2	T1/T2	T1	T1
Protected update mode (PU)	T1/T2	T1	T1	T1
Exclusive mode (EX)	T1	T1	T1	T1

Legend:

T1/T2: Both the earlier and later transactions can run concurrently.

T1: Only the earlier transaction can run.

In the preceding table, if the relationship between lock modes is T1, the later transaction cannot reserve the locked resource. Therefore, the later transaction will take either of the following actions:

- The later transaction results in an error and is rolled back.
- The later transaction waits until the locked resource becomes able to be reserved.#

For details, see 2.10.4 Locked resources that are reserved and their lock modes.

#

While a transaction waits for a locked resource to be freed, if the following phenomenon occurs on a locked resource that the transaction has already reserved, the transaction will result in an error:

- A transition of the lock mode arose for the reserved locked resource, but the lock mode could not be changed because conflict with another transaction occurred.

Note, however, that in the following cases, an error does not occur as an extension of the processing that archives a chunk by using the `adbarchivechunk` command. In these cases, the transaction will wait until the transition of the lock mode succeeds.

- The lock mode is changed from shared retrieval mode (SR) to exclusive mode (EX) when the pre-processing table has been locked.
- The lock mode is changed from protected update mode (PU) to exclusive mode (EX) when the processing-target table has been locked.

(3) Transition of the lock mode

If a resource is locked twice successively in different lock modes within a transaction, the lock mode will change as shown in the following table.

Table 2-13: Transition of the lock mode

Lock mode used first	Lock mode used next			
	SR	SU	PU	EX
Shared retrieval mode (SR)	--	Changes to SU	Changes to PU	Changes to EX#1, #2
Shared update mode (SU)	--	--	Changes to PU	Changes to EX#2
Protected update mode (PU)	--	--	--	Changes to EX#2
Exclusive mode (EX)	Changes to SR#3	--	Changes to PU#4	--

Legend:

--: The mode does not change.

#1

The lock mode changes to exclusive mode (EX) when the following locked resources are reserved as an extension of processing of an `ALTER VIEW` statement. In other cases, an error occurs when reserving the locked resources.

- Pre-processing table

#2

The lock mode changes to exclusive mode (EX) when the following locked resources are reserved as an extension of the `adbarchivechunk` command processing. In other cases, an error occurs when a locked resource is reserved.

- Pre-processing table

- DB area (processing that stores the processing-target table and indexes)
- Processing-target table

#3

The lock mode changes to shared retrieval mode (SR) when the following locked resources are reserved as an extension of the `adbarchivechunk` command processing. In other cases, the lock mode does not change.

- Pre-processing table
- DB area (processing that stores the processing-target table and indexes)

#4

The lock mode changes to protected update mode (PU) when the following locked resource is reserved as an extension of the `adbarchivechunk` command processing. In other cases, the lock mode does not change.

- Processing-target table

2.10.2 Locked resources

The following table describes locked resources.

For details about the lock modes for individual locked resources, see Table 2-15: Locked resources that are reserved and their lock modes.

Table 2-14: Locked resources

No.	Locked resource	Description
1	Single connection	This is a locked resource that is reserved when an application program connects to the HADB server or when one of the following commands is executed: • `adbimport` command • `adbidxrebuild` command • `adbgetcst` command • `adbdbstatus` command • `adbmodarea` command • `adbexport` command • `adbmergechunk` command • `adbchgchunkcomment` command • `adbchgchunkstatus` command • `adbarchivechunk` command • `adbunarchivechunk` command • `adbreorgsystemdata` command • `adbclientdefmang` command • `adbsyndict` command • `adbaudittrail` command • `adbconvertaudittrailfile` command • `adbmodbuff` command It is locked to prevent the `adbmodarea` command from executing concurrently with application programs or other commands.
2	Database access	This is a locked resource that is reserved at the following times: 1. When the maintenance processing of the updated-row columnizing facility is performed 2. When the transaction of an application program is executed 3. When a command that connects to the HADB server is executed

No.	Locked resource		Description
			This locked resource is used to control locks so that the maintenance processing in 1 and the application program transaction in 2 do not occur concurrently. Locks are also controlled so that the maintenance processing in 1 and the command execution in 3 do not occur concurrently. The control prevents the maintenance processing from starting while a transaction or command is running so that the processing performance of the transaction or command is not affected. If a transaction or command starts while maintenance processing is in progress, the maintenance processing is interrupted. This locked resource is of no concern when disabling the updated-row columnizing facility. For details about the commands that connect to the HADB server, see *List of commands* in *List of Commands and Common Rules* in the manual *HADB Command Reference*.
3	Dictionary		This is a locked resource that is reserved when a definition SQL statement is executed.
4	Pre-processing table		This is a locked resource that is reserved for each processing-target table at the following times: • When a data manipulation SQL statement is preprocessed • When a definition SQL statement is executed • When a command is executed
5	DB area	Data DB area	This is a locked resource that is reserved when data in a data DB area is updated. It is locked to prevent the data in the same data DB area from being updated concurrently by an application and the command.
6		Dictionary DB area	This is a locked resource that is reserved at the following times: • When a definition SQL statement is executed • When the `adbdbstatus` command is executed However, even when the `adbdbstatus` command is executed, this locked resource is not reserved in the following cases: • When summary information for DB areas is output • When information is output by specifying the `--shared-lock` option • When information about the need for reorganization is output
7		System-table DB area	This is a locked resource that is reserved when data in a system-table DB area is accessed.
8	Table	Processing-target table	This is a locked resource that is reserved when the table to be processed is accessed.
9		Dictionary table	This is a locked resource that is reserved when a dictionary table (base table) is accessed.
10		System table (cost information)	This is a locked resource that is reserved when a system table (base table) that stores cost information is accessed. The target system tables (base tables) are as follows: • `STATUS_TABLES` • `STATUS_COLUMNS` • `STATUS_INDEXES`
11		System table (chunk information)	This is a locked resource that is reserved when a system table (base table) that stores chunk information is accessed. The target system table (base table) is `STATUS_CHUNKS`.
12		System table (synonym dictionary information)	This is a locked resource that is reserved when a system table (base table) that stores synonym dictionary information is accessed. The target system table (base table) is `STATUS_SYNONYM_DICTIONARIES`.

2.10.3 Period during which locked resources are reserved

Locked resources reserved by a transaction are released when the transaction is committed or rolled back.

Note that when one of the following commands is executed, locked resources are reserved when the command starts and are released when the command terminates:

- `adbimport` command
- `adbidxrebuild` command
- `adbgetcst` command
- `adbdbstatus` command
- `adbmodarea` command
- `adbexport` command
- `adbmergechunk` command
- `adbchgchunkcomment` command
- `adbchgchunkstatus` command
- `adbarchivechunk` command
- `adbunarchivechunk` command
- `adbreorgsystemdata` command
- `adbclientdefmang` command
- `adbsyndict` command
- `adbaudittrail` command
- `adbconvertaudittrailfile` command
- `adbmodbuff` command

2.10.4 Locked resources that are reserved and their lock modes

The following table shows the locked resources that are reserved and their lock modes.

For details about each locked resource, see Table 2-14: Locked resources. For examples of locks performed for each type of processing, see 2.10.7 Examples of locks.

Table 2-15: Locked resources that are reserved and their lock modes

Processing		Locked resources										
		Single connection[#16]	Dictionary	Pre-processing table	DB area			Table				
					Data DB area	Dictionary DB area	System-table DB area	Processing-target table	Dictionary table	System table (cost information)	System table (chunk information)	System table (synonym dictionary information)
Definition SQL execution	When a definition SQL statement other than the following SQL statements is executed • CREATE VIEW statement • ALTER VIEW statement • CREATE AUDIT statement • DROP AUDIT statement	--	EX (WAIT)	EX	EX	SR	SR	EX	EX	SR	SR[#2]	--
	When the CREATE VIEW statement is executed	--	EX (WAIT)	EX[#1]	--	SR	SR	--	SR (WAIT) and EX[#14]	SR	--	SR
	When the ALTER VIEW statement is executed	--	EX (WAIT)	SR (WAIT) and EX[#1, #18]	--	SR	SR	--	SR (WAIT) and EX[#14]	SR	--	SR
	When the CREATE AUDIT	--	EX (WAIT)	EX	EX	SR	SR	--	EX	SR	--	--

Processing		Locked resources										
		Single connection[#16]	Dictionary	Pre-processing table	DB area			Table				
					Data DB area	Dictionary DB area	System-table DB area	Processing-target table	Dictionary table	System table (cost information)	System table (chunk information)	System table (synonym dictionary information)
	statement or DROP AUDIT statement is executed											
Retrieval SQL execution		--	--	SR (WAIT)[#3]	--	--	--	SR (WAIT)[#4, #5, #6]	SR (WAIT)	SR	SR[#7]	SR
Execution of the INSERT, UPDATE, or DELETE statement	Table being updated	--	--	SR (WAIT)[#3]	SU[#4, #5, #6]	--	--	SU	SR (WAIT)	SR	SR[#7]	SR
	Table not being updated	--	--	SR (WAIT)[#3]	--	--	--	SR (WAIT)[#4, #5, #6]	SR (WAIT)	SR	SR[#7]	SR
Execution of the TRUNCATE TABLE statement		--	--	SR (WAIT)	EX	--	SR[#17]	EX	SR (WAIT)	SR	SR[#17]	--
Execution of the PURGE CHUNK statement	Table being updated	--	--	SR (WAIT)	EX	--	SR	EX	SR (WAIT)	SR	SR[#15]	SR
	Table not being updated[#8]	--	--	SR (WAIT)[#3]	--	--	--	SR	SR (WAIT)	SR	SR[#15]	SR
Execution of the adbimport command	When the background-import facility is used	SR	--	SR (WAIT)	PU	--	SR	SR (WAIT)	SR (WAIT)	SR	SR	--
	When the background-import facility is not used	SR	--	SR (WAIT)	EX	--	SR[#9]	EX (WAIT)	SR (WAIT)	SR	SR[#9]	--
Execution of the adbidxrebuil	When executed after background import	SR	--	SR (WAIT)	EX	--	SR	SR (WAIT)	SR (WAIT)	SR	SR	--

Processing		Locked resources										
		Single connection[#16]	Dictionary	Pre-processing table	DB area			Table				
					Data DB area	Dictionary DB area	System-table DB area	Processing-target table	Dictionary table	System table (cost information)	System table (chunk information)	System table (synonym dictionary information)
`d` command	processing was interrupted[#10]											
	When executed other than after background import processing was interrupted	SR	--	SR (WAIT)	EX	--	--	EX (WAIT)	SR (WAIT)	SR	--	--
Execution of the `adbgetcst` command		SR	--	SR (WAIT)	EX	--	SR	SR (WAIT)	SR (WAIT)	SR	--	--
Execution of the `adbdbstatus` command	When summary information for DB areas is output	SR	--	--	--	--	--	--	SR (WAIT)	--	--	--
	When information is output by specifying the `--shared-lock` option or information about the need for reorganization is output	SR	--	SR[#19]	--	--	--	SR[#19]	SR (WAIT)	--	--	--
	Other	SR	--	SR[#19]	EX	EX	EX	--	SR (WAIT)	SR	SR	--
Execution of the `adbmodarea` command		EX	--	--	--	--	--	--	SR (WAIT)	--	--	--

Processing		Locked resources										
		Single connection[#16]	Dictionary	Pre-processing table	DB area			Table				
					Data DB area	Dictionary DB area	System-table DB area	Processing-target table	Dictionary table	System table (cost information)	System table (chunk information)	System table (synonym dictionary information)
Execution of the `adbexport` command[#11]		SR	--	SR (WAIT)[#3]	--	--	--	SR (WAIT)	SR (WAIT)	SR	SR[#7]	SR
Execution of the `adbmergechunk` command		SR	--	SR (WAIT)	EX	--	SR	SU (WAIT)	SR (WAIT)	SR	SR	--
Execution of the `adbchgchunkcomment` command		SR	--	SR (WAIT)	EX	--	SR	SR (WAIT)	SR (WAIT)	SR	SR	--
Execution of the `adbchgchunkstatus` command		SR	--	EX (WAIT)	EX	--	SR	EX (WAIT)	SR (WAIT)	SR	SR	--
Execution of the `adbarchivechunk` command		SR	--	SR (WAIT) and EX (WAIT)[#12, #13]	SR and EX[#12]	--	SR	PU and EX (WAIT)[#12, #13]	SR (WAIT)	SR	SR	--
Execution of the `adbunarchivechunk` command		SR	--	SR (WAIT)	EX	--	SR	EX (WAIT)	SR (WAIT)	SR	SR	--
Execution of the `adbreorgsystemdata` command		SR	--	SR (WAIT)	--	--	EX	PU	SR (WAIT)	SR	--	--
Execution of the `adbclientdefmang` command		SR	--	--	--	--	--	--	SR (WAIT)	--	--	--
Execution of the `adbsyndict` command	When registering, updating, or deleting a synonym dictionary	SR	--	--	--	--	SR	--	--	--	--	SR
	When synchronizing a synonym dictionary file, outputting a synonym	SR	--	--	--	--	--	--	--	--	--	SR

Processing	Locked resources										
	Single connection[#16]	Dictionary	Pre-processing table	DB area			Table				
				Data DB area	Dictionary DB area	System-table DB area	Processing-target table	Dictionary table	System table (cost information)	System table (chunk information)	System table (synonym dictionary information)
list, or deleting unnecessary files											
Execution of the `adbaudittrail` command	SR	--	--	--	--	--	--	SR (WAIT)	--	--	--
Execution of the `adbconvertaudittrailfile` command	SR	--	--	--	--	--	--	SR (WAIT)	--	--	--
Execution of the `adbmodbuff` command	SR	--	--	--	--	--	--	--	--	--	--

Legend:

--: Locked resources are not reserved.

EX: Locked resources are reserved in exclusive mode (EX).

PU: Locked resources are reserved in protected update mode (PU).

SU: Locked resources are reserved in shared update mode (SU).

SR: Locked resources are reserved in shared retrieval mode (SR).

WAIT: Stays in wait status until the locked resources can be reserved.

Note

- For details about the behavior of HADB when multiple transactions attempt to reserve the same locked resource, see (2) Relationship of concurrent execution between lock modes (behavior of transactions competing for locked resources) in 2.10.1 Lock modes.
- Database access of the locked resource is not explained in the preceding table. For details about the purpose and functionality of database access of the locked resource, see 2.10.2 Locked resources.

#1

The locked resources are reserved in the shared retrieval mode (SR) for the underlying table of a viewed table. In this case, if the locked resources cannot be reserved, the transaction is placed in WAIT status until it can reserve the locked resources.

#2

The locked resources are reserved when the `CREATE TABLE` statement is executed to create a multi-chunk table and when the `DROP TABLE` statement is executed to delete a multi-chunk table.

The locked resources are also reserved when a definition SQL statement (such as DROP SCHEMA) is executed and, as a result of executing the statement, a multi-chunk table is deleted.

#3

If the processing-target table is an archivable multi-chunk table, the processing-target table and system table (STATUS_CHUNKS) are locked. Note, however, that this does not apply when the INSERT statement is executed for the processing-target table.

#4

If there is a ResultSet object that was created with HOLD_CURSORS_OVER_COMMIT specified for holdability within the same transaction, the locked resource is not released even if the transaction is committed. Committing the transaction after closing the ResultSet object releases the locked resource. Note that the locked resource is released if the transaction is rolled back.

#5

If an error occurs during execution of the SQL statement, the locked resources that were reserved by the transaction are not released until the transaction terminates.

#6

If an error occurs in a transaction that reserves multiple locked resources because of locked resource contention, the locked resources that had already been reserved before the error occurred are not released until the transaction terminates.

#7

The locked resources are reserved only if the processing-target table is an archivable multi-chunk table.

#8

This indicates the tables contained in a subquery when the PURGE CHUNK statement is executed with the subquery specified in the search condition.

#9

The locked resources are reserved only when the adbimport command with the -d or -m option specified is executed for a multi-chunk table.

#10

This is the case where the adbidxrebuild command is executed with the --create-temp-file option specified after the adbimport command to which the background-import facility was applied (with the -b option specified) has been interrupted.

#11

When the adbexport command is executed, a retrieval SQL statement is also executed as part of extended processing. Consequently, locked resources that are similar to those reserved by a retrieval SQL statement are also reserved.

#12

Transition of the lock mode occurs as a result of executing the adbarchivechunk command. After the lock mode changes, the locked resources are reserved in exclusive mode (EX).

#13

If transition of the lock mode occurs when the adbarchivechunk command is executed, the transition might cause a wait for reservation of the locked resources. If a wait occurs, other connections might reserve the locked resources first.

#14

The locked resources are reserved in shared retrieval mode (SR) during preprocessing. In this case, if the locked resources cannot be reserved, the transaction is placed in WAIT status until it can reserve the locked resources.

When the SQL statement is executed, the locked resources are reserved in exclusive mode (EX).

#15

If an archivable multi-chunk table is included in the tables that are not to be updated, the locked resources are reserved during preprocessing. In other cases, the locked resources are reserved when the SQL statement is executed.

#16

The locked resources are reserved when the relevant command is executed. The locked resources are also reserved when the application program connects to the HADB server. When the application program is connected to the HADB server, the locked resources are reserved in shared retrieval mode (SR).

#17

The locked resources are reserved only if the processing-target table is a multi-chunk table.

#18

The lock mode changes when the `ALTER VIEW` statement is executed. After the lock mode changes, the locked resources are reserved in exclusive mode (EX).

#19

If the target of the `adbdbstatus` command is an index, the lock is obtained for the table for which the index is defined.

If the target of the `adbdbstatus` command is a DB area, the lock is obtained for the tables stored in that DB area, and the tables stored in the DB area for which an index is defined.

The following explains how to interpret the above table using an example in which a retrieval SQL statement is executed.

When a retrieval SQL statement is executed

When a retrieval SQL statement is executed for an archivable multi-chunk table, the following locked resources are reserved in shared retrieval mode (SR):

- Pre-processing table
- Processing-target table
- Dictionary table
- System table (cost information)
- System table (chunk information)
- System table (synonym dictionary information)

Processing to reserve the preceding locked resources in shared retrieval mode (SR), shared update mode (SU), and protected update mode (PU) can be executed. For example, `adbgetcst` command can be executed.

However, processing to reserve the preceding locked resources in exclusive mode (EX) cannot be executed. For example, definition SQL statements and the `TRUNCATE TABLE` statement cannot be executed.

2.10.5 Base tables for which a lock is obtained when dictionary tables and system tables are referenced

When SQL statements are executed to reference dictionary tables and system tables, a lock is obtained for the dictionary tables (base tables) and system tables (base tables). The following table describes the base tables for which a lock is obtained when dictionary tables and system tables are referenced.

Table 2-16: Base tables for which a lock is obtained when dictionary tables and system tables are referenced

Dictionary table and system table to be referenced	Dictionary table (base table) and system table (base table) for which a lock is obtained																							
	SQL_TABLES	SQL_COLUMNS	SQL_INDEXES	SQL_DIV_TABLE	SQL_DIV_INDEX	SQL_DBAREAS	SQL_SCHEMATA	SQL_VIEWS	SQL_VIEW_TABLE_USAGE	SQL_VIEW_OBJECT	SQL_DEFINE_SOURCE	SQL_DEFINE_ENVIRONMENT	SQL_USERS	SQL_TABLE_CONSTRAINTS	SQL_INDEX_COLINF	SQL_KEY_COLUMN_USAGE	SQL_REFERENTIAL_CONSTRAINTS	SQL_TABLE_PRIVILEGES	SQL_AUDITS	STATUS_TABLES	STATUS_COLUMNS	STATUS_INDEXES	STATUS_CHUNKS	STATUS_SYNONYM_DICTIONARIES
SQL_TABLES	P/E	P	P	P	P	--	--	P/E	--	P	--	--	E	--	--	--	--	P/E	--	P	P	P	--	--
SQL_COLUMNS	P/E	P/E	P	P	P	--	--	P/E	--	P	--	--	--	--	--	--	--	P/E	--	P	P	P	--	--
SQL_INDEXES	P	P	P/E	P	P	--	--	P	--	P	--	--	E	--	--	--	--	P/E	--	P	P	P	--	--
SQL_DIV_TABLE	P	P	P	P/E	P	--	--	P	--	P	--	--	--	--	--	--	--	P/E	--	P	P	P	--	--
SQL_DIV_INDEX	P	P	P	P	P/E	--	--	P	--	P	--	--	--	--	--	--	--	P/E	--	P	P	P	--	--
SQL_DBAREAS	P	P	P	P	P	E	--	P	--	P	--	--	--	--	--	--	--	P	--	P	P	P	--	--
SQL_SCHEMATA	P	P	P	P	P	--	E	P	--	P	--	--	E	--	--	--	--	P/E	--	P	P	P	--	--
SQL_VIEWS	P	P	P	P	P	--	--	P/E	--	P	--	--	--	--	--	--	--	P/E	--	P	P	P	--	--
SQL_VIEW_TABLE_USAGE	P	P	P	P	P	--	--	P/E	E	P	--	--	E	--	--	--	--	P/E	--	P	P	P	--	--
SQL_DEFINE_SOURCE	P	P	P	P	P	--	--	P/E	--	P	E	--	--	--	--	--	--	P/E	--	P	P	P	--	--
SQL_DEFINE_ENVIRONMENT	P	P	P	P	P	--	--	P/E	--	P	--	E	--	--	--	--	--	P/E	--	P	P	P	--	--
SQL_USERS	P	P	P	P	P	--	--	P	--	P	--	--	E	--	--	--	--	P	--	P	P	P	--	--
SQL_TABLE_CONSTRAINTS	P	P	P	P	P	--	--	P	--	P	--	--	--	E	--	--	--	P/E	--	P	P	P	--	--
SQL_INDEX_COLINF	P	P	P	P	P	--	--	P	--	P	--	--	--	--	E	--	--	P/E	--	P	P	P	--	--
SQL_KEY_COLUMN_USAGE	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	E	--	P/E	--	P	P	P	--	--
SQL_REFERENTIAL_CONSTRAINTS	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	--	E	P/E	--	P	P	P	--	--
SQL_TABLE_PRIVILEGES	P/E	P	P	P	P	--	--	P/E	--	P	--	--	--	--	--	--	--	P/E	--	P	P	P	--	--
SQL_AUDITS	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	--	--	P	E	P	P	P	--	--
STATUS_TABLES	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	--	--	P/E	--	P/E	P	P	--	--
STATUS_COLUMNS	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	--	--	P/E	--	P/E	P/E	P	--	--
STATUS_INDEXES	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	--	--	P/E	--	P	P	P/E	--	--
STATUS_CHUNKS	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	--	--	P/E	--	P	P	P	E	--
STATUS_SYNONYM_DICTIONARIES	P	P	P	P	P	--	--	P	--	P	--	--	--	--	--	--	--	P	--	P	P	P	--	E

Legend:

P/E: The resource is locked in shared retrieval mode (SR) during preprocessing and execution of the SQL statement. If the target resource has already been locked in exclusive mode (EX) when the SQL statement is executed, the statement is placed in wait status until the resource can be reserved.

E: The resource is locked in shared retrieval mode (SR) when the SQL statement is executed. If the target resource has already been locked in exclusive mode (EX), the statement is placed in wait status until the resource can be reserved.

P: The resource is locked in shared retrieval mode (SR) during preprocessing of the SQL statement.

--: No lock is obtained for the target resource.

2.10.6 Dictionary tables and system tables locked when a database is referenced or updated

When an SQL statement or command is executed to reference or update a database, the dictionary tables (base tables) and system tables (base tables) are locked. The following two tables show the dictionary tables and system tables that are locked when a database is referenced or updated.

Table 2-17: Dictionary tables and system tables locked when a database is referenced or updated (1)

Processing that references or updates a database (executed SQL statement or command)			Locked dictionary tables (base tables) and system tables (base tables)																							
			SQL_TABLES	SQL_COLUMNS	SQL_INDEXES	SQL_DIV_TABLE	SQL_DIV_INDEX	SQL_DBAREAS	SQL_SCHEMATA	SQL_VIEWS	SQL_VIEW_TABLE_USAGE	SQL_VIEW_OBJECT	SQL_DEFINE_SOURCE	SQL_DEFINE_ENVIRONMENT	SQL_USERS	SQL_TABLE_CONSTRAINTS	SQL_INDEX_COLINF	SQL_KEY_COLUMN_USAGE	SQL_REFERENTIAL_CONSTRAINTS	SQL_TABLE_PRIVILEGES	SQL_AUDITS	STATUS_TABLES	STATUS_COLUMNS	STATUS_INDEXES	STATUS_CHUNKS	STATUS_SYNONYM_DICTIONARIES
Connection		CONNECT													SRW											
		DISCONNECT																								
Definition SQL statement	HADB user	CREATE USER statement													EX							SR	SR	SR	SR	
		ALTER USER statement													EX							SR	SR	SR	SR	
		DROP USER statement	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX		SR	SR	SR	SR	
	Schema	CREATE SCHEMA statement							EX						EX							SR	SR	SR	SR	
		DROP SCHEMA statement	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX		EX	EX	EX	EX	EX		SR	SR	SR	SR	
	Base table	CREATE TABLE statement	EX	EX	EX	EX	EX	EX	EX							EX	EX	EX		EX		SR	SR	SR	SR	
		ALTER TABLE statement (adding column)	EX	EX		EX	EX															SR	SR	SR	SR	
		ALTER TABLE statement (changing maximum number of chunks)	EX			EX	EX															SR	SR	SR	SR	
		ALTER TABLE statement (changing regular multi-chunk table to archivable multi-chunk table)	EX	EX	EX	EX	EX	EX	EX	EX	EX					EX	EX	EX		EX		SR	SR	SR	SR	
		ALTER TABLE statement (changing archivable multi-chunk table to regular multi-chunk table)	EX	EX	EX	EX	EX	EX		EX	EX	EX	EX	EX		EX	EX	EX	EX	EX		SR	SR	SR	SR	
		ALTER TABLE statement (changing column name)	EX	EX						EX	EX						EX	EX				SR	SR	SR	SR	
		DROP TABLE statement	EX	EX	EX	EX	EX	EX		EX	EX	EX	EX	EX		EX	EX	EX	EX	EX		SR	SR	SR	SR	
	Viewed table	CREATE VIEW statement (during preprocessing)	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			●					SRW		SR	SR	SR	SR	SR
		CREATE VIEW statement (during execution)	EX	EX					EX	EX	EX	EX	EX	EX						EX		SR	SR	SR	SR	
		ALTER VIEW statement (during preprocessing)	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW	SRW	SRW	●					SRW		SR	SR	SR	SR	SR
		ALTER VIEW statement (during execution)	EX	EX					EX	EX	EX	EX		EX						EX		SR	SR	SR	SR	
		DROP VIEW statement	EX	EX						EX	EX	EX	EX	EX						EX		SR	SR	SR	SR	
	Index	CREATE INDEX statement	EX	EX	EX		EX	EX									EX					SR	SR	SR	SR	
		DROP INDEX statement	EX	EX	EX		EX	EX									EX					SR	SR	SR	SR	
	Audit target definition	CREATE AUDIT statement													EX						EX	SR	SR	SR	SR	
		DROP AUDIT statement													EX						EX	SR	SR	SR	SR	
	Privilege	GRANT statement (granting user privilege, schema operation privilege, or audit privilege)													EX							SR	SR	SR	SR	
		GRANT statement (granting access privilege)	EX												EX					EX		SR	SR	SR	SR	
		REVOKE statement (revoking user privilege, schema operation privilege, or audit privilege)	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX		SR	SR	SR	SR	
		REVOKE statement (revoking access privilege)	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX		SR	SR	SR	SR	
Data manipulation SQL		SELECT statement	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			●					SRW		SR	SR	SR	SR	SR
		INSERT statement	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			●					SRW		SR	SR	SR	SR	SR
		UPDATE statement	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			●					SRW		SR	SR	SR	SR	SR
		DELETE statement	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			●					SRW		SR	SR	SR	SR	SR
		TRUNCATE TABLE statement	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW								SRW		SR	SR	SR	SR	
		PURGE CHUNK statement	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			●					SRW		SR	SR	SR	SR	SR

Legend:

EX

The resource is locked in exclusive mode (EX).

SRW

The resource is locked in shared retrieval mode (SR). Note that if a transaction attempts to reserve a locked resource that has already been reserved by another transaction, the transaction waits until the resource becomes able to be reserved.

SR

The resource is locked in shared retrieval mode (SR).

- The resource is locked in shared retrieval mode (SR) when any of the following objects are specified in the SQL statement. Note that if a transaction attempts to reserve a locked resource that has already been reserved by another transaction, the transaction waits until the resource becomes able to be reserved.

  1. Table function derived table for which the ADB_AUDITREAD function is specified
  2. SQL_AUDITS dictionary table
  3. Viewed table defined by specifying in a CREATE VIEW statement a table function derived table that specifies the ADB_AUDITREAD function
  4. Viewed table defined by specifying the SQL_AUDITS dictionary table in a CREATE VIEW statement
  5. Viewed table that depends on the preceding viewed tables in the list (items 3 or 4)

Table 2-18: Dictionary tables and system tables locked when a database is referenced or updated (2)

Database reference and update processing (SQL statements and commands to be executed)		Dictionary tables (base tables) and system tables (base tables) to be locked																						
		SQL_TABLES	SQL_COLUMNS	SQL_INDEXES	SQL_DIV_TABLE	SQL_DIV_INDEX	SQL_DBAREAS	SQL_SCHEMATA	SQL_VIEWS	SQL_VIEW_TABLE_USAGE	SQL_VIEW_OBJECT	SQL_DEFINE_SOURCE	SQL_DEFINE_ENVIRONMENT	SQL_USERS	SQL_TABLE_CONSTRAINTS	SQL_INDEX_COLINF	SQL_KEY_COLUMN_USAGE	SQL_REFERENTIAL_CONSTRAINTS	SQL_TABLE_PRIVILEGES	SQL_AUDITS	STATUS_TABLES	STATUS_COLUMNS	STATUS_INDEXES	STATUS_CHUNKS	STATUS_SYNONYM_DICTIONARIES
Commands	adbimport (if the background-import facility is not used)	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbimport (if the background-import facility is used)	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbexport	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	SR
	adbgetcst	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR		
	adbidxrebuild	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbmergechunk	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbchgchunkcomment	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbchgchunkstatus	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbdbstatus	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbmodarea																								
	adbarchivechunk	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbunarchivechunk	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW			SRW					SRW		SR	SR	SR	SR	
	adbreorgsystemdata	SRW	SRW	SRW	SRW	SRW	SRW		SRW		SRW								SRW		SR/SUW	SR/SUW	SR/SUW	-/SUW	-/SUW
	adbclientdefmang													SRW											
	adbsyndict																								SR
	adbaudittrail (if the --start option is specified)													SRW						SRW					
	adbaudittrail (if an option other than --start is specified)													SRW											
	adbconvertaudittrailfile													SRW											

Legend:

SRW

The resource is locked in shared retrieval mode (SR). Note that if a transaction attempts to reserve a locked resource that has already been reserved by another transaction, the transaction waits until the resource becomes able to be reserved.

SR

The resource is locked in shared retrieval mode (SR).

SR/SUW

If the target resource is a table that is not to be manipulated by using an SQL statement or command, the resource is locked in shared retrieval mode (SR).

If the target resource is a table that is to be manipulated by using an SQL statement or command, the resource is locked in shared update mode (SU). Note that if a transaction attempts to reserve a locked resource that has already been reserved by another transaction, the transaction waits until the resource becomes able to be reserved.

-/SUW

If the target resource is a table that is not to be manipulated by using an SQL statement or command, the resource is not locked.

If the target resource is a table that is to be manipulated by using an SQL statement or command, the resource is locked in shared update mode (SU). Note that if a transaction attempts to reserve a resource that has already been reserved by another transaction, the transaction waits until the resource becomes able to be reserved.

2.10.7 Examples of locks

This section shows examples of locks performed by HADB.

To understand the mechanism of locks, read also 2.10.2 Locked resources and 2.10.4 Locked resources that are reserved and their lock modes.

(1) Locks performed when the CREATE TABLE statement is executed (to define a base table)

When the `CREATE TABLE` statement is executed, the following resources are locked:

- Dictionary (exclusive mode (EX))
- Pre-processing table (exclusive mode (EX))
- DB area
  - Data DB area (exclusive mode (EX))
  - Dictionary DB area (shared retrieval mode (SR))
  - System-table DB area (shared retrieval mode (SR))
- Table
  - Processing-target table (exclusive mode (EX))
  - Dictionary table (exclusive mode (EX))
  - System table (cost information) (shared retrieval mode (SR))
  - System table (chunk information) (shared retrieval mode (SR))

Dictionaries are locked in exclusive mode (EX). Therefore, a definition SQL statement that is executed subsequently to a `CREATE TABLE` statement is placed in a wait state until execution of the `CREATE TABLE` statement finishes. Also, dictionary tables are locked in exclusive mode (EX). Therefore, a retrieval or update SQL statement that is executed subsequently to a `CREATE TABLE` statement will be placed in a wait state until execution of the `CREATE TABLE` statement finishes.

The following shows examples of cases where a retrieval SQL statement (`SELECT` statement) that is executed subsequently to a `CREATE TABLE` statement is placed in a wait state:

Example 1: When the SELECT statement is executed for a base table that is being defined

```
CREATE TABLE T1("ID" INT, "NAME" VARCHAR(100)) in ADBUTBL01   ... 1
SELECT * FROM T1                                              ... 2
```

The `SELECT` statement on line 2 is placed in a wait state until execution of the `CREATE TABLE` statement on line 1 finishes.

Example 2: When the SELECT statement is executed for a base table that is not a base table being defined

```
CREATE TABLE T1("ID" INT, "NAME" VARCHAR(100)) in ADBUTBL01   ... 1
SELECT * FROM T2                                              ... 2
```

Note: In this example, base table T2 is defined in a data DB area in which base table T1 is not defined.

The SELECT statement on line 2 is placed in a wait state until execution of the CREATE TABLE statement on line 1 finishes.

In addition to retrieval SQL statements, there are some operations which, if executed subsequently to a CREATE TABLE statement, are placed in a wait state until execution of the CREATE TABLE statement finishes. The following shows examples of operations that are placed in a wait state (other than retrieval SQL statements) when they are executed subsequently:

- INSERT statement
- UPDATE statement
- DELETE statement
- TRUNCATE TABLE statement
- PURGE CHUNK statement
- adbimport command
- adbdbstatus command
- adbexport command
- adbmergechunk command

(2) Locks performed when the adbimport command is executed (to import data)

The processing of locks performed when the adbimport command is executed differs depending on whether the background-import facility to be used.

(a) Locks performed when the background-import facility is used

If the background-import facility is used and the adbimport command is executed, the data DB area is locked in protected update mode (PU). Therefore, the update operations that are subsequently executed for the import-target table will result in an error when attempting to reserve already locked resources.

The following shows examples of operations# that result in an error when they are executed subsequently to the adbimport command if the background-import facility is used:

- Definition SQL statements
- INSERT statement
- UPDATE statement
- DELETE statement
- TRUNCATE TABLE statement
- PURGE CHUNK statement
- adbdbstatus command

  Note that the adbdbstatus command can be executed in the following cases:

- When summary information for DB areas is output
- When information is output by specifying the `--shared-lock` option
- When information about the need for reorganization is output

- `adbmergechunk` command

#

Operations that meet any of the following conditions:

- The operation-target table of the SQL statement or command is the same as the import-target table.
- The operation-target table of the SQL statement or command is stored in the same data DB area as the import-target table.
- The operation-target table of the SQL statement or command is stored in the same data DB area as the index defined for the import-target table.
- The index defined for the operation-target table of the SQL statement or command is stored in the same data DB area as the import-target table.

(b) Locks performed when the background-import facility is not used

When the `adbimport` command is executed without using the background-import facility, the data DB area and processing-target table are locked in exclusive mode (EX). Therefore, operations for the import-target table will result in an error when attempting to reserve already locked resources or will be placed in a wait state until the necessary resource is unlocked.

■ Operations that will result in an error when executed subsequently

The following shows examples of operations[#] that result in an error when they are executed subsequently to the `adbimport` command if the background-import facility is not used:

- Definition SQL statements
- `INSERT` statement
- `UPDATE` statement
- `DELETE` statement
- `TRUNCATE TABLE` statement
- `PURGE CHUNK` statement
- `adbdbstatus` command

  Note that the `adbdbstatus` command can start without waiting in the following cases:

  - When summary information for DB areas is output
  - When information is output by specifying the `--shared-lock` option
  - When information about the need for reorganization is output
- `adbmergechunk` command

#

Operations that meet any of the following conditions:

- The operation-target table of the SQL statement or command is the same as the import-target table.
- The operation-target table of the SQL statement or command is stored in the same data DB area as the import-target table.

- The operation-target table of the SQL statement or command is stored in the same data DB area as the index defined for the import-target table.
- The index defined for the operation-target table of the SQL statement or command is stored in the same data DB area as the import-target table.

■ Operations that will be placed in a wait state when executed subsequently

The following shows examples of operations that will be placed in a wait state when they are executed subsequently to the `adbimport` command if the background-import facility is not used:

- Retrieval SQL statements
- `adbexport` command

> **Note**
>
> The preceding operations can start without waiting when they are executed for a table other than the import-target table (these operations are not affected by a lock on the data DB area).

(3) Locks performed when the adbexport command is executed (to export data)

When the `adbexport` command is executed, the following resources are locked in shared retrieval mode (SR):

- Single connection
- Pre-processing table
- Table
  - Processing-target table
  - Dictionary table
  - System table (cost information)
  - System table (chunk information)
  - System table (synonym dictionary information)

Therefore, you can execute any operations (other than those that lock the preceding resources) in exclusive mode (EX) simultaneously with the `adbexport` command. Operations that lock the preceding resources in exclusive mode (EX) will result in an error when attempting to lock a resource or will be placed in a wait state until the necessary resource is unlocked.

■ Operations that will result in an error when they are executed subsequently for the export-target table

The following shows examples of operations that will result in an error when they are executed subsequently to the `adbexport` command for the export-target table:

- Definition SQL statements
- `TRUNCATE TABLE` statement
- `PURGE CHUNK` statement

Note

The preceding operations can start without resulting in an error when they are executed for a table other than the export-target table (these operations are not affected by a lock on the data DB area).

■ Operations that will be placed in a wait state when they are executed subsequently for the export-target table

The following shows examples of operations that are placed in a wait state when they are executed subsequently to the `adbexport` command for the export-target table:

- `adbimport` command for which the background-import facility is disabled
- `adbmergechunk` command (in which the `--purge-chunk` option is omitted or in which `WAIT` is specified for the `--purge-chunk` option)
- `adbreorgsystemdata` command (if the export target is a system table)

Note

The preceding operations can start without waiting when they are executed for a table other than the export-target table (these operations are not affected by a lock on the data DB area).

(4) Locks performed when the adbmergechunk command is executed (to merge chunks)

When the `adbmergechunk` command is executed, the following resources are locked:

- Single connection (shared retrieval mode (SR))
- Pre-processing table (shared retrieval mode (SR))
- DB area
  - Data DB area (exclusive mode (EX))
  - System-table DB area (shared retrieval mode (SR))
- Table
  - Processing-target table (shared update mode (SU))
  - Dictionary table (shared retrieval mode (SR))
  - System table (cost information) (shared retrieval mode (SR))
  - System table (chunk information) (shared retrieval mode (SR))

Therefore, the following operations will result in an error when they are executed subsequently and attempt to reserve already locked resources:

- Subsequently executing an operation that locks a table stored in the same DB area as the merge chunk target table
- Subsequently executing an operation that locks the merge chunk target table in protected update mode (PU) or exclusive mode (EX)
- Subsequently executing an operation that locks a single connection, pre-processing table, dictionary table, or system table in exclusive mode (EX)

The following shows examples of operations[#] that result in an error when they are executed subsequently to the adbmergechunk command:

- Definition SQL statements
- INSERT statement
- UPDATE statement
- DELETE statement
- TRUNCATE TABLE statement
- PURGE CHUNK statement
- adbimport command
- adbdbstatus command

  Note that the adbdbstatus command can start without waiting in the following cases:

  - When summary information for DB areas is output
  - When information is output by specifying the --shared-lock option
  - When information about the need for reorganization is output

#

Operations that meet any of the following conditions:

- The operation-target table of the SQL statement or command is the same as the merge chunk target table.
- The operation-target table of the SQL statement or command is stored in the same data DB area as the merge chunk target table.
- The operation-target table of the SQL statement or command is stored in the same data DB area as the index defined for the merge chunk target table.
- The index defined for the operation-target table of the SQL statement or command is stored in the same data DB area as the merge chunk target table.

(5) Locks performed when the adbdbstatus command is executed (to analyze the database status)

The processing of locks performed when the adbdbstatus command is executed differs depending on the information that will be output.

(a) Locks performed when summary information about a DB area is output

When the adbdbstatus command is used to output summary information about a DB area, a single connection and dictionary table are locked in shared retrieval mode (SR). Therefore, operations that lock the dictionary table in exclusive mode (EX) will result in an error when they are executed subsequently and attempt to reserve already locked resources.

The following shows examples of operations that will result in an error when they are executed subsequently to the adbdbstatus command:

- Definition SQL statements

(b) Locks performed when information is output by specifying the --shared-lock option and when information about the need for reorganization is output

When the `adbdbstatus` command with the `--shared-lock` option specified outputs information or when the `adbdbstatus` command outputs information about the need for reorganization, the following resources are locked in shared retrieval mode (SR):

- Single connection
- Pre-processing table
- Table
  - Processing-target table
  - Dictionary table

Therefore, operations that lock the preceding resources in exclusive mode (EX) will result in an error when they are executed subsequently and attempt to reserve already locked resources.

The following shows examples of operations that will result in an error when they are executed subsequently to the `adbdbstatus` command:

- Definition SQL statements
- `TRUNCATE TABLE` statement
- `PURGE CHUNK` statement
- `adbimport` command for which the background-import facility is disabled

(c) Locks performed when other types of information is output

In cases where neither (a) Locks performed when summary information about a DB area is output nor (b) Locks performed when information is output by specifying the --shared-lock option and when information about the need for reorganization is output applies, the following resources are locked:

- Single connection (shared retrieval mode (SR))
- Pre-processing table (shared retrieval mode (SR))
- DB area
  - Data DB area (exclusive mode (EX))
  - Dictionary DB area (exclusive mode (EX))
  - System-table DB area (exclusive mode (EX))
- Table
  - Dictionary table (shared retrieval mode (SR))

Therefore, operations that lock the data DB area, dictionary DB area, or system-table DB area will result in an error when they are executed subsequently and attempt to reserve already locked resources. In addition, operations that lock a single connection, pre-processing table, or dictionary table in exclusive mode (EX) will also result in an error when they are executed subsequently.

The following shows examples of operations that will result in an error when they are executed subsequently to the `adbdbstatus` command:

- Definition SQL statements

- INSERT statement
- UPDATE statement
- DELETE statement
- TRUNCATE TABLE statement
- PURGE CHUNK statement
- adbimport command
- adbmergechunk command

(6) Locks performed when the adbmodarea command is executed (to add or change a DB area)

When the adbmodarea command is executed, a single connection is locked in exclusive mode (EX). Therefore, all operations that lock a single connection (operations for connecting to the HADB server) will result in an error during execution of the adbmodarea command.

The following shows examples of operations that will result in an error when they are executed subsequently to the adbmodarea command:

- Definition SQL statements#
- Retrieval SQL statements#
- INSERT statement#
- UPDATE statement#
- DELETE statement#
- TRUNCATE TABLE statement#
- PURGE CHUNK statement#
- adbimport command
- adbdbstatus command
- adbexport command
- adbmergechunk command

#

The SQL statement cannot be executed because an error occurs when the application program connects to the HADB server.

(7) Locks performed when the adbls command is executed (to display the status of the HADB server)

The adbls command does not lock any resources. Therefore, this command can be executed simultaneously with operations that lock resources.

2.11 Database recovery

This section explains how a database can be recovered if an error causes the HADB server to terminate abnormally.

2.11.1 Recovery flow based on a restart

If the HADB server terminated abnormally, restarting HADB causes the HADB server to automatically restart using the status information. Once restarted, HADB uses the system logs to recover the database and restores it to the state it was in before the HADB server terminated abnormally.

(1) Status information

To guard against abnormal termination, the HADB server outputs the information necessary for automatic restarting. This information is called the *status information*.

Status information is output whenever the status of the HADB server changes, and it is stored in the status file. The following types of information are output as status information:

- HADB server termination mode (normal termination, abnormal termination, and so on)
- HADB server operational status (start processing, running, termination processing, and so on)

The status file is stored under the DB directory (`$DBDIR/ADBSYS/ADBSTS`).

(2) System log

When a database is updated, its update history information is output. This information is called the *system log*. Files in which the system logs are stored is called *system log files*.

System logs are used for restarting HADB after an abnormal termination, and for the recovery processing after a rollback has occurred.

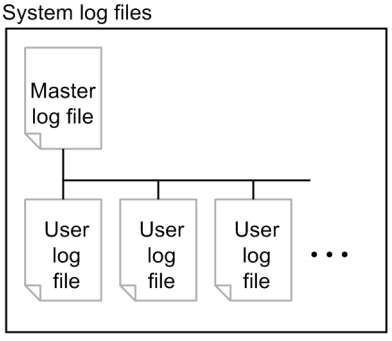
(a) System log file creation location

The system log files are stored under the DB directory (`$DBDIR/ADBSYS/ADBSLG`).

(b) System log file configuration

System log files actually consist of two different types of files, a master log file and user log files. The following figure shows the configuration of the system log files.

Figure 2-35: System log file configuration



- Master log file

  The master log file stores the system control-related history information (event log) such as the start and end of transactions. It also manages user log file groups.

  When the HADB server starts, a single master log file is automatically created. The file name is `MSTLOG`.

- User log file

  A user log file stores database-related operation history (*user logs*).

  A separate user log file is allocated for each transaction that updates the database and for each real thread.

  When the HADB server starts, a user log file is created based on settings specified by the user. The file name begins with `USRLOG`.

  If the specifications related to user log files are changed, the system log files are re-created when the HADB server is started.

2.12 Client-group facility

This section explains the client-group facility.

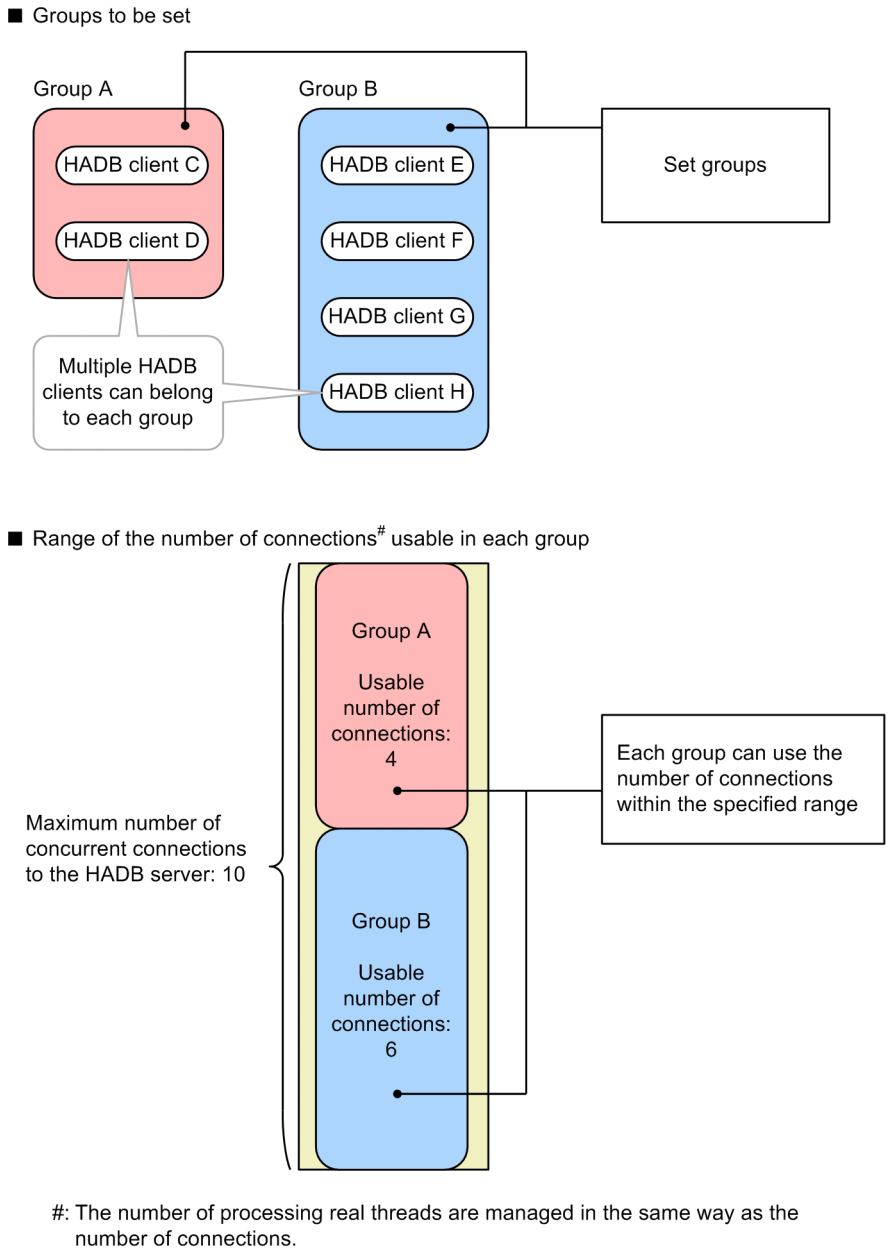
2.12.1 Overview of the client-group facility

HADB supports the *client-group facility* that groups multiple HADB clients together and manages them as a group.

The client-group facility enables you to set the numbers of connections and processing real threads that are available to the group. The HADB clients belonging to a group will perform processing within the numbers of connections and processing real threads set for that group.

The following provides an overview of the client-group facility using management of the number of connections as an example.

Figure 2-36: Example application of the client-group facility (managing the number of connections)



When you use the client-group facility and you set appropriate values for the numbers of connections and processing real threads required for each of the various groups, you can achieve the following:

- Prevent specific HADB clients and commands from monopolizing the available processing real threads and connections to the HADB server (this prevents adverse effects on the processing of other HADB clients and commands).
- Reserve the number of connections and processing real threads required for the periodic processing that is performed by specific HADB clients and commands (this prevents adverse effects from the processing being performed by other HADB clients and commands).

2.12.2 Types of groups that can be set by the client-group facility

The client-group facility supports the following types of groups:

- **Client group**

  HADB clients can belong to a client group. Multiple HADB clients can belong to the same client group, and you can set up multiple client groups.

- **Command group**

  Commands that connect to the HADB server can belong to the command group. For example, commands for importing data and rebuilding indexes can belong to this group. For details about the commands that connect to the HADB server, see *List of commands* in the manual *HADB Command Reference*. You can set up only one command group.

  > **Note**
  >
  > HADB clients can also belong to the command group. For details, see adbcltgrp (for setting a command group) in 7.2.12 Operands and options related to the client-group facility (command format).

2.12.3 Setting the numbers of connections and processing real threads for each group

The client-group facility enables you to set for each group a maximum number of connections and a maximum number of processing real threads that are to be made available to HADB clients and commands. You can also set minimum numbers for each group.

(1) Setting the number of connections for each group

The client-group facility supports the following connection limitations:

- **Maximum number of concurrent connections**

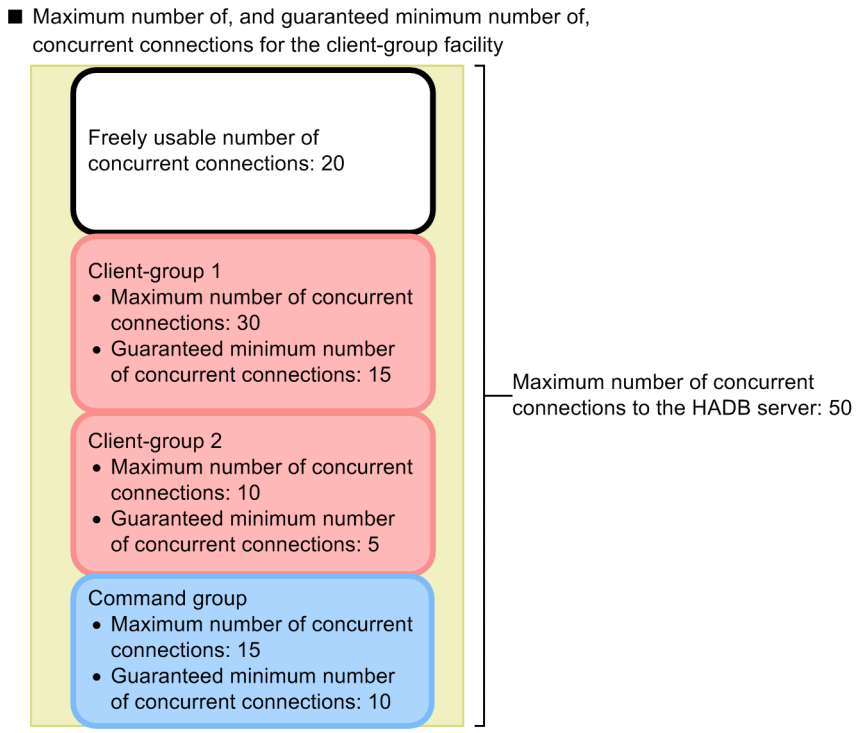
  You can set a maximum number of concurrent connections that are to be made available to a target group. The HADB clients and commands belonging to the group can establish concurrent connections only within the set maximum number of concurrent connections.

- **Guaranteed minimum number of concurrent connections**

  You can set a minimum number of connections that will always be available to a target group. This minimum number of concurrent connections is guaranteed to this group at any time. This number of connections is not affected by the connections used by HADB clients and commands that belong to other groups.

The following figure provides an overview of the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections supported by the client-group facility.

Figure 2-37: Overview of the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections supported by the client-group facility



Explanation

The target group can always use up to the set guaranteed minimum number of concurrent connections. For example, 15 concurrent connections (guaranteed minimum number of concurrent connections) will always be available to client group 1.

The maximum number of concurrent connections to the HADB server (50) minus the sum of the guaranteed minimum numbers of concurrent connections set for all groups (30) is the freely usable number of connections (20).

All HADB clients and commands can establish connections within the scope of the freely usable number of connections. Whether they belong to a group is irrelevant.

If an HADB client or command belongs to a group and there are freely usable connections (20), the maximum number of concurrent connections set for that group is available to that HADB client or command. For example, 15 connections are available for client group 1 in addition to the guaranteed minimum number of concurrent connections (15), because this group's maximum number of concurrent connections is 30.

If the freely usable number of connections (20) are all unused, a maximum of 20 connections are available to the HADB clients and commands that belong to no group.

Note

You can enable a setting at the group level that outputs a warning message `KFAA40020-W` when the number of connections to the HADB server approaches the maximum number of concurrent connections.

(2) Setting the number of processing real threads for each group

The client-group facility supports the following types of processing real thread counts.

- Maximum number of processing real threads that can be used

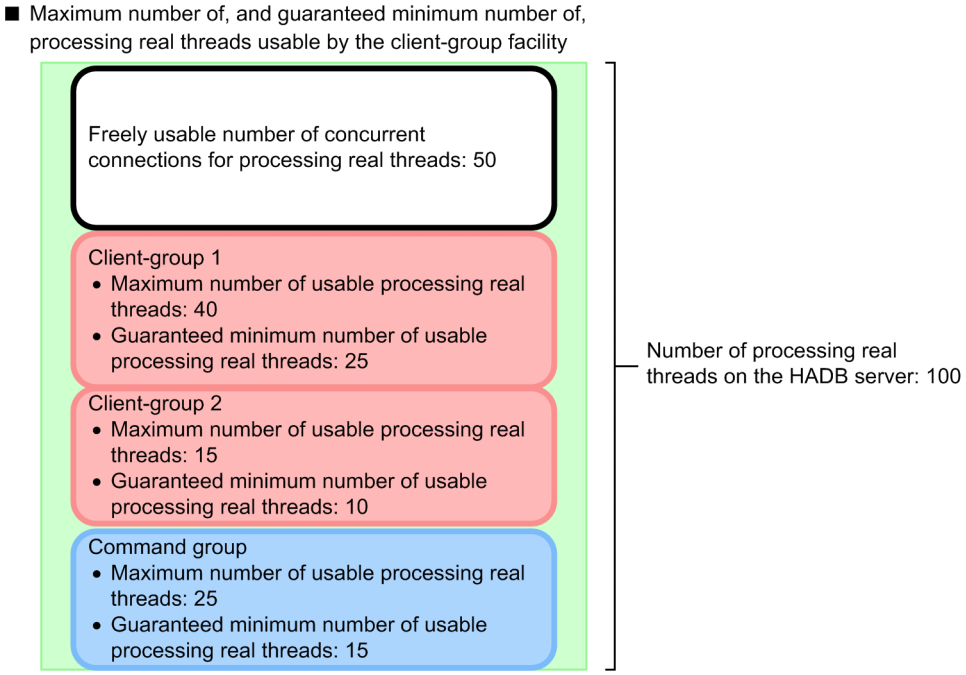
You can set a maximum number of processing real threads that can be used by a target group. The HADB clients and commands belonging to the target group will perform processing within the set maximum number of processing real threads.

- Guaranteed minimum number of processing real threads that can be used

  You can set a guaranteed minimum number of processing real threads that can be used by a target group. This guaranteed minimum number of processing real threads will always be available to the target group. This number is not affected by the connections used by HADB clients and commands that do not belong to the local group.

The following figure provides an overview of the maximum number of processing real threads and the guaranteed minimum number of processing real threads supported by the client-group facility.

Figure 2-38: Overview of the maximum number of processing real threads and the guaranteed minimum number of processing real threads supported by the client-group facility



Explanation

The guaranteed minimum number of processing real threads will always be available to the target group. For example, up to the guaranteed minimum number of processing real threads (25) are always available to client group 1.

The number of processing real threads on the HADB server (100) minus the sum of the guaranteed minimum numbers of processing real threads set for all groups (50) is the freely usable number of processing real threads (50).

All HADB clients and commands can use processing real threads within the freely usable number of processing real threads, regardless of whether HADB clients and commands belong to a group.

If an HADB client or command belongs to a group and there are freely usable processing real threads (50), the maximum number of processing real threads set for that group are available to that HADB client or command. For example, for client group 1, 15 processing real threads are available in addition to the guaranteed minimum number of processing real threads (25) because this group's maximum number of processing real threads is 40.

If the freely usable number of processing real threads (50) are all unused, a maximum of 50 processing real threads are available to the HADB clients and commands that belong to no group.

2.13 Centralized management of client definitions

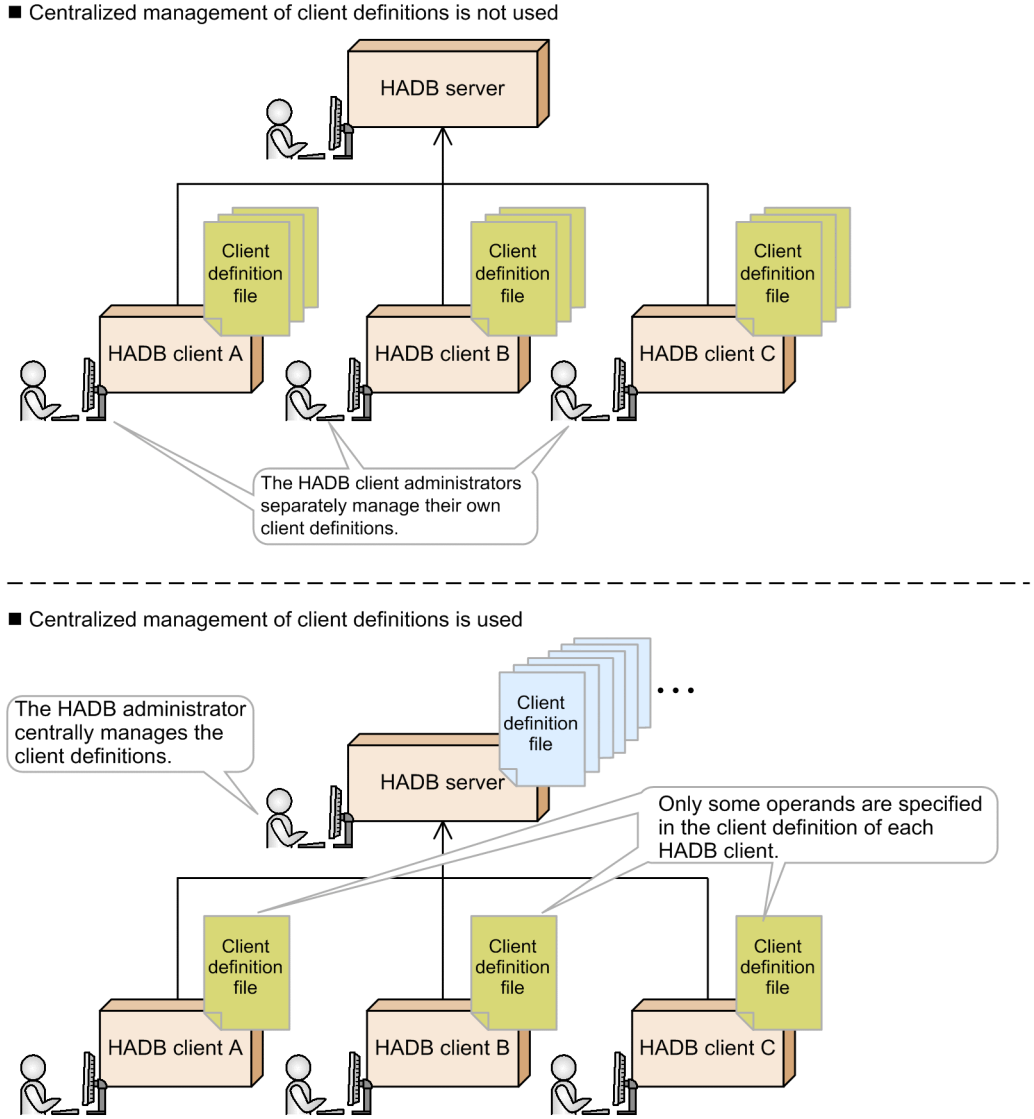
This section describes centralized management of client definitions.

2.13.1 What is centralized management of client definitions?

Each HADB client manages its own client definition separately. However, the client definitions of all HADB clients can be centrally managed on the HADB server. This function is called centralized management of client definitions. If this function is used, because the HADB administrator can centrally manage the client definitions with the HADB server, the administrator of each HADB client does not need to individually manage the client definition.

The following figure compares the cases in which centralized management of client definitions is used and not used.

Figure 2-39: Comparison of cases in which centralized management of client definitions is used and not used



Explanation

If centralized management of client definitions is used, the HADB administrator can add or modify the client definitions of all HADB clients on the HADB server.

Exceptions are the following operands, which must be specified in the client definition on each HADB client:

- adb_clt_rpc_srv_host
- adb_clt_rpc_srv_port
- adb_clt_rpc_con_wait_time
- adb_clt_rpc_sql_wait_time

> **Note**
>
> The preceding operands specify the information required for connecting to the HADB server. Therefore, they must be specified in the client definition on each HADB client.

▪ **Advantages of using centralized management of client definitions**

- The HADB administrator can manage the values of the operands in each client definition. For example, the HADB administrator can change the values of performance-related operands (such as adb_sql_exe_max_rthd_num, which specifies the maximum number of real threads that process SQL statements) based on the results of tuning.
- If the client-group facility is used for the first time, the adb_clt_group_name operand, which specifies a client group name, must be added to the client definitions of the relevant HADB clients. In this case, the administrator of the HADB client must add the adb_clt_group_name operand. However, if centralized management of client definitions is used, one administrator only has to add the adb_clt_group_name operand to the client definitions on the HADB server. Therefore, the administrator of each HADB client does not need to add the adb_clt_group_name operand to the client definition on each HADB client.

As described before, centralized management of client definitions allows you to change the client definitions more easily and quickly.

2.13.2 Files to be created before using centralized management of client definitions

Before you can use centralized management of client definitions, you must create the following files on the HADB server:

- Client management definition file
- Client definition files

(1) Information to be specified in the client management definition file

In the client management definition file, specify the authorization identifiers of HADB users who connect to the HADB server. In addition, specify the client definition files to be applied to the application programs (including the adbsql command) that connect to the server with those authorization identifiers. The following shows a specification example.

Specification example

```
adbclientmang -f client01.def -i ADBUSER01,ADBUSER02             ...1
adbclientmang -f client02.def -i ADBUSER03                       ...2
adbclientmang -f client03.def -i ADBUSER04,ADBUSER05,ADBUSER06   ...3
      :               :              :
```

Explanation

1. The client definition in the client definition file `client01.def` is applied to the application program that connects by using the authorization identifier `ADBUSER01` or `ADBUSER02`.
2. The client definition in the client definition file `client02.def` is applied to the application program that connects by using the authorization identifier `ADBUSER03`.
3. The client definition in the client definition file `client03.def` is applied to the application program that connects by using the authorization identifier `ADBUSER04`, `ADBUSER05`, or `ADBUSER06`.

Store the client management definition file in the `$ADBDIR/conf` directory on the HADB server.

(2) Information to be specified in a client definition file

In a client definition file, specify the definition of a client. Store client definition files in the `$ADBDIR/conf` directory on the HADB server.

Note that the following operands specified in a client definition file on the HADB server are ignored, and therefore you do not need to specify the operands in a client definition file. Specify the following operands in the client definitions on HADB clients.

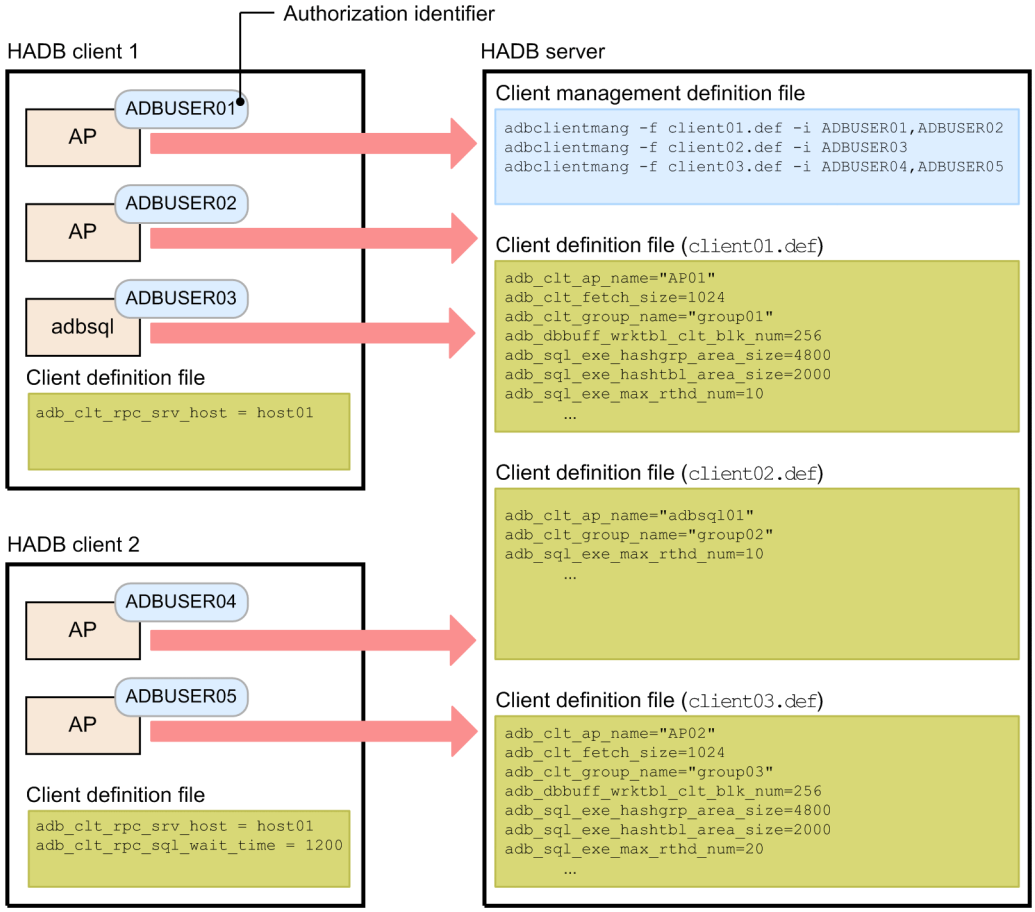
- `adb_clt_rpc_srv_host`
- `adb_clt_rpc_srv_port`
- `adb_clt_rpc_con_wait_time`
- `adb_clt_rpc_sql_wait_time`

If you do not specify operands other than the preceding ones, the defaults are used.

2.13.3 Application example of the processing by centralized management of client definitions

The following figure shows an application example of the processing by centralized management of client definitions.

Figure 2-40: Application example of the processing by centralized management of client definitions



Explanation

For an application program or the `adbsql` command that connects to the HADB server, the client definition to be applied is determined by the specified authorization identifier. In the preceding example, if the authorization identifier is `ADBUSER01`, the client definition in the `client01.def` file is applied.

▪ Notes

- If an application program or the `adbsql` command whose authorization identifier is not specified in the client management definition file (for example, in the preceding figure, `ADBUSER06`) attempts to connect to the HADB server, the HADB client's own definition is applied.
- Centralized management of client definitions works only when either of the following items is executed (the function does not work when a command other than `adbsql` is executed):
  - Application program
  - `adbsql` command

2.14 Background-import facility

This section describes the background-import facility.

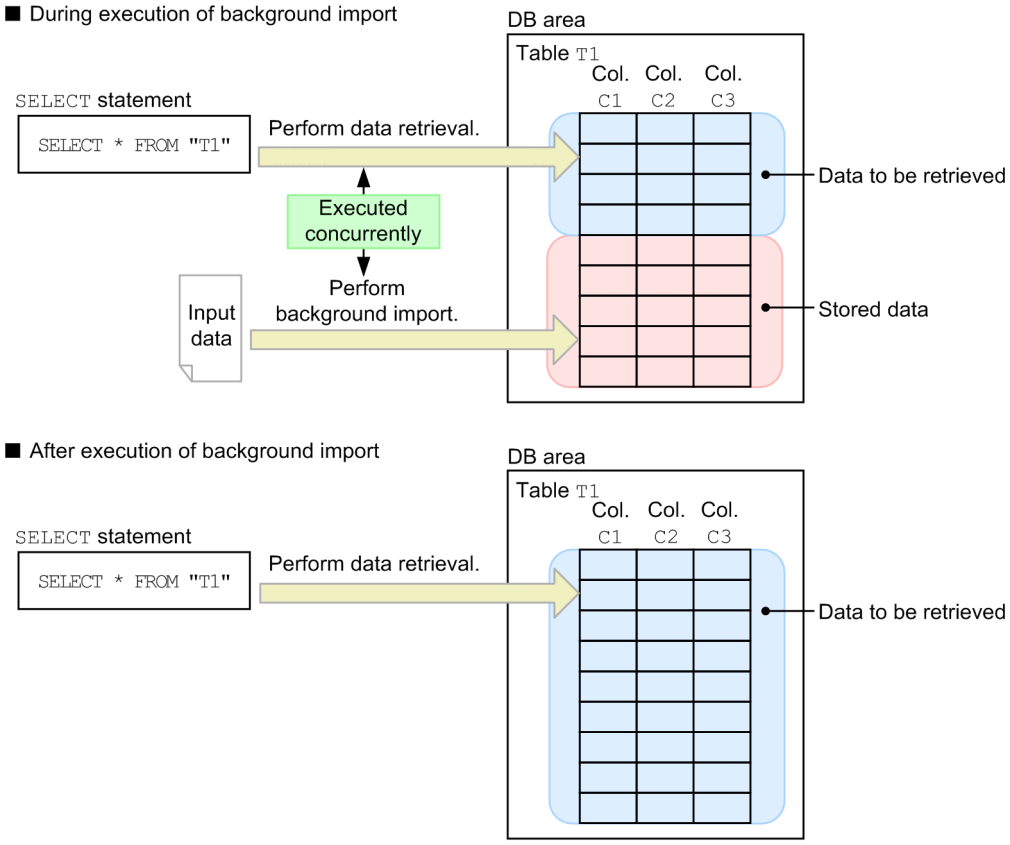
2.14.1 About background import

HADB provides the *background-import facility* to enable concurrent execution of data retrieval and data importing.

Applying the background-import facility for importing data enables you to perform data-retrieval and data-storage processes concurrently for the same table.

The following figure provides an overview of background import.

Figure 2-41: Overview of background import



Explanation

Retrieving data stored in table `T1` and storing data in table `T1` by means of background import can be performed concurrently.

Note that the data being imported using background import becomes retrievable after the data-storage process is completed.

> **Note**
>
> If you want to retrieve data and store a large amount of data concurrently, we recommend that you use the background-import facility. If you want to retrieve data and store a small amount of data concurrently,

consider using an SQL statement (`INSERT` statement) instead of the background-import facility so as not to reduce the data storage efficiency. For details, see 5.2.4 Points to consider in defining a multi-chunk table.

2.14.2 Managing data in data-import units (chunks)

To implement background import, HADB manages as a single unit the data that is stored in a base table by each background-import operation. This unit is called a *chunk*.

(1) Chunk overview

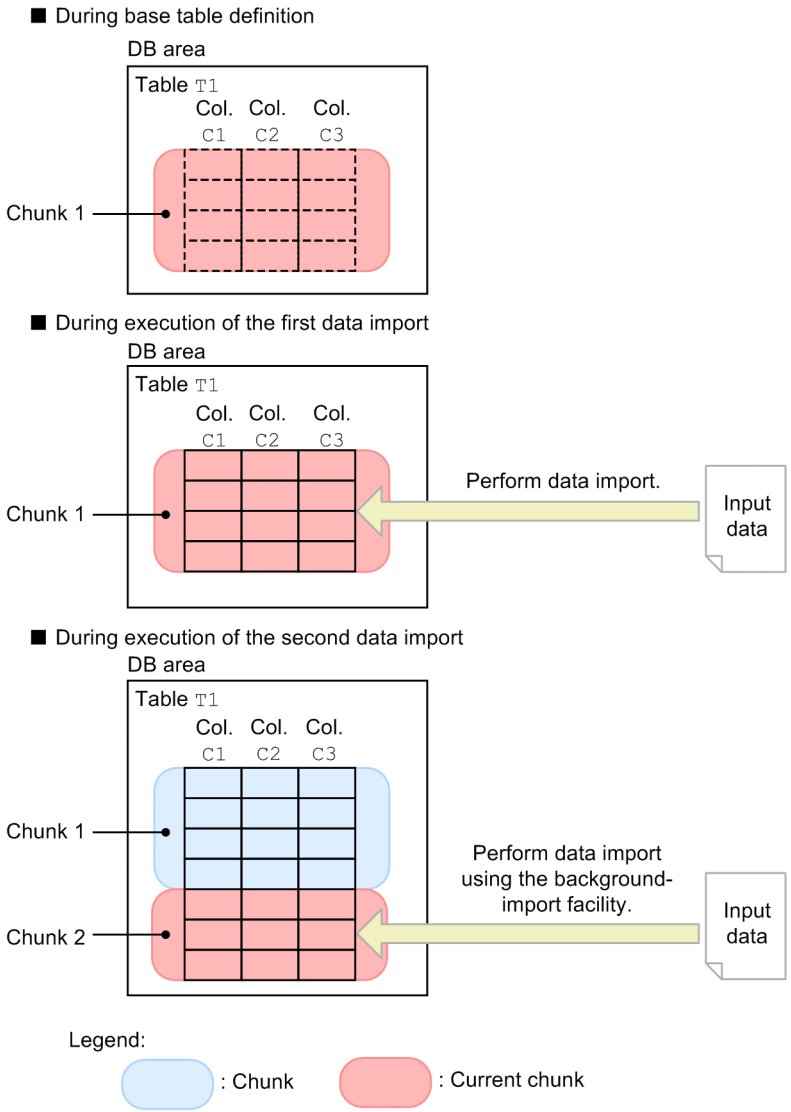
HADB can manage multiple chunks. It assigns a unique number to each chunk within the base table. This number is called a *chunk ID*.

Each time background import is executed, a new chunk is created. Among the created chunks, the one to which data is to be added is called the *current chunk*. There is only one current chunk in a base table.

When a new chunk is created by an action such as execution of background import, the current chunk changes (that is, the chunk to which data is to be added changes). This process of changing the current chunk is called *current chunk swapping*.

The following figure shows the relationship between background import and the current chunk.

Figure 2-42: Relationship between background import and the current chunk



Explanation

When table `T1` is defined as a new base table, a blank chunk (chunk 1) is created as the current chunk.

Executing the first data-import operation on table `T1` stores data. The cluster of data that is stored is managed as chunk 1.

Executing the second data-import operation on table `T1` as background import creates a new chunk (chunk 2), and data is stored. The cluster of data that is stored is managed as chunk 2, and the current chunk changes from chunk 1 to chunk 2.

Note

■ Timing at which current chunk swapping occurs

The current chunk changes in the following cases:

- When background import is executed
- When multiple chunks, including the current chunk, are merged

Additionally, the current chunk might be changed in the following case:

- When the chunk status is changed

■ Relationship between chunks and row insertion or row updating

When the `INSERT` statement is used to insert a row into a base table in which data has been stored by means of background import, the inserted row is stored in the current chunk. On the other hand, when the `UPDATE` statement is used to update a row, the updated row is stored in the chunk in which the update-target row is located.

■ Chunk creation and chunk IDs when indexes are defined

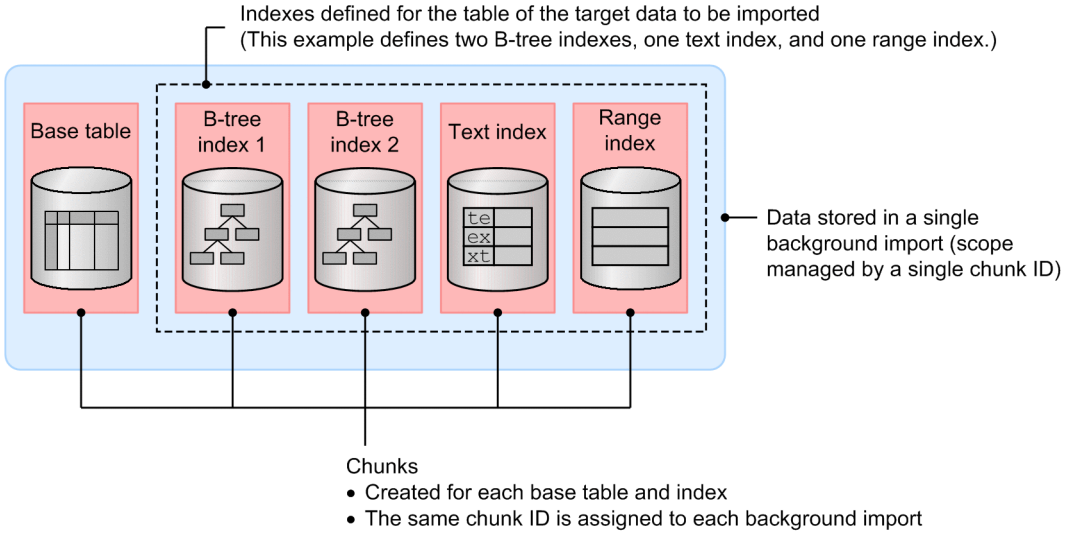
When indexes are defined for the base table into which data is to be imported, the data in the index created during background import is also managed in units of chunks.

For example, when indexes (B-tree, text, or range indexes) are defined for the base table into which data is to be imported, chunks are created separately for the data in the base table and the data of the indexes (B-tree, text, range indexes) created during background import.

HADB manages chunks in the background-import units. Therefore, the same chunk ID is assigned to the above-mentioned chunks.

The following figure provides an overview of chunks and chunk IDs when indexes are defined.

Figure 2-43: Overview of chunk creation and chunk IDs when indexes are defined



■ Relationship between the number of created chunks and the number of indexes

When indexes are defined for base tables into which data is to be imported, the number of chunks created in a single background-import operation equals the sum total of the base tables into which data is to be imported and the number of indexes defined for those base tables.

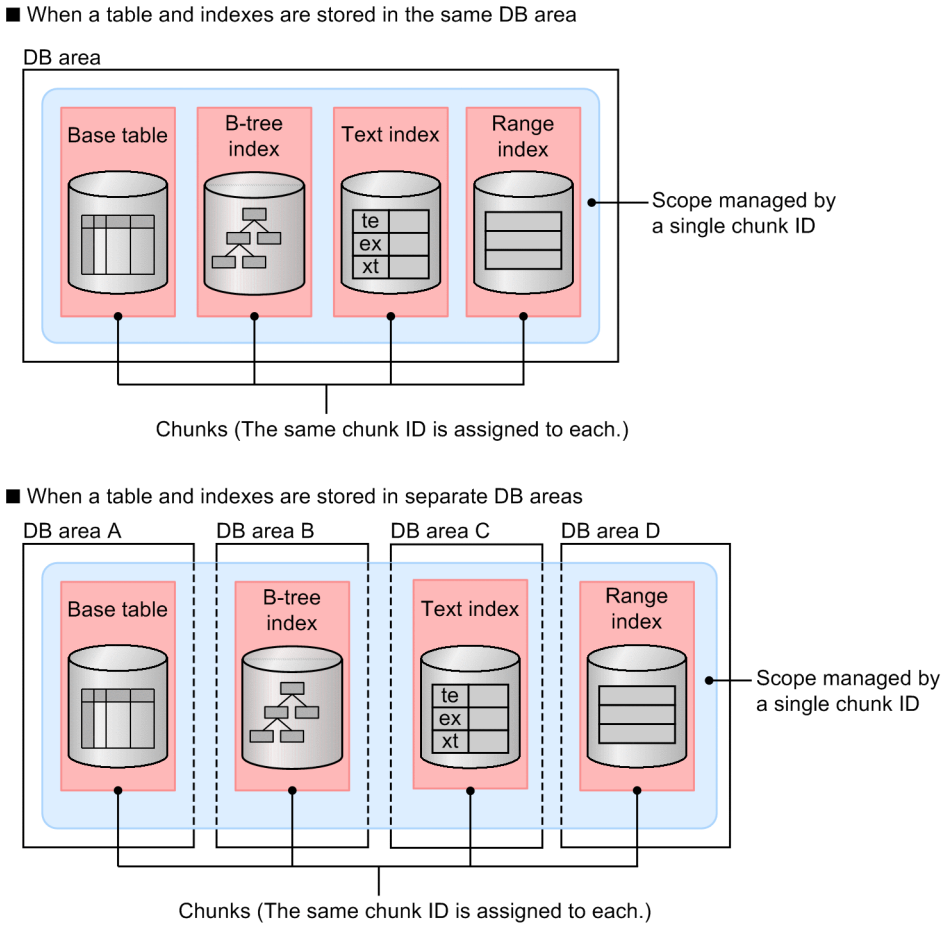
For example, if two B-tree indexes, one text index, and one range index are defined for a base table, the number of chunks created in a single background-import operation would be five.

In HADB, there is a limit to the number of chunks that can be managed in a single DB area and a single base table. For details about the maximum number of chunks in a single DB area and the maximum number of chunks that can be created in a single base table, see D.1 Maximum and minimum values related to system configuration.

■ Relationship between chunks and DB areas

Background import can be executed regardless of whether tables and indexes are stored in the same DB area or in separate DB areas. In either case, a separate chunk is created for each base table and index, and a single chunk ID is assigned. The following figure shows the relationship between chunks and DB areas.

Figure 2-44: Relationship between chunks and DB areas



(2) Merging chunks

Multiple chunks created by executing background import can be merged into a single new chunk. Merging chunks can reduce the total number of chunks being used.

Chunks can be merged even if there is a user who is executing retrieval on a chunk designated for merging.

Important

■ Relationship between the merging of chunks and the performance of retrieval using B-tree indexes and text indexes

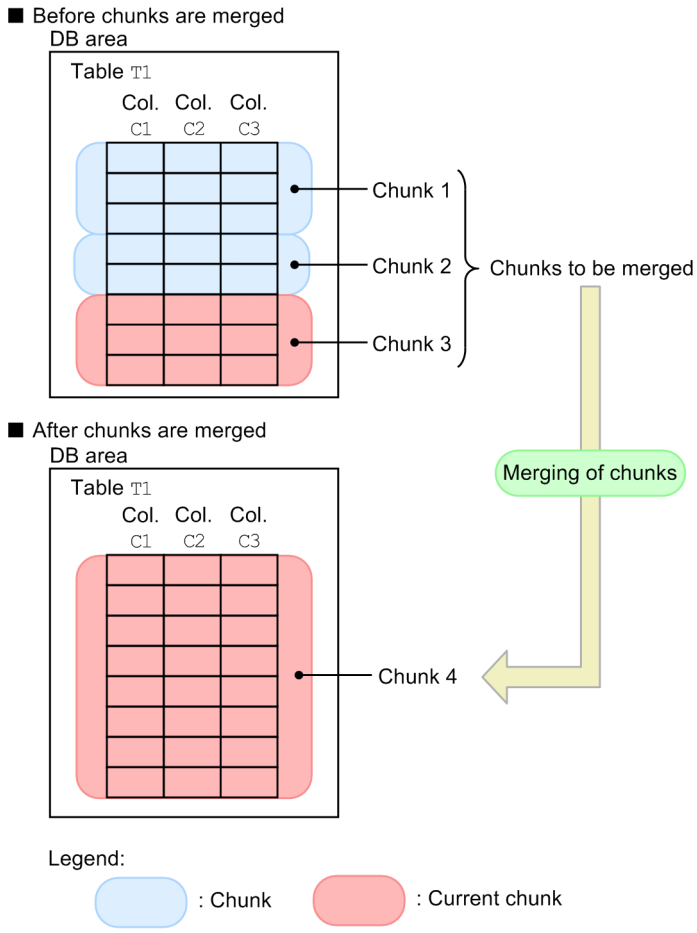
If the number of chunks increases as a result of repeated background import, the performance of retrieval using B-tree indexes and text indexes might decline. Reducing the total number of chunks by merging them can prevent such deterioration of retrieval performance.

■ Relationship between the merging of chunks and the number of chunks that can be created

In HADB, there is a limit to the number of chunks that can be managed in a single DB area and a single base table. Reducing the total number of chunks by merging them can prevent this upper limit from being reached. For details about the maximum number of chunks in a single DB area and the maximum number of chunks that can be created in a single base table, see D.1 Maximum and minimum values related to system configuration.

The following figure shows an example of merging multiple chunks.

Figure 2-45: Example of merging multiple chunks



Explanation

Three chunks which include the current chunk (chunks 1 through 3) are merged. After the merger, the three chunks are managed as a single new chunk (chunk 4). In this case, the current chunk changes from chunk 3 to chunk 4.

Note

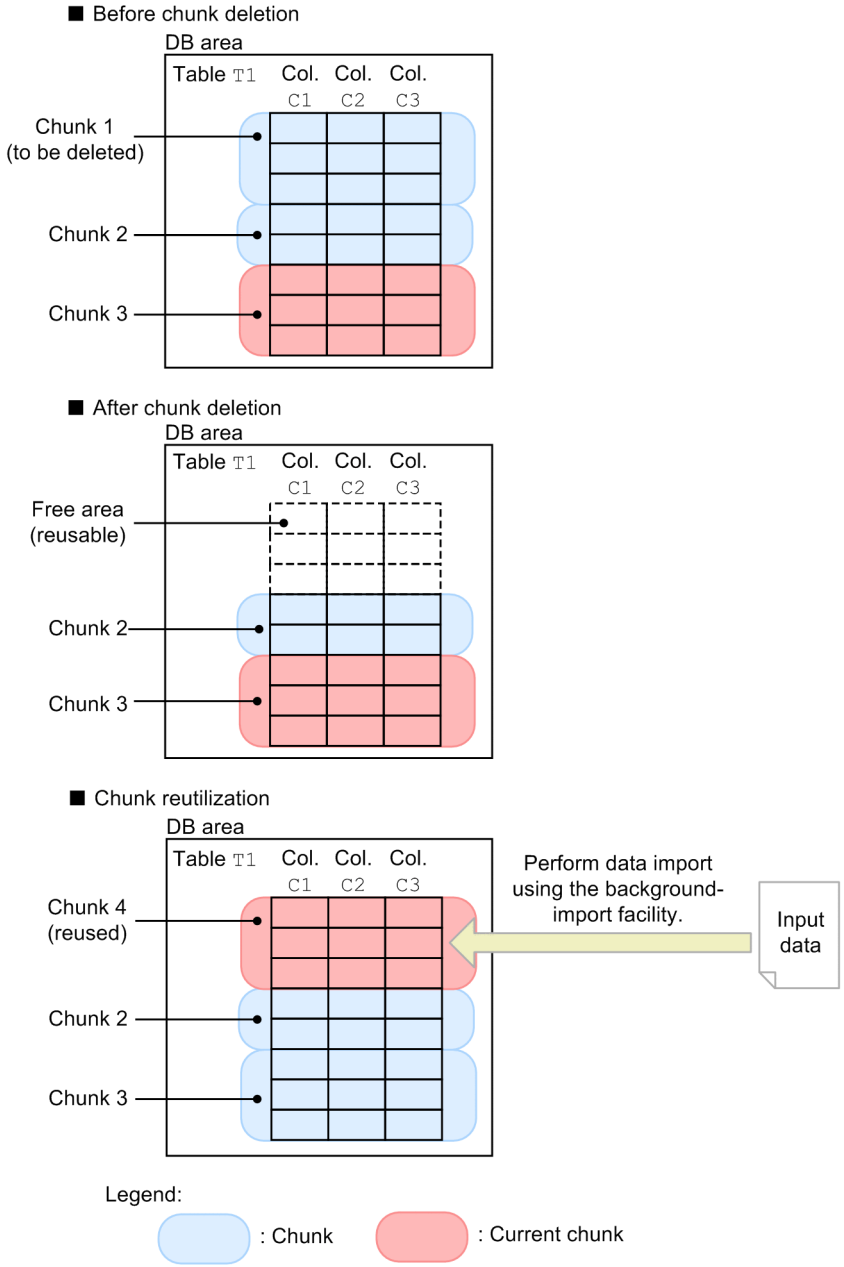
If the current chunk is not included in the chunks designated for merging, the current chunk does not change.

(3) Deleting chunks

Data stored in a base table can be deleted in units of chunks. When a chunk is deleted, its data storage area can be reused, unlike when a row is deleted. Note, however, that the current chunk cannot be deleted.

The following figure shows an example of deleting chunks and reusing the storage area.

Figure 2-46: Example of deleting chunks and reusing the storage area



Explanation

When chunk 1 is deleted, the area that was used by chunk 1 can be reused. When background import is executed on table `T1` after chunk 1 has been deleted, a new chunk (chunk 4) is created in the area that was used by chunk 1, and data is stored in it.

(4) Changing the chunk status

A chunk can be in one of several statuses.

When you perform background import, you can specify the status of the chunks to be created. You can also change the chunk status after background import.

The following table shows the chunk statuses that are managed by the HADB server.

Table 2-19: Chunk statuses

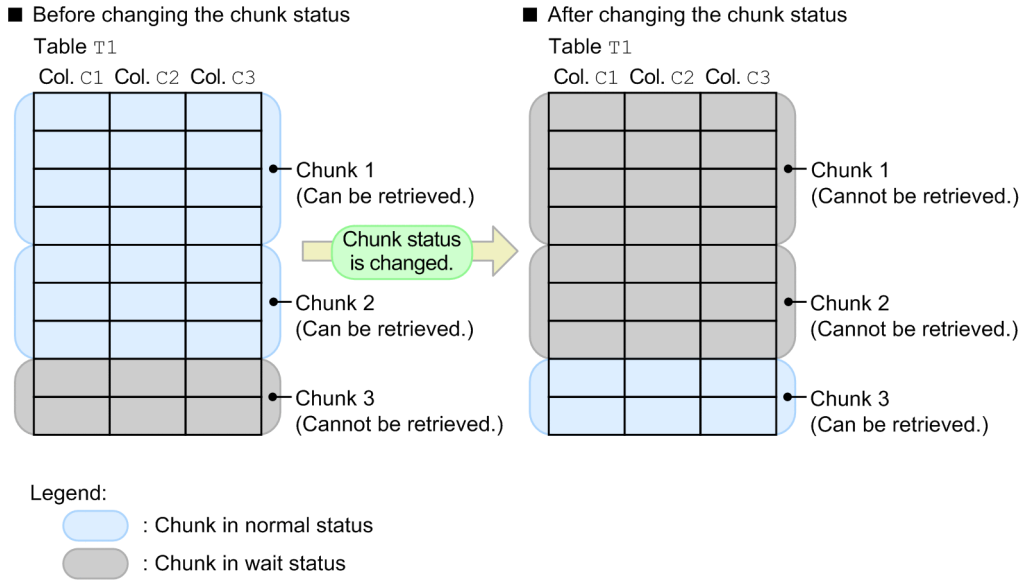
No.	Chunk status	Explanation
1	Normal status	When a chunk is in normal status, the data contained in the chunk can be manipulated by a data manipulation SQL statement. Therefore, data contained in chunks that are in normal status can be retrieved. A chunk in normal status is created when background import is executed. You can also change a chunk's status from normal status to wait status.
2	Wait status	When a chunk is in wait status, the data it contains cannot be manipulated by a data manipulation SQL statement. Therefore, such data cannot be retrieved. However, you can delete the chunk by using the `PURGE CHUNK` or `TRUNCATE TABLE` statement. A chunk in wait status is created when you specify this status for the chunk to be created during performance of background import. You can also change a chunk's status from wait status to normal status. Note that a chunk in wait status cannot become the current chunk. Using a chunk in wait status, you can perform the following operations: • You can use background import to store data in a chunk that is in wait status, and then change the chunk's status to normal status at a desired time so that the data can be retrieved. • You can make data stored in a chunk in normal status non-retrievable by changing the chunk's status from normal status to wait status at a desired time.
3	Delete-pending status	A delete-pending chunk is a chunk whose data cannot be manipulated by a data manipulation SQL statement. Therefore, such data cannot be retrieved. However, you can delete the chunk by using the `PURGE CHUNK` or `TRUNCATE TABLE` statement. A delete-pending chunk is an unneeded (merge-source) chunk that remains without being deleted because merge chunk processing was interrupted. You cannot change the status of a delete-pending chunk. Unlike chunks in wait status, delete-pending chunks are not needed and therefore must be manually deleted.

Note:

Depending on the status of a chunk, the SQL statements and commands that can be executed on the data contained in the chunk vary. For details, see (2) Relationship of chunk statuses with SQL statements and commands that can be executed in 11.4.12 Changing the chunk status.

The following figure shows an example of changing the statuses of multiple chunks that were created by executing background import.

Figure 2-47: Example of changing chunk statuses



Explanation

In this example, the status of two chunks (chunks 1 and 2) is changed from normal status to wait status. Once the chunk status changes to wait status, the data stored in chunks 1 and 2 can no longer be retrieved.

Additionally, a chunk in wait status (chunk 3) is changed to normal status. After the chunk status has become normal status, the data stored in chunk 3 can be retrieved.

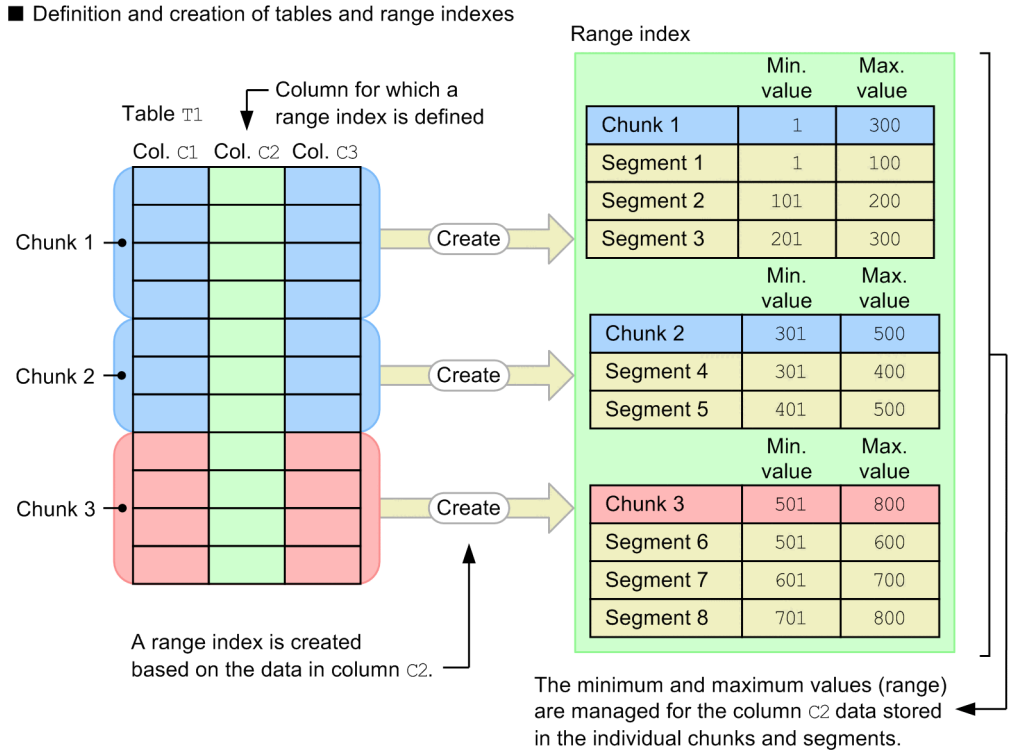
2.14.3 Relationship between chunks and range indexes

This section explains the relationship between chunks and range indexes. For details about range indexes, see 2.3.3 Range indexes.

A range index manages the range of data stored within segments of a base table (the minimum and maximum values of a column's data), as well as the range of data stored within chunks. By using a range index, you can skip retrieval processing of chunks and segments that store ranges of data that do not satisfy a search condition. This improves retrieval performance.

The following figure provides an overview of range indexes, chunks, and segments.

Figure 2-48: Overview of range indexes, chunks, and segments



Explanation

A range index is defined for column `C2` of table `T1`. Executing background import on table `T1` stores data in the table. A range index is also created based on the data in column `C2`. These groups of data are managed as a single chunk.

In this case, the range index manages the range of table data within segments as well as the range of data within the chunk.

(1) Retrieval using a range index (skipping of chunks)

When a table is retrieved using a range index, retrieval processing of chunks and segments that store ranges of data that do not satisfy the search condition can be skipped in stages, as described in the following.

■ Retrieval processing using a range index

1. Skipping of chunks
2. Skipping of segments

> **Important**
>
> A range index defined in a version earlier than version 02-02 cannot skip chunks.

The following describes these two types of retrieval processing.

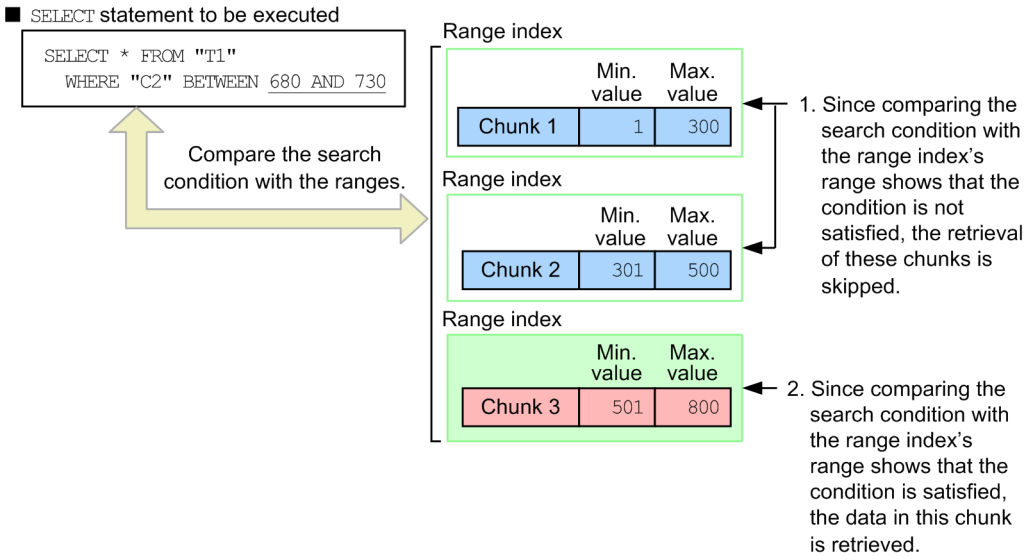
1. Skipping of chunks

   In table retrieval using a range index, retrieval processing can first be skipped in units of chunks. HADB determines whether a chunk stores ranges of data that satisfy the search condition, and does not retrieve the chunk if it stores ranges of data that do not satisfy the search condition.

   In this case, retrieval performance improves because table data, B-tree index data, and text index data are not accessed.

   The following figure provides an overview of chunk skipping.

   Figure 2-49: Overview of chunk skipping

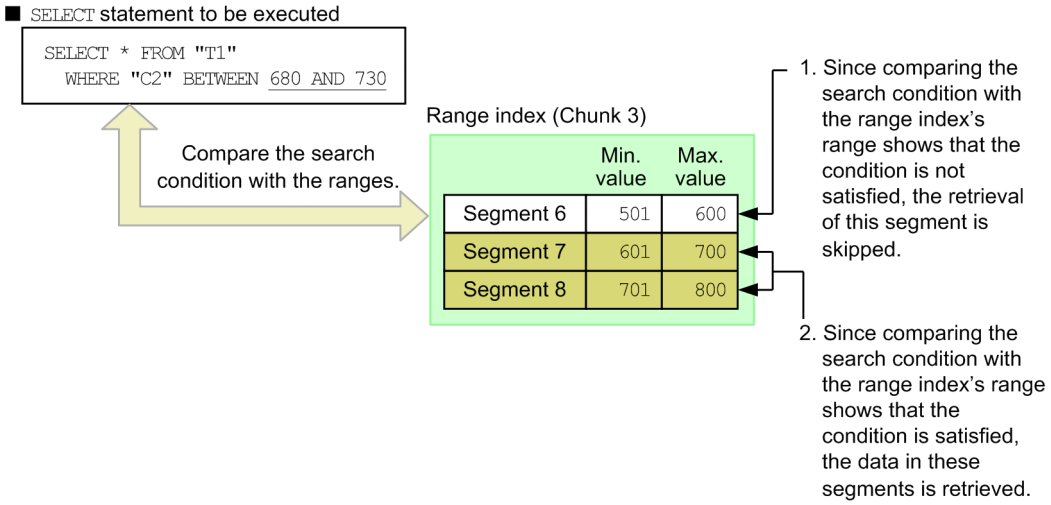


2. Skipping of segments

   If a comparison between a search condition and the range specified by a range index shows that chunk 3 satisfies the condition, retrieval processing is performed on the table segments within chunk 3. During this process, the retrieval of segments that store ranges of data that do not satisfy the search condition can be skipped in the same way as described in 2.3.3 Range indexes.

   The following figure provides an overview of segment skipping.

Figure 2-50: Overview of segment skipping



(2) Range index characteristics (skipping of chunks)

This section explains the characteristics of a range index that can skip chunks, and the four major characteristics of a range index explained in (2) Range index characteristics under 2.3.3 Range indexes.

(a) Using B-tree indexes or text indexes together with range indexes

Range indexes that can skip chunks can be used together with B-tree indexes and text indexes.

Also described below is retrieval processing that uses a B-tree index or a text index.

■ Retrieval using a range index that can skip chunks and a B-tree index

1. Retrieval using a range index (skipping of chunks)

   First, retrieval using a range index is executed. This process determines whether a chunk stores ranges of data that satisfy the search condition, and it skips retrieval of the chunk if it stores ranges of data that do not satisfy the search condition. Skipping chunks narrows the chunks to those that satisfy the search condition.

2. Retrieval using a B-tree index

   After the chunks have been narrowed to those that satisfy the search condition, retrieval is performed using the B-tree index within the target chunks.

■ Retrieval using a range index that can skip chunks and a text index

1. Retrieval using a range index (skipping of chunks)

   First, retrieval using a range index is executed. This process determines whether a chunk stores ranges of data that satisfy the search condition, and it skips retrieval of chunks that store ranges of data that do not satisfy the search condition. Skipping chunks narrows the chunks to those that satisfy the search condition.

2. Retrieval using a text index

   After the chunks have been narrowed to those that satisfy the search condition, retrieval is performed using the text index within the target chunks.

> **Note**
>
> If a range index is used together with a B-tree index or text index, retrieval using a range index (segment skipping) is not performed.

(b) Using multiple range indexes

There is no difference. For details about using multiple range indexes, see (b) Multiple range indexes can be used in (2) Range index characteristics under 2.3.3 Range indexes.

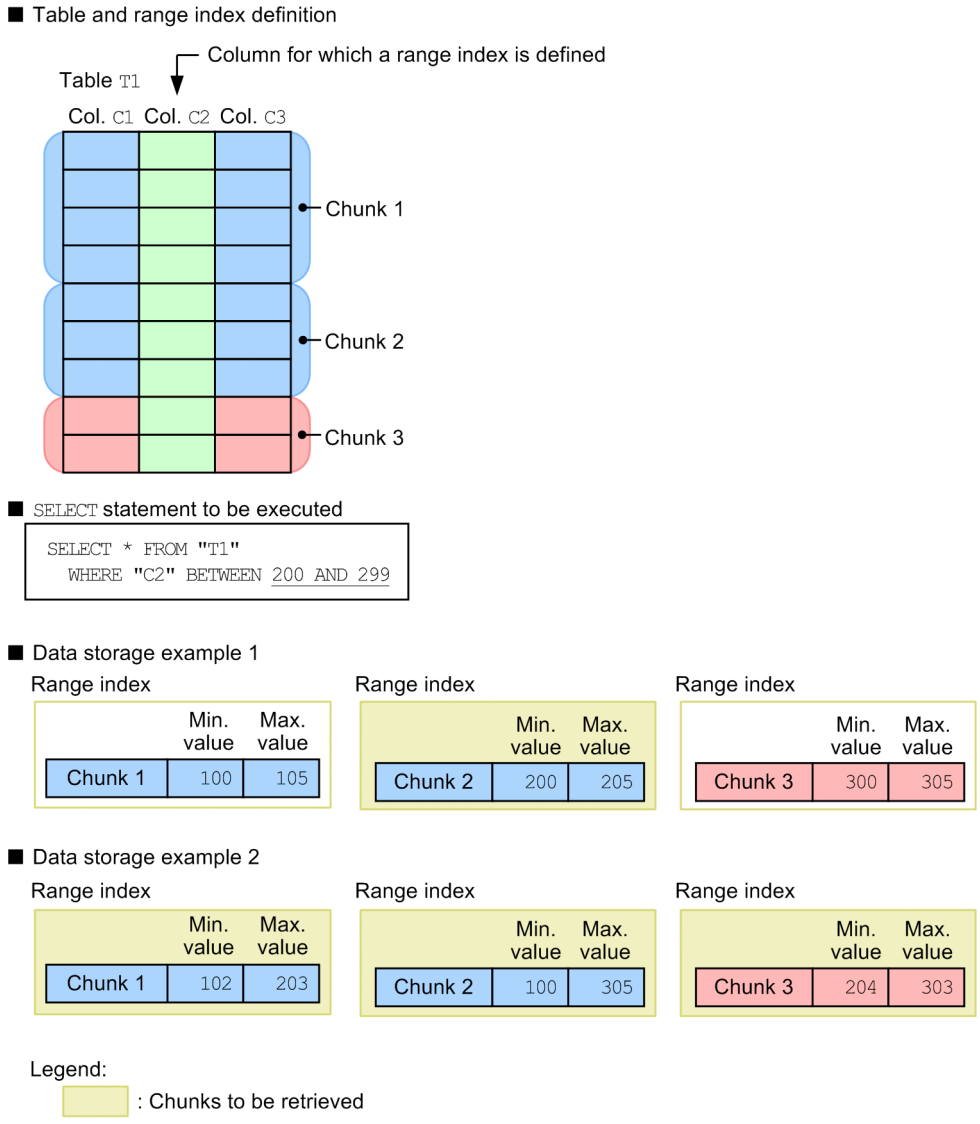
(c) When the amount of data is less than that in the B-tree index or text index

There is no difference. For details about situations when the amount of data is less than that in the B-tree index or text index, see (c) Smaller data size than a B-tree index or a text index in (2) Range index characteristics under 2.3.3 Range indexes.

(d) Variation in retrieval performance depending on the ranges of data stored in chunks

The performance of retrieval using a range index varies depending on the ranges of data stored in chunks.

Figure 2-51: Examples in which retrieval performance varies depending on the ranges of data stored in chunks



Explanation

In data storage example 1, retrieval processing is performed on a single chunk. On the other hand, in data storage example 2, there is a range overlap and the range in each chunk is wider than in storage example 1. Therefore, retrieval processing is performed on three chunks. In this way, the retrieval performance varies depending on the range of data stored in chunks even when the same search condition is used.

(3) Updating the ranges in a range index (chunk and segment)

When addition and update processing are performed repeatedly on rows, the ranges of chunks and segments managed by the range index expand. As a result, the benefits of using the range index might begin to diminish (that is, the ranges never become narrower). When deletion processing is performed repeatedly on rows, the range index's ranges might become too wide for the ranges of data stored in the chunks and segments, and as a result, the benefits of using the range index diminish.

When the benefits of using a range index diminish, re-create the range index by performing index rebuilding.

The following table shows the conditions that result in the expansion of the range index's ranges.

Table 2-20: Conditions that expand the ranges in a range index (chunks and segments)

No.	Operation	Condition that expands ranges in a range index
1	Row addition	When a value outside a range is added, the range of the chunk and segment that store the added row expands to accommodate the added value.
2	Row update	When a value is updated so that it is no longer within a range, the range of the chunk and segment that store the updated row expands to accommodate the updated value.
3	Row deletion	No range expansion occurs.

2.15 Chunk archiving function (compressing data in a chunk)

This section describes the chunk archiving function, which compresses data in a chunk.

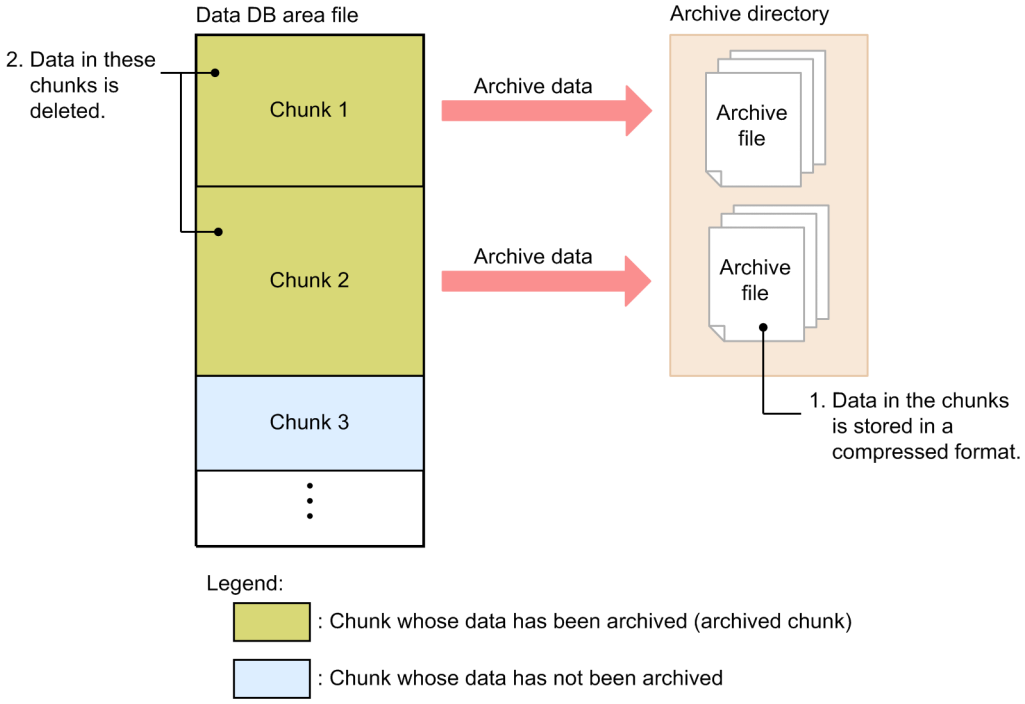
The chunk archiving function can be used for only archivable multi-chunk tables. For details about archivable multi-chunk tables, see (2) What is a multi-chunk table? in 2.2.2 Single-chunk tables and multi-chunk tables.

2.15.1 Overview of the chunk archiving function

Data such as sensor data and sales data increases by a large amount every day. If you need to store such data for a long term, the storage capacity might become low or the cost of adding storage might exceed your expectation. If you are dealing with such problems, you can compact the database size by compressing and outputting the data in the DB area to a file. This processing is called archiving. Data can be archived on a chunk basis. This is called the chunk archiving function.

The following figure shows an overview of archiving data by using the chunk archiving function.

Figure 2-52: Archiving data by using the chunk archiving function



Explanation

1. The data in a chunk is compressed and output (archived) to a file. This file is called an archive file. When a chunk is archived, one or more archive files are created.

   The directory in which archive files are stored is called an archive directory.

   Only table data is archived. Index data is deleted without being archived.

2. When a chunk is archived, the data in the chunk is deleted.

   A chunk whose data has been archived is called an archived chunk.

The preceding sequence of tasks is called chunk archiving. To archive a chunk, execute the `adbarchivechunk` command.

Note

If the chunk archiving function is used, the database size can be reduced. However, searching archived data might take longer than searching non-archived data for the following reasons:

- Time is required to expand the data.
- Index-based searches cannot be performed.

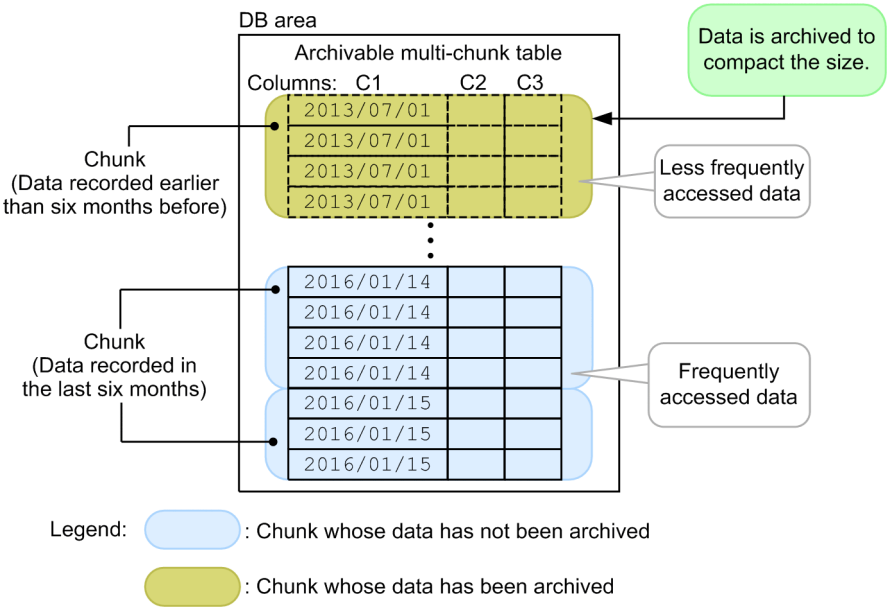
2.15.2 When the chunk archiving function is to be used

We recommend that you use the chunk archiving function when all of the following conditions are met:

- A large amount of chronological data, such as sensor data and sales data, is imported every day.
- The preceding chronological data must be stored for a long term.
- Recent data is frequently referenced and old data is infrequently referenced for analysis.

The following figure shows an example of using the chunk archiving function.

Figure 2-53: Example of using the chunk archiving function



Explanation

In this example, the data stored in the last six months is not archived because the data is frequently referenced. The other, older, data is archived to compact the data size because it is infrequently referenced.

A search in archived data takes more time than a search in non-archived data.

Important

- The data to be archived by the chunk archiving function must include datetime information.

- Archived data cannot be updated or deleted. Before you can update or delete archived data, you must expand and restore it in the DB area. Therefore, we recommend that you do not use the chunk archiving function for data that might be updated or deleted in the future.

2.15.3 Searching an archivable multi-chunk table

When an archivable multi-chunk table is defined, a range index is automatically defined for the archive range column. A location table is also automatically created.

The following describes the archive range column and location table.

(1) Archive range column

The HADB server manages the range (maximum and minimum values) of data that is stored in an archivable multi-chunk table for each chunk. The range of data managed for each chunk can be used to narrow down the search range. Therefore, although the amount of data that has been managed for a long term is large, you can perform a search for only the data you need.

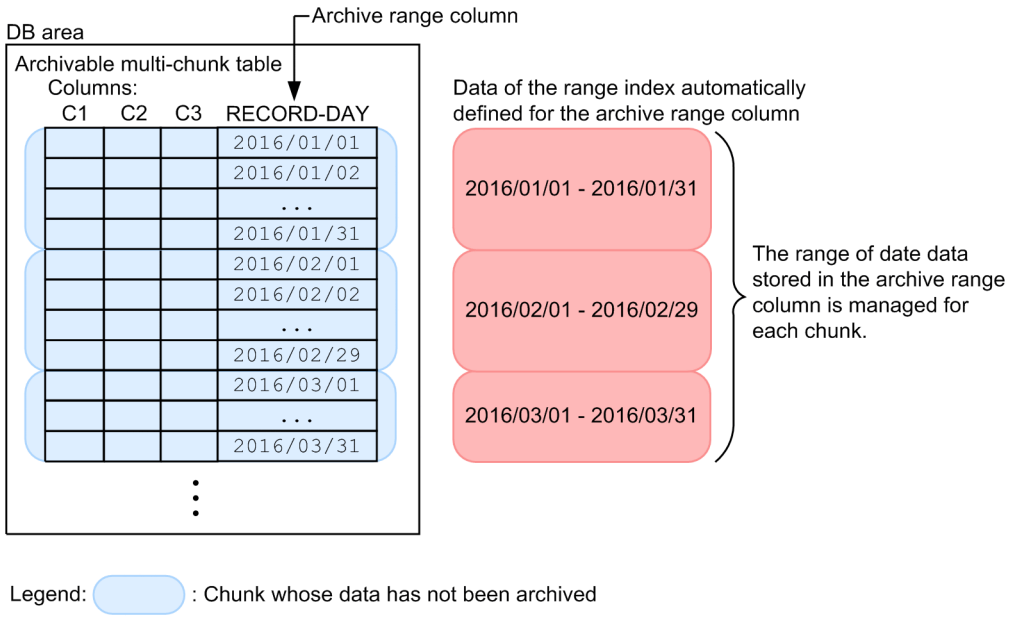
A column that contains data and is used to narrow down the search range is called the archive range column.

Any archivable multi-chunk table requires a column that is to be used as the archive range column. For example, a column that contains datetime data or numeric data can be used as the archive range column.

For the archive range column, the HADB server automatically defines a range index. The HADB server manages the range (maximum and minimum values) of data stored in the archive range column for each chunk that has not been archived.

The following figure shows an overview of the archive range column.

Figure 2-54: Overview of the archive range column (if a date data column is used as the archive range column)



(2) Location table

When an archivable multi-chunk table is defined, the HADB server automatically defines a location table and the index for the location table. A location table is a table that the system uses to manage the information about archived chunks. The following shows the information that is managed by using a location table.

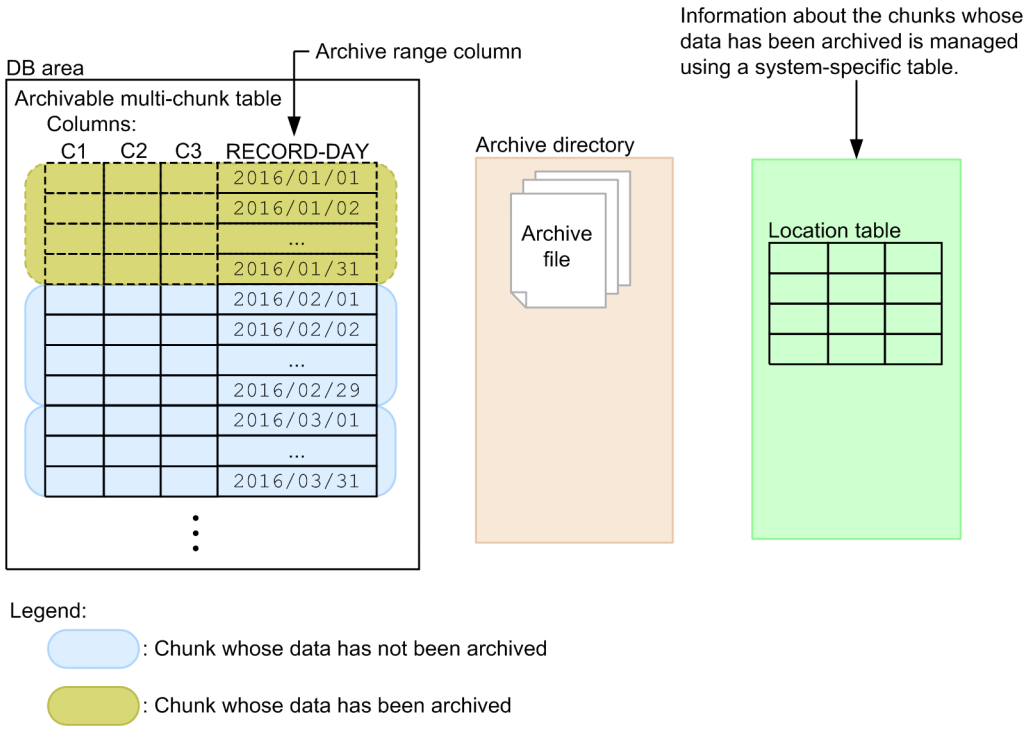
Information managed by using a location table

- Chunk IDs
- Paths of archive files
- Range of values (maximum and minimum values) in the archive range column of each archive file

The index data of a chunk is excluded when the chunk is archived. Therefore, the data in an archived chunk cannot be searched by using an index. Instead, the search range can be narrowed down by managing the data of the archive range column in a location table.

The following figure shows an overview of a location table.

Figure 2-55: Overview of a location table



Note

- Location tables are used by the HADB server and are not used by users.
- One location table is created for each archivable multi-chunk table.
- The location table for an archivable multi-chunk table and the index defined for that location table are stored in the data DB area that stores the archivable multi-chunk table.

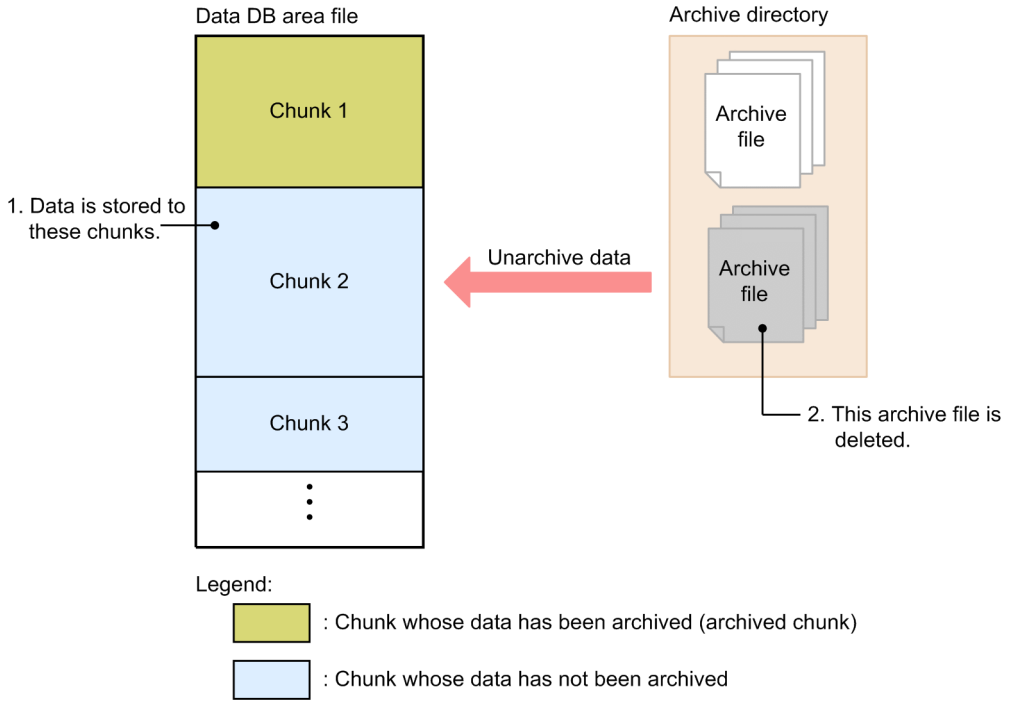
When the data in an archivable multi-chunk table is searched, the HADB server narrows down the search-target archive files based on the data stored in the archive range column. For details, see *Considerations when searching an archivable multi-chunk table* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

2.15.4 Expanding (unarchiving) data

Data that is archived cannot be updated or deleted. Therefore, if a need to update or delete archived data arises, you must expand (unarchive) the data. If the frequency of referencing archived data has become higher, consider whether you need to unarchive the data.

The following figure shows an overview of unarchiving data by using the chunk archiving function.

Figure 2-56: Unarchiving data by using the chunk archiving function



Explanation

In the preceding figure, the data in Chunk 2 has been archived.

1. The data is retrieved to Chunk 2 from the corresponding archive file. At this time, an index is also created.
2. When the data is stored in the chunk and an index is created, the archive file is deleted.

The preceding sequence of tasks is called chunk unarchiving. To unarchive a chunk, execute the `adbunarchivechunk` command.

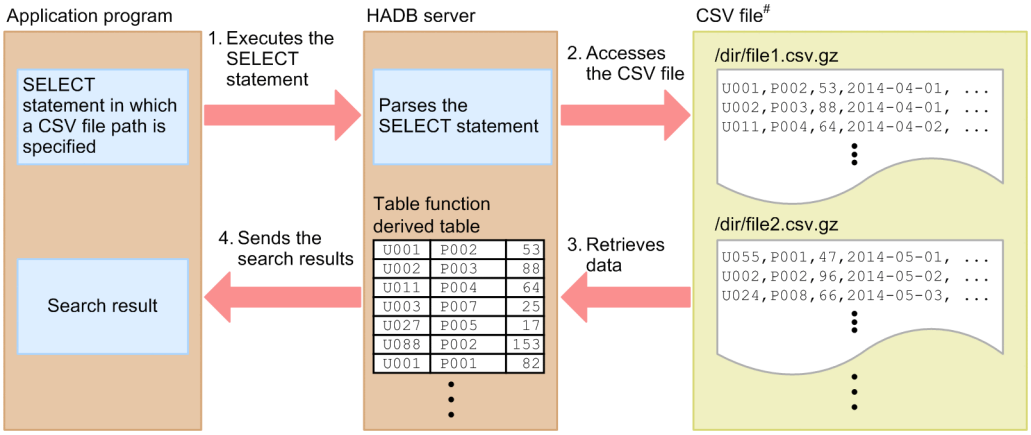
> **Note**
>
> Data storage and index creation are performed by the data import processing of the `adbimport` command.

2.16 Retrieving data from CSV files

HADB enables you to use the SELECT statement to retrieve data in CSV files (CSV-format files). For example, if your last year's sales management data is stored in a CSV file, you can retrieve the sales management data without having to import it to a table.

To do this, execute the SELECT statement with the path of the target CSV file specified. If you specify a search condition in the SELECT statement, you can retrieve the data that satisfies the condition in the same manner as when you retrieve data that has been imported to a table. The following figure provides an overview of data retrieval from CSV files.

Figure 2-57: Overview of data retrieval from CSV files



#

Data can be retrieved from CSV files that are not compressed or that are compressed in GZIP format.

Explanation

1. Executes the SELECT statement with the CSV file path specified.

   Specify the CSV file path in a function provided by HADB (ADB_CSVREAD function for retrieving data from CSV files).
2. The HADB server accesses the CSV file whose path is specified in the function.
3. The HADB server retrieves data from the CSV file and stores it temporarily as a table. This set of data stored temporarily in tabular format is called a *table-function derived table*.
4. The HADB server retrieves the table-function derived table according to the search condition specified in the SELECT statement and returns the retrieval results to the application program.

■ Example of SELECT statement with the ADB_CSVREAD function specified

This example retrieves data in the following CSV files that have been compressed in GZIP format:

- /dir/file1.csv.gz
- /dir/file2.csv.gz
- /dir/file3.csv.gz

```
SELECT * FROM TABLE(ADB_CSVREAD(
           MULTISET['/dir/file1.csv.gz','/dir/file2.csv.gz','/dir/file3.csv.gz'],
          'COMPRESSION_FORMAT=GZIP;
           FIELD_NUM=1,2;
           ENCLOSING_CHAR=";
           DELIMITER_CHAR=,;')
           AS "T1" ("C1" INTEGER, "C2" VARCHAR(32))
```

Explanation

The underlined parts indicate the information specified in the `ADB_CSVREAD` function.

The information specified in the `ADB_CSVREAD` function includes the absolute path names of the CSV files and the data to be retrieved from the CSV files.

For details about the `ADB_CSVREAD` function, see *ADB_CSVREAD function* under *System-defined functions* in *Constituent Elements* in the manual *HADB SQL Reference*.

Important

- The CSV files must meet any of the following conditions:
  · CSV files that are not compressed
  · CSV files that are compressed in GZIP format by using the OS command `gzip`
  · CSV files that are compressed in GZIP format and exported by using the `adbexport` command
- The data in such a CSV file cannot be updated.
- Retrieval of data from a CSV file requires more time than retrieval of data that has been imported to a table because the CSV file must be analyzed.

2.17 Searching text data

This section describes convenient text data search functions.

2.17.1 Correction search

The following describes the correction search in text data.

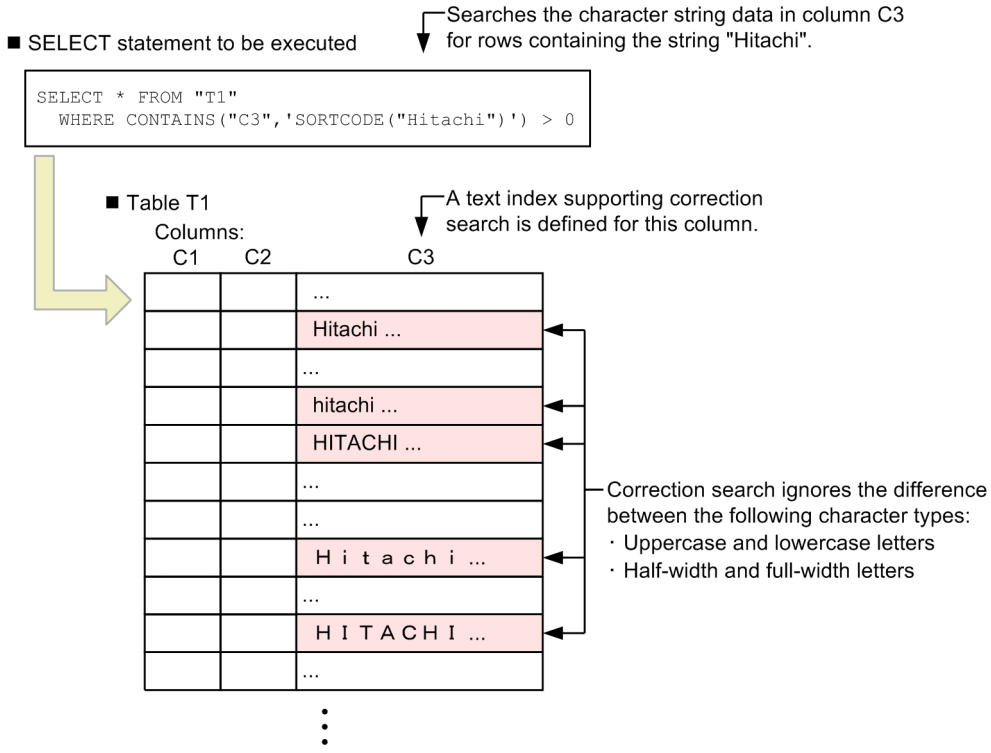
(1) Overview of correction search

Correction search is a function that performs a text data search, ignoring the differences between uppercase and lowercase letters, between half-width and full-width characters, and between Japanese hiragana and katakana characters. For example, if you specify `Hitachi` as a search string, the function searches not only for `Hitachi`, but also for `HITACHI` and `ＨＩＴＡＣＨＩ` at the same time. The correction search function is useful when searching text data that has notational inconsistencies because you can perform a collective search as if you execute an SQL statement in which multiple search conditions are combined by using the `OR` operator.

Notational inconsistencies are likely to arise in text data created by multiple persons, such as log records of a call center and applications for use of products. If you want to retrieve from such text data all records that include a specific keyword (such as the name of a company, product, or person in charge), you can conveniently use the correction search function.

The following figure shows an example of correction search.

Figure 2-58: Example of correction search



Explanation

If `Hitachi` is specified as a search condition, the function retrieves all character strings that include the word `Hitachi`, irrespective of character case or width.

To perform correction search, use the scalar function CONTAINS. For details about the scalar function CONTAINS, see *CONTAINS* in *Character string functions (acquisition of character string information)* in *Scalar Functions* in the manual *HADB SQL Reference*.

Note that if you perform correction search, you can reduce the number of pages to be loaded by defining a text index that supports correction search. This improves table retrieval performance.

Important

The correction search function can be used if the character encoding used on the HADB server is Unicode (the value specified for the environment variable ADBLANG is UTF8). It cannot be used if the character encoding is Shift-JIS (the value specified for the environment variable ADBLANG is SJIS).

(2) Rules of correction search

The following table describes the rules of correction search.

Table 2-21: Rules of correction search

No.	Character type	Rule	Example of correction search
1	Alphabetic character	Correction search ignores the differences among the following character types: • Uppercase letter • Lowercase letter • Half-width alphabetic character • Full-width alphabetic character	If max is specified as a search condition, the correction search function searches for the following character strings: ・"max" ・"MAX" ・"ＭＡＸ" ・"ｍａｘ"
2	Number	Correction search ignores the differences between the following character types: • Half-width number • Full-width number	
3	• Hiragana character • Katakana character	**▪ Japanese hiragana and katakana characters** Correction search ignores the differences between the following character types: • Hiragana character • Katakana character **▪ Full-width and half-width katakana characters** Correction search ignores the differences between the following character types: • Full-width katakana character • Half-width katakana character **▪ Japanese dakuten and handakuten marks** If one of the following characters is followed by a full-width or half-width dakuten mark (voiced sound mark), the correction search function assumes that the character and sign make up a single character. • Hiragana character • Full-width katakana character • Half-width katakana character	**▪ Example 1** If you specify "りんご" as a search string, correction search treats the following strings as search strings: • "りんご" • "リンゴ" • "ﾘﾝｺﾞ" • "りんこ゛" • "りんこﾞ" **▪ Example 2** If you specify "バット" as a search string, correction search treats the following strings as search strings: • "バット" • "ﾊﾞｯﾄ" • "ばっと" • "は゛つと" **▪ Example 3** If you specify "プリンター" as a search string, "プリンタ" is not treated as a search string. The difference between "プリンタ" and "プリンター" is acknowledged.

No.	Character type	Rule	Example of correction search
		The same rule also applies to a full-width or half-width handakuten mark (semi-voiced sound mark). ▪ **Japanese youon and sokuon signs** The correction search function equates Japanese youon and sokuon signs to their corresponding regular-sized characters. ▪ **Japanese ombiki sign** The correction search function does not ignore whether the ombiki sign is used, and treats the sign as a single ordinary character.	
4	Diacritical marks	The correction search function ignores the difference between the following character types: • Characters with a diacritical mark (such as umlaut) • Characters without a diacritical mark (such as umlaut)	The correction search function assumes the following characters to be the same: ・"A" ・"ä" ・"Ä" ・"â" Therefore, if MAX is specified as a search condition, the correction search function searches for the following character strings: ・"MAX" ・"mäx" ・"MÄX" ・"mâx"
5	Single character representing a specific character string	If the correction search function encounters a character that represents a specific character string, the function expands the character to the character string. The function equates the character to the character string.	The correction search function assumes the following characters to be the same: ■ Example 1 "㌘" and "グラム" ■ Example 2 "㎏" and "kg" If you specify "㌘" (in Example 1) as a search string, correction search treats the following strings as search strings: • "㌘" • "グラム" • "ぐらむ" • "ｸﾞﾗﾑ"

For hiragana, full-width katakana, and half-width katakana characters, a character with a dakuten or handakuten sign and a character without a dakuten or handakuten sign are assumed to be different. Therefore, the correction search function does not ignore the difference between those characters. The following shows examples. In the following combinations of characters, each combination uses the same (dakuten or handakuten) mark. Therefore, both characters are assumed to be the same character.

- は and ハ
- ば and バ
- ぱ and パ

On the other hand, は , ば , and ぱ do not have the same (dakuten or handakuten) mark. Therefore, these characters are assumed to be different. For example, if バイク is specified as a search condition, バイク and ばいく can be retrieved. However, ハイク , はいく , パイク , and ぱいく cannot be retrieved because the correction search function does not ignore the difference among them.

Correction search uses sort codes, which are codes specified in the ISO/IEC 14651:2011 standard for sorting and comparing characters. The characters of the same sort code are assumed to be the same character.

Note

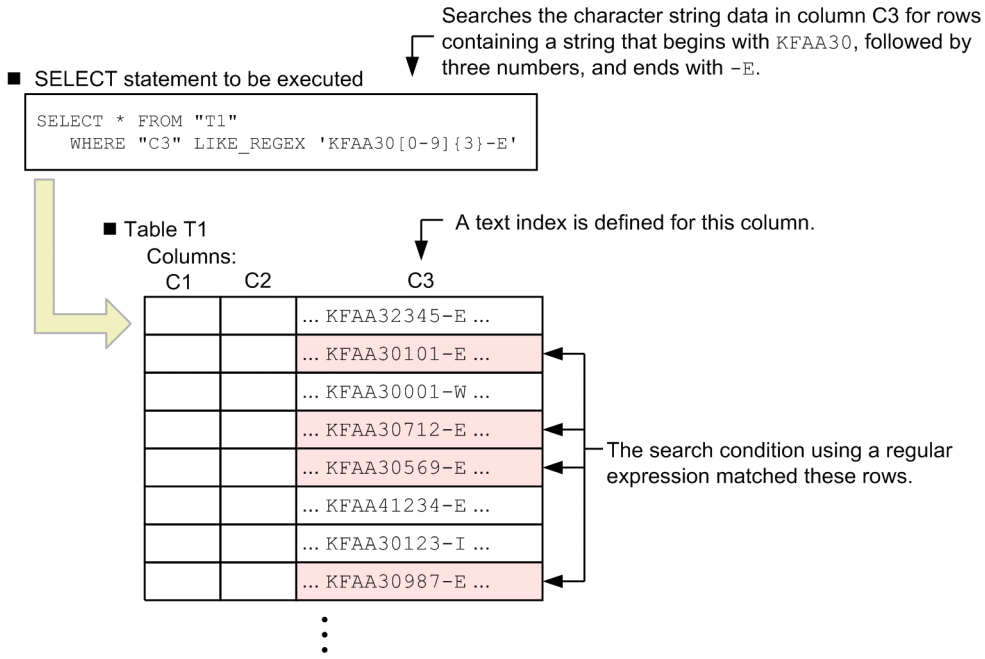
- For symbols and other characters that are assumed to be non-existent in the sort codes, the bytecode is used for sorting and comparison.
- The sort code is used if `SORTCODE` (*simple-string-specification*) is specified as *notation-correction-search-specification* of the scalar function `CONTAINS`.

2.17.2 Search using a regular expression

You can use a regular expression to search text data. A regular expression is a means to represent complex search conditions in a single, short conditional expression. For example, you can search for the character string `ABC` followed by five numbers.

The following figure shows an example of a search using a regular expression.

Figure 2-59: Example of a search using a regular expression



To perform a search using a regular expression, specify the `LIKE_REGEX` predicate. For details about the `LIKE_REGEX` predicate, see *LIKE_REGEX predicate* in *Predicates* in *Constituent Elements* in the manual *HADB SQL Reference*.

Note that when you use a regular expression to search text data, you can reduce the number of pages to be loaded by using a text index. This improves table retrieval performance.

Note

You can use POSIX-based regular expressions (extended regular expressions compliant with POSIX1003.2). Note the following exceptions:

- The collation element (`[. .]`) and character equivalence class (`[= =]`) cannot be used in a character list.
- The back reference (*n*) (*n* is an integer in the range from `1` to `9`) cannot be used.
- To use a special character as a literal, you must always add an escape character. You must also add an escape character when using a right square bracket (`]`) as a literal in a character list or using a caret (`^`) as a literal rather than to mean the beginning of a string.
- Some operators for Perl extensions can be used. For details about the representations that can be used, see the description of character class in *LIKE_REGEX predicate* in *Predicates* in *Constituent Elements* in the manual *HADB SQL Reference*.

Important

- You can use a regular expression to create, for example, a representation that matches multiple characters or means some repetitions. Note that, depending on the regular expression that you specify, column data might be evaluated multiple times. Therefore, if the length of column data is long, a search using a regular expression might consume a large amount of memory. Also, the search might require a long time.
- If you specify a large number as a repetition factor or specify nested repetition factors in succession, the search might consume a large amount of memory. Also, the search might require a long time.

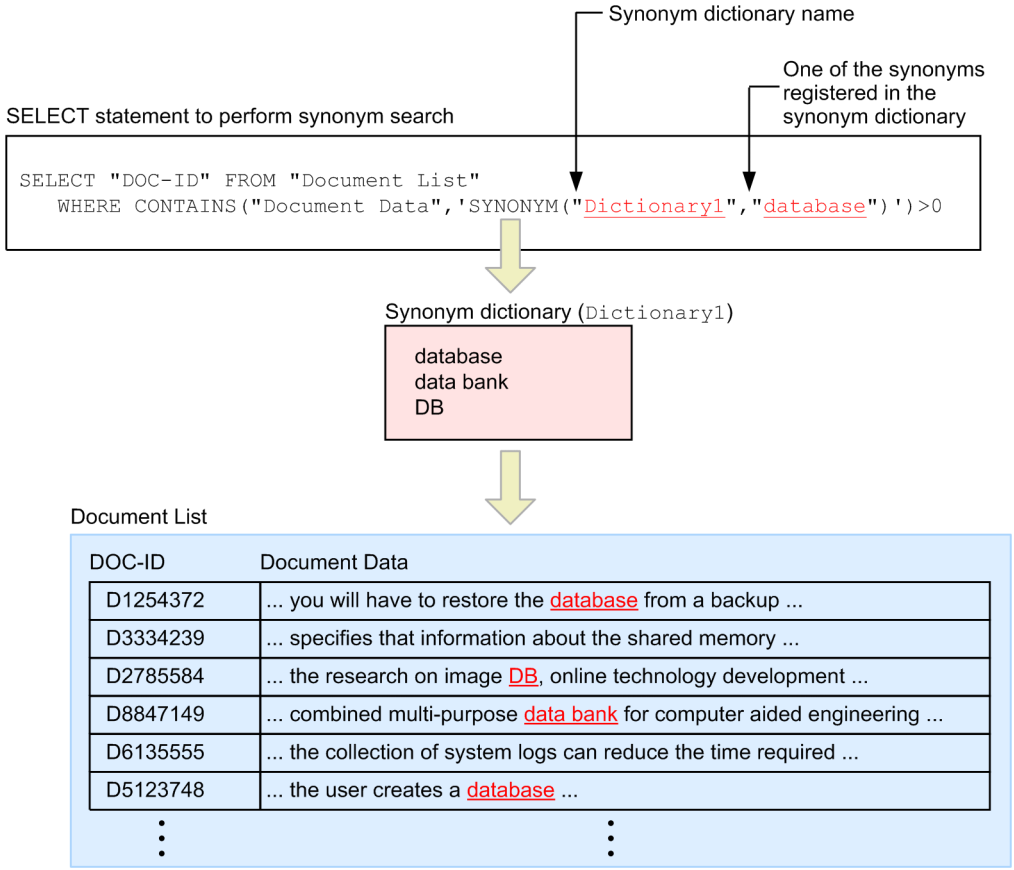
2.17.3 Synonym search

The following describes the synonym search in text (document) data.

(1) Overview of synonym search

When you search literatures, research papers, and other document data for a specific word, you can also search for synonyms of the word at the same time. This function is called synonym search. Before you can perform a synonym search, you must register the target word and its synonyms as a list in a dictionary. This dictionary is called a synonym dictionary. The following figure shows an overview of synonym search.

Figure 2-60: Overview of synonym search



Explanation

- Before you can perform a synonym search, you must create a synonym dictionary. In this example, a synonym dictionary named `Dictionary1` is created. In the synonym dictionary, `database`, `data bank`, and `DB` are registered as synonyms.
- You can perform a synonym search by specifying the scalar function `CONTAINS` in the `SELECT` statement. For the scalar function `CONTAINS`, specify the name of the synonym dictionary to be used and one of the synonyms that are registered in the dictionary. In the preceding example, `database` is specified as one of the synonyms.
- When the preceding `SELECT` statement is executed, the synonym search function searches the document data for `database`, `data bank`, or `DB`.

▪ Use in English text search

For search in English text, you can register inflected forms of a word in a synonym dictionary. For example, assume that you register `electromagnet`, `electromagnets`, and `electromagnetic` in a synonym dictionary. If you specify `electromagnet` as a search string when performing a synonym search, `electromagnet`, `electromagnets`, and `electromagnetic` are all assumed to be search strings.

Example:

```
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary2","electromagnet")')>0
```

Note

For details about the environment setup and usage of synonym search, see 11.16 Performing synonym search operations.

(2) Combined use with correction search

You can use both synonym search and correction search at the same time. You can specify the scalar function CONTAINS to use both synonym search and correction search.

Example:

```
SELECT "Document ID" FROM "List of Documents"
    WHERE CONTAINS("Document Data",'SYNONYM("Dictionary1",SORTCODE("database"))')>0
```

When the preceding SELECT statement is executed, the synonym search function searches the document data for database, data bank, and DB. In addition, the correction search function searches the document data that includes character strings such as DATABASE, Data bank, and db.

(3) Relationship with text indexes

When you perform synonym search, you might be able to improve the search performance by using a text index. Therefore, we recommend that you define a text index for the column that contains the document data to be searched.

Note that if you use a text index when using both synonym search and correction search, you must define a text index that supports correction search. Also note that a text index that supports correction search is larger in size than a text index that does not.

2.17.4 Word-context search

This section describes word-context searches of text data (document data).

(1) Overview of word-context search

Word-context search is a function that quickly searches English-language text data (document data) for specific English words. Word-context search allows the following two methods of searching for English words:

- Complete-match retrieval

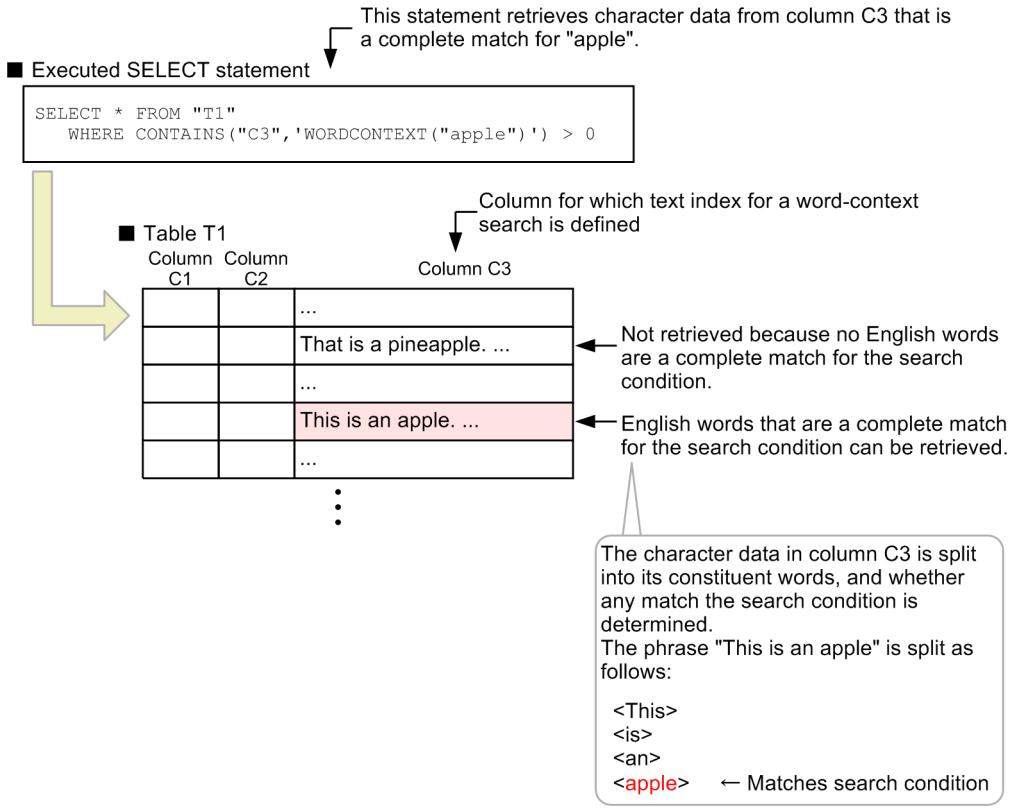
  This method searches for words that exactly match the specified word. For example, if you want to search for the word apple, specify apple as the search term.
- Leading-match search

  In this method, you specify the first few letters of the words you want to retrieve. For example, if you want to search for the word apple, you might specify app as the search term.

Use a word-context search when you want to quickly search English-language text data (document data) for words using complete-match retrieval or leading-match search. The following figure shows an overview of word-context search:

Figure 2-61: Overview of word-context search

● Example of complete-match word-context search



To perform a word-context search, you use the `CONTAINS` scalar function. For details about the `CONTAINS` scalar function, see *CONTAINS* in *Character string functions (Acquisition of character string information)* under *Scalar Functions* in the manual *HADB SQL Reference*.

When performing a word-context search, you can reduce the number of pages to be loaded by defining a text index for a word-context search. By doing so, you can improve table retrieval performance. For details about text indexes for word-context searches, see *CREATE INDEX (define an index)* under *Definition SQL* in the manual *HADB SQL Reference*.

The text data (document data) targeted by a word-context search must satisfy the following conditions:

- The text data consists of single-byte alphanumeric characters.
- Individual words are separated by a delimiting character such as a single-byte space, tab, line feed, or period.

The delimiting character itself is ignored by the word-context search. Text that is not written in English can also be used as target data for a word-context search if it satisfies these conditions. For details about delimiting characters, see (2) Relationship between word-context search and delimiting characters.

Note

Word-context search will work with words that contain full-width characters or full-width spaces. For example, if the search term `Hitach ｉ` is specified (where the letter ｉ is a full-width character), the word-context search will be executed with `Hitach ｉ` as the search term.

Important

Word-context search does not use trailing-match or middle-match retrieval. For example, you cannot find the word `pineapple` by executing a word-context search with `apple` specified.

If you want to perform middle-match or trailing-match retrieval, you can do so by using a `LIKE` predicate to perform middle-match retrieval. Alternatively, you can use the `CONTAINS` scalar function to perform middle-match retrieval.

(2) Relationship between word-context search and delimiting characters

A word-context search retrieves words in English language documents that are separated by delimiting characters. The following characters are handled as delimiting characters in word-context searches:

- Single-byte spaces (`0x20`)
- Tabs (`0x09`)
- Line feeds (`0x0A`)
- Carriage returns (`0x0D`)
- Periods (`0x2E`)
- Question marks (`0x3F`)
- Exclamation marks (`0x21`)

The delimiting characters themselves do not constitute part of the word. This means that specifying one delimiting character or two or more consecutive delimiting characters does not impact the surrounding words.

■ When data contains consecutive delimiting characters

In the following sentences, `This`, `is`, `an`, and `apple` are treated as individual words.

- `This△is◊an△apple.`
- `This△is◊an△△◊apple.`

Legend:

△: Single-byte space

◊: Tab

You specify the characters to handle as delimiting characters when defining a text index for a word-context search. For details, see 5.4.6 Selecting the delimiting characters for word-context searches (DELIMITER).

You can also specify multiple words (phrases) when conducting a word-context search using complete-match retrieval. Delimiting characters do not constitute part of the words in this case either. This means that whether words are separated by one delimiting character or two or more consecutive delimiting characters has no impact on the words retrieved by the search.

■ Searching for multiple words (phrases) by using complete-match retrieval in a word-context search

When you execute a word-context search that specifies `this△△△is`, you can retrieve multiple words (phrases) as follows. Note that the search will not retrieve phrases in which the words are in a different order (such as `is△this`) or when there are additional characters between the words (as in `this△apple△is`).

- `this△is△an△apple`
- `this△◊is◊an◊apple`

Legend:

Δ: Single-byte space

◊: Tab

> **Note**
>
> If the search term consists only of one or more delimiting characters, a full search is performed regardless of the content of the text data (document data) targeted by the word-context search.

(3) Combining word-context search with correction search

Word-context search operations can be performed together with correction search operations. For details about correction search, see 2.17.1 Correction search.

You can specify the scalar function `CONTAINS` to use both word-context search and correction search. Both complete-match and leading-match forms of word-context search operations can be combined with correction search operations.

By combining word-context search and correction search, all occurrences of a word or phrase can be retrieved while ignoring differences like the following:

- Upper and lower case characters
- Characters with and without diacritical marks (such as umlauts)

(4) Combining word-context search with synonym search

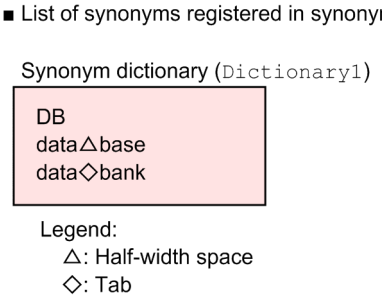
Complete-match word-context search operations can be combined with synonym search operations. For details about synonym search, see 2.17.3 Synonym search.

You can specify the scalar function `CONTAINS` to use both word-context search and synonym search. This means that words targeted by the search and synonyms of the words targeted by the search can be retrieved in one operation by the word-context search.

> **Important**
>
> You cannot combine leading-match word-context search operations with synonym search operations.

Because word-context search searches at the word level, data is retrieved if it matches the search-target word regardless of the type of delimiting character used between words. However, a word retrieved by a synonym search will include a delimiting character if the word is registered in the synonym dictionary with the delimiting character forming part of the word. The following figure shows an example.

Figure 2-62: Search example when combining word-context search with synonym search



Explanation

- When combining word-context search and synonym search, if you search for the term `data◊base`, the search term will not match the synonym `data△base` registered in the synonym dictionary. The synonym registered in the synonym dictionary will not be retrieved.
- When combining word-context search and synonym search, if you search for the term `data△base`, it matches the synonym `data△base` registered in the synonym dictionary. Because this operation searches synonyms registered in the synonym dictionary, the synonym `data◊bank` is also retrieved. Because the search operation also performs word-context search, it will also retrieve `data△bank` which has a different delimiting character.

(5) Combining word-context search, correction search, and synonym search

Complete-match word-context search operations can be combined with correction search and synonym search operations.

2.18 Audit trail facility

The following describes the audit trail facility that outputs the audit trail information required when auditing database usage.

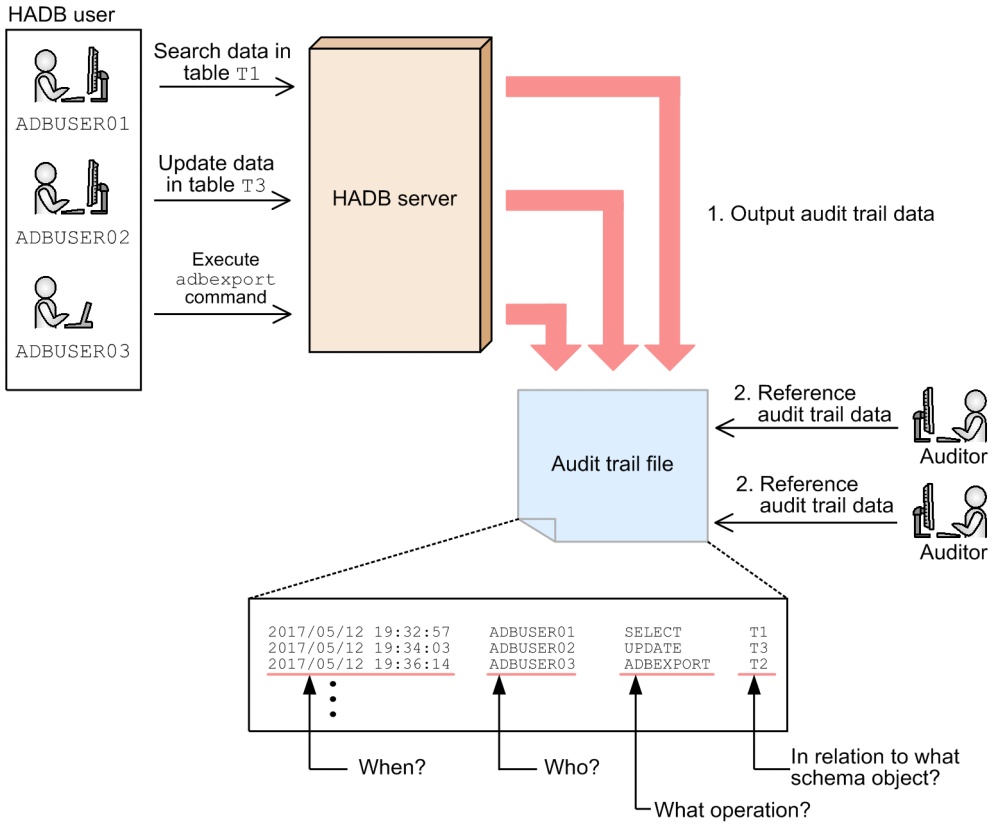
2.18.1 Overview of the audit trail facility

Information about activity by HADB users, such as database access and command execution, is recorded and output to a file called an audit trail file. The facility that outputs this information is called the *audit trail facility*. The records of access and operation output by this facility are called an *audit trail*. For example, when an HADB user accesses a table, information about the operations they perform is output as an operation record (audit trail). This might include the time at which the user accessed the table, their authorization identifier, the operations they performed, and the schema object on which they performed the operations. By viewing the output audit trail, an auditor can find out information such as who accessed what schema object at what time, and what operations they performed. The following are examples of the ways in which audit trails might be used:

- When auditing database usage, audit trails can be used to investigate the operations performed by a specific database user.
- When a security incident occurs, audit trails can be used to investigate the cause, or find out what data might have been divulged.

The following figure provides an overview of the audit trail facility.

Figure 2-63: Overview of audit trail facility



Explanation

1. An operation record is output to an audit trail file when an HADB user performs an operation such as searching or updating a table, or executing a command.
2. When auditing database usage, the person performing the audit (the auditor) can use the audit trail information as part of their investigation. An audit trail can also be used as a resource when looking into the cause of a security incident.

When using a database system, you need to conduct regular audits to ensure that no unauthorized use of the database is taking place, and that the security policy is being followed. Audit trails serve as evidentiary material for auditors conducting such regular audits.

> **Note**
>
> The audit trail facility is not a facility that directly enhances security. Its purpose is to help assess whether the database is being used appropriately and in keeping with the security policy. Use of the audit trail facility does not necessarily prevent database users from engaging in unauthorized activity. However, simply knowing that auditors are using the audit trail facility to monitor database usage could be enough to discourage users with malicious intent. Therefore, the audit trail facility can be seen to have a role in enhancing overall security.

2.18.2 Creating an auditor

To use the audit trail facility, you need to create an auditor. An auditor is an HADB user who has an audit privilege (audit admin privilege or audit viewer privilege). An auditor is responsible for the following:

- Operating the audit trail facility

  The tasks of an auditor who operates the audit trail facility include selecting the operations (audit events) for which to output an audit trail, initiating audit trail output, and revoking the audit privilege of HADB users. To operate the audit trail facility, an HADB user must have *audit admin privilege*.
- Auditing

  The auditor audits database usage and outputs common format audit trail files by referencing the audit trail information.

  To reference audit trail information, an HADB user must have *audit viewer privilege*.

To create an auditor, an HADB user with the `DBA` privilege grants audit privilege to an HADB user by executing the `GRANT` statement. You can create an auditor who has only audit admin privilege, who has only audit viewer privilege, or who has both audit admin privilege and audit viewer privilege. You can also create multiple auditors. Grant the necessary privilege to the HADB user, keeping in mind the role of the auditor and the tasks they are likely to perform.

2.18.3 Operations that trigger audit trail output (audit target events)

An operation for which an audit trail is output is called an *audit target event*. There are two types of audit target event: *mandatory audit events* and *optional audit events*. An audit trail is always output for mandatory audit events if the audit trail facility is enabled. For optional audit events, the auditor can select whether an audit trail is output.

The following are examples of operations that constitute audit target events. An audit trail is output at the completion of these audit target events.

- Examples of mandatory audit events

- Using the adbstart command to start the HADB server
- Using the adbmodarea command to add or modify a DB area
- Using a GRANT statement to grant audit privileges
- Using the adbauditttrail command to disable the audit trail facility
- Using the ADB_AUDITREAD function to reference an audit trail

- Examples of optional audit events
  - Connecting to the HADB server
  - Using a CREATE TABLE statement to define a table
  - Using a SELECT statement to retrieve data from a table
  - Using an UPDATE statement to modify rows
  - Using the adbimport command to import data

For details about the operations that constitute audit target events, see 12.9.1 List of audit target events and output items.

Important

If multiple schema objects are the target of an audit target event, an audit trail is output for each schema object.

Example:

```
SELECT * FROM "T1","T2"
```

Explanation

When the preceding SELECT statement is executed, an audit trail is output in relation to table T1 and another is output in relation to table T2.

In this manner, an audit trail is output for each schema object targeted by the audit target event. If the output of audit trails is triggered as a mandatory audit event and an optional audit event, an extremely large number of audit trails might be output. A large number of audit trails might also impact the effectiveness with which the auditor can perform his or her tasks. For this reason, you must have the objectives of the auditing process clearly in mind when selecting whether to output audit trails for optional audit events. If you decide to output audit trails for optional audit events, you define the audit targets by using the CREATE AUDIT statement.

Note

- If an audit target event applies to just one schema object or there is no schema object targeted by the event, only one audit trail is output.
- An error caused by an incorrectly specified command does not trigger output of an audit trail.

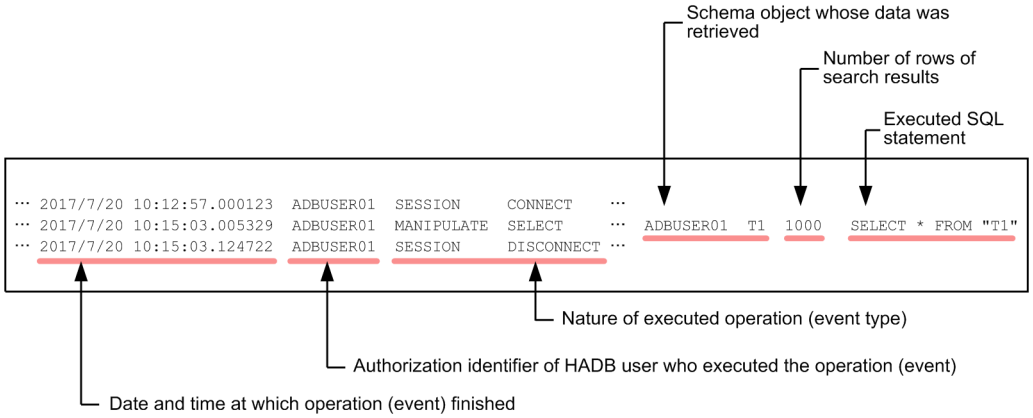
2.18.4 Information output in an audit trail

An audit trail records information such as who performed an operation, when the operation was performed, and the nature of the operation. The following shows an example of the information output in an audit trail:

Example 1:

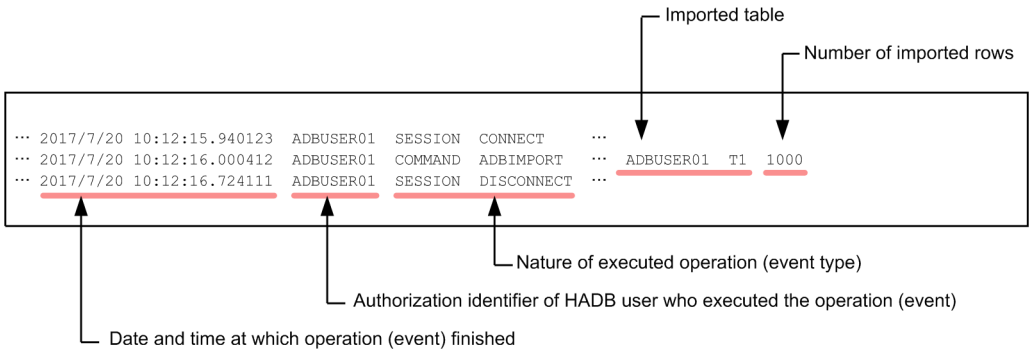
```
SELECT * FROM "T1" WHERE "C1"=100
```

When the preceding SQL statement is executed, audit trail information is output as follows:



Example 2:

The following shows the information output as an audit trail if the `adbimport` command were executed:



These examples show just some of the information output in audit trails. For more details about the information output in audit trails, see 12.9.2 Column structure of table function derived table when retrieving audit trails and 12.9.3 Audit trail output triggers and output items.

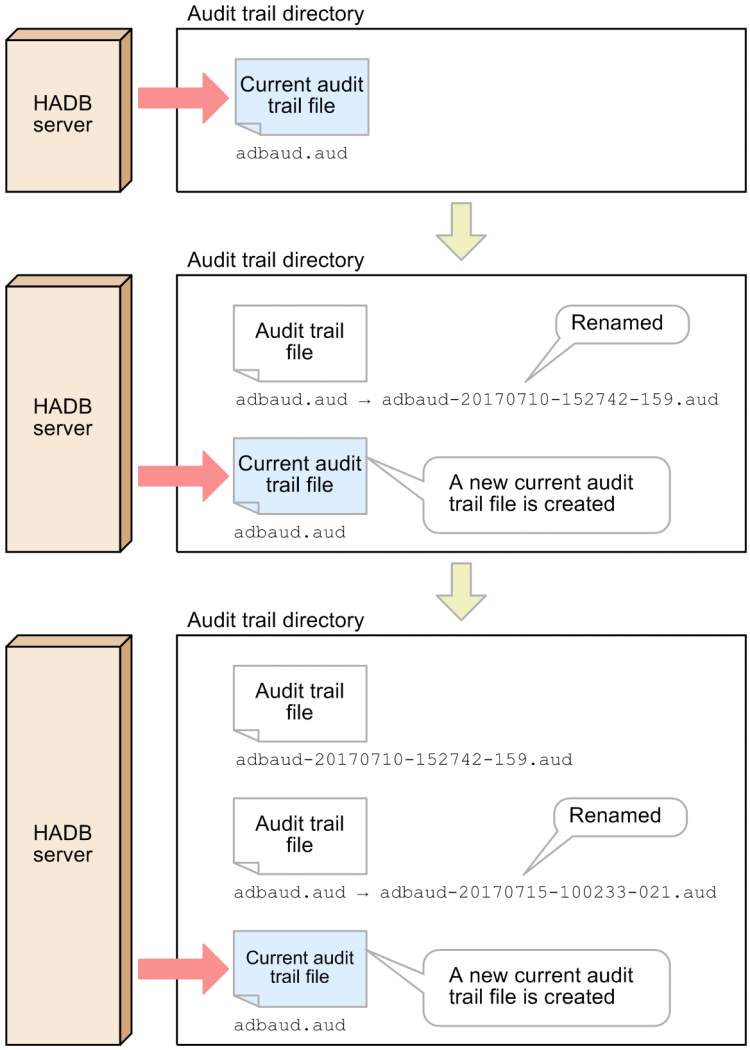
2.18.5 Output destination of audit trails (audit trail file)

Audit trail information is output to a file in a specific directory (the audit trail directory). This file is called an *audit trail file*.

The file that is currently being used as the output destination for audit trail information is called the *current audit trail file*. If the size of the current audit trail file reaches the maximum allowed, the HADB server renames the current audit trail file. It then creates a new current audit trail file, outputting subsequent audit trail information to the new file. This process is called *swapping the current audit trail file*.

The following figure shows the process of swapping the current audit trail file.

Figure 2-64: Process of swapping current audit trail file



Explanation

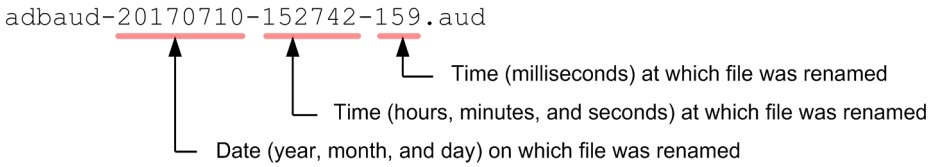
- Audit trail information is output to the current audit trail file in the audit trail directory. The name of the current audit trail file is always `adbaud.aud`.
- When the size of the current audit trail file reaches the maximum (256 MB), the current audit trail file is renamed.

> **Note**
>
> You can change the maximum size for the audit trail file by using the `adb_audit_log_max_size` operand in the server definition.

- The HADB server creates a new current audit trail file, and outputs subsequent audit trail information to the new file.

■ Rules for renaming the audit trail file

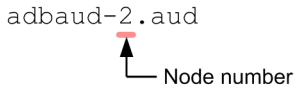
When swapping the current audit trail file, the audit trail file that was the current file is renamed according to the following rules:



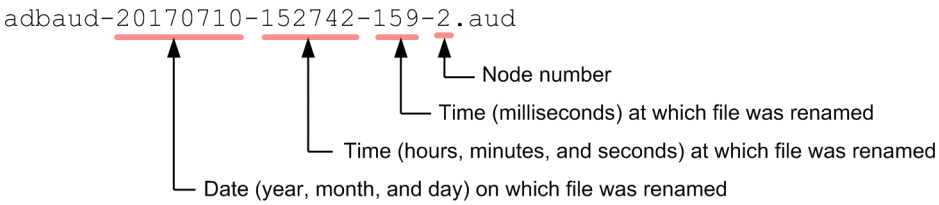
When using the multi-node function

The file name of the current audit trail file and the rules for renaming audit trail file are as follows:

- Naming rules for current audit trail file



- Naming rules for renamed audit trail files



Note

- Audit trail files that do not conform to these naming rules or that are stored in a location other than the audit trail directory are handled as files to which output of audit trail information has completed.
- If an audit trail file with the same name already exists in the audit trail directory, a different name is used to avoid a conflict.

■ Timing of swapping of current audit trail file

The current audit trail file is swapped at the following times:

- When the size of the current audit trail file reaches the maximum
- When the `adbaudittrail --swap` command is used to swap the current audit trail file

Note

If any of the following operations are performed, the current audit trail file is renamed after an audit trail for the operation is output to the file. In this case, a new current audit trail file is not created.

- HADB server termination
- Using the `adbaudittrail --stop` command to disable the audit trail facility

A new current audit trail file is created when the HADB server next starts or when the audit trail facility is re-enabled.

■ Maximum number of audit trail files that can be stored in audit trail directory

You can specify the maximum number of audit trail files that can be stored in the audit trail directory. To specify this maximum number, you use the `adb_audit_log_max_num` operand in the server definition. For example, if you specify `100` for this operand, a maximum of 100 audit trail files can be stored in the audit trail directory. This number includes the current audit trail file. When the HADB server creates a 101st audit trail file, the oldest audit trail file is deleted.

Important

By default, no maximum number of files is set. Because the number of audit trail files in the audit trail directory increases continuously over time, you must take care to ensure the disk does not run out of free space.

2.18.6 Referencing audit trails

Audit trail information is output to the audit trail file in binary format. For this reason, it cannot be opened and referenced directly in the host operating system. Audit trail information can be referenced by using either of the following methods:

- **Executing a `SELECT` statement with the `ADB_AUDITREAD` function specified**

  `ADB_AUDITREAD` is a function that converts the audit trail information in the audit trail file to data in tabular format (as a table function derived table). Audit trail information can be referenced by executing a `SELECT` statement with the `ADB_AUDITREAD` function specified.

- **Using the `adbconvertaudittrailfile` command to convert audit trail information**

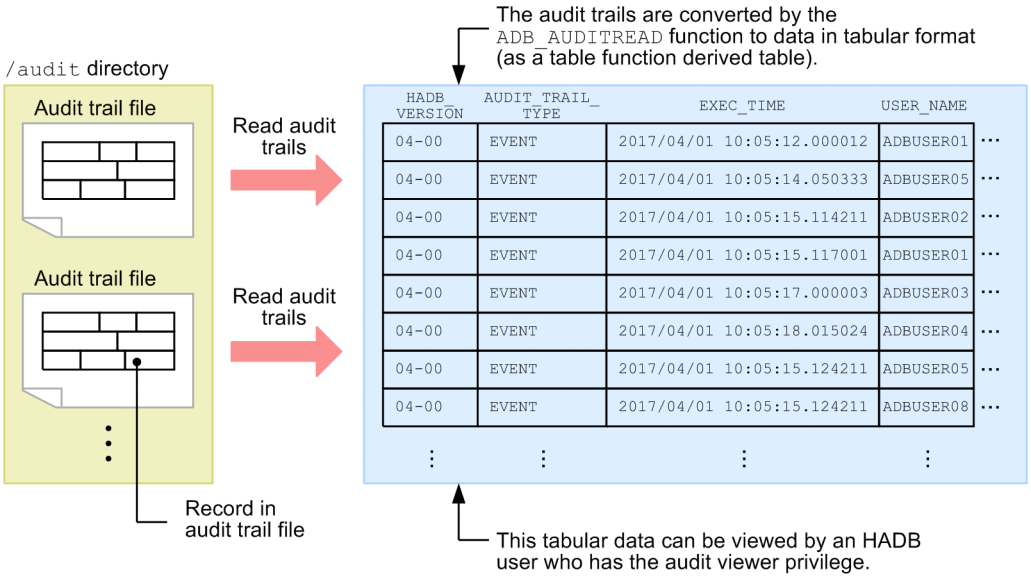
  The audit trail information that has been output to an audit trail file can be converted by using the `adbconvertaudittrailfile` command so that the audit trail information can be referenced with JP1/Audit, which is another product. For details about conversion of audit trail information, see 2.18.9 Conversion of audit trail information (linkage with JP1/Audit Management - Manager).

This subsection describes the audit trail reference method that executes a `SELECT` statement with the `ADB_AUDITREAD` function specified.

Specify in the `ADB_AUDITREAD` function the path of the audit trail file that stores the audit trail information you want to investigate. The HADB server reads the audit trail information from the audit trail file and converts it to data in tabular format.

The following figure shows an example of using the `ADB_AUDITREAD` function to read the audit trail information in an audit trail file.

Figure 2-65: Example of using `ADB_AUDITREAD` function to read audit trail information from audit trail file



Explanation

When the following SELECT statement is executed, the audit trail information in the audit trail file specified in the underlined portion is read and converted to data in tabular format.

```
SELECT * FROM
  TABLE(ADB_AUDITREAD(MULTISET['/audit/*.aud'])) "DT"
```

Important

The path of the audit trail file (/audit/*.aud) is specified in the ADB_AUDITREAD function. In this example, the special character * is specified. Specifying the path in this way means that the ADB_AUDITREAD function reads all audit trail files in the /audit directory.

For details about the ADB_AUDITREAD function, see *ADB_AUDITREAD function* in the manual *HADB SQL Reference*.

The HADB server determines the column name and data type of each column in the tabular data converted by the ADB_AUDITREAD function. For details about the column names and data type of individual columns and the information they contain, see 12.9.2 Column structure of table function derived table when retrieving audit trails.

■ Example of retrieving audit trails

You can search the audit trail information by using a SELECT statement that specifies search conditions such as a specific period of interest. The following shows an example of retrieving specific audit trail information:

Example:

In this example, a list of HADB users who accessed the HADB server between April 1st and April 30th, 2017 is retrieved.

```
SELECT DISTINCT "USER_NAME"                                              ...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/audit/*.aud'])) "DT"
    WHERE "EXEC_TIME" BETWEEN TIMESTAMP'2017/04/01 00:00:00.000000'      ...2
                          AND TIMESTAMP'2017/04/30 23:59:59.999999'
```

Example search results

```
USER_NAME
---------
ADBUSER02
ADBUSER06
ADBUSER07
```

Explanation

1. The USER_NAME column of the tabular data converted by the ADB_AUDITREAD function contains the authorization identifiers of the HADB users who accessed the HADB server.
2. The EXEC_TIME column contains the time at which the HADB user performed the operation (the event completion time).

Important

Only HADB users with the audit viewer privilege can reference audit trail information. That is, only these users can execute SELECT statements that specify the ADB_AUDITREAD function.

2.18.7 Enabling and disabling the audit trail facility

Because the output of audit trail information significantly impacts the performance of the HADB server, the system is configured to not output audit trail information by default. In this state, the audit trail facility is said to be disabled.

After you have completed environment setup for the audit trail facility, executing the `adbauditttrail --start` command enables the facility. Audit trail information will be output from this point. When the audit trail facility is enabled, the following operations can be performed in relation to the facility:

- Using the `ADB_AUDITREAD` function to reference an audit trail
- Use `CREATE AUDIT` statements to define audit targets
- Use `DROP AUDIT` statements to delete audit target definitions
- Use `REVOKE` statements to revoke audit privileges

To disable the enabled audit trail facility, you execute the `adbauditttrail --stop` command. Disabling the audit trail facility stops the output of audit trail information.

> **Important**
>
> Only an HADB user with audit admin privilege can enable and disable the audit trail facility.

2.18.8 Preparing the disks used by the audit trail facility

Depending on the number of audit target events that occur, an extremely large number of audit trails might be output. This can result in a large number of I/O operations with respect to the disk to which the audit trails are output, potentially affecting the processing performance of the HADB server. For this reason, we recommend that you prepare a disk (audit trail output disk) to serve as a dedicated output destination for audit trail information. We also recommend that you prepare additional disks for storing audit trail files (an audit trail storage disk and an audit trail long-term storage disk). We recommend that you have these three disks in place when using the audit trail facility. The purpose of each disk is as follows.

- Audit trail output disk

  The only directory created on the audit trail output disk is the audit trail directory. The HADB server outputs audit trail information to this directory. Generally, audit trail files stored on this disk will not be referenced.

  Audit trail files are to be regularly moved from the audit trail output disk to the audit trail storage disk. We recommend that you create a batch program that moves the files, and run the program regularly.

- Audit trail storage disk

  The audit trail storage disk stores the audit trail files used for auditing purposes. During auditing, the auditor references audit trail information by specifying the path of the audit trail file on the audit trail storage disk in the `ADB_AUDITREAD` function. Suppose an auditor will be using one year of audit trail information for auditing purposes. In this case, you would store one year of audit trail files on this disk.

  Files older than the auditing period are moved to the audit trail long-term storage disk.

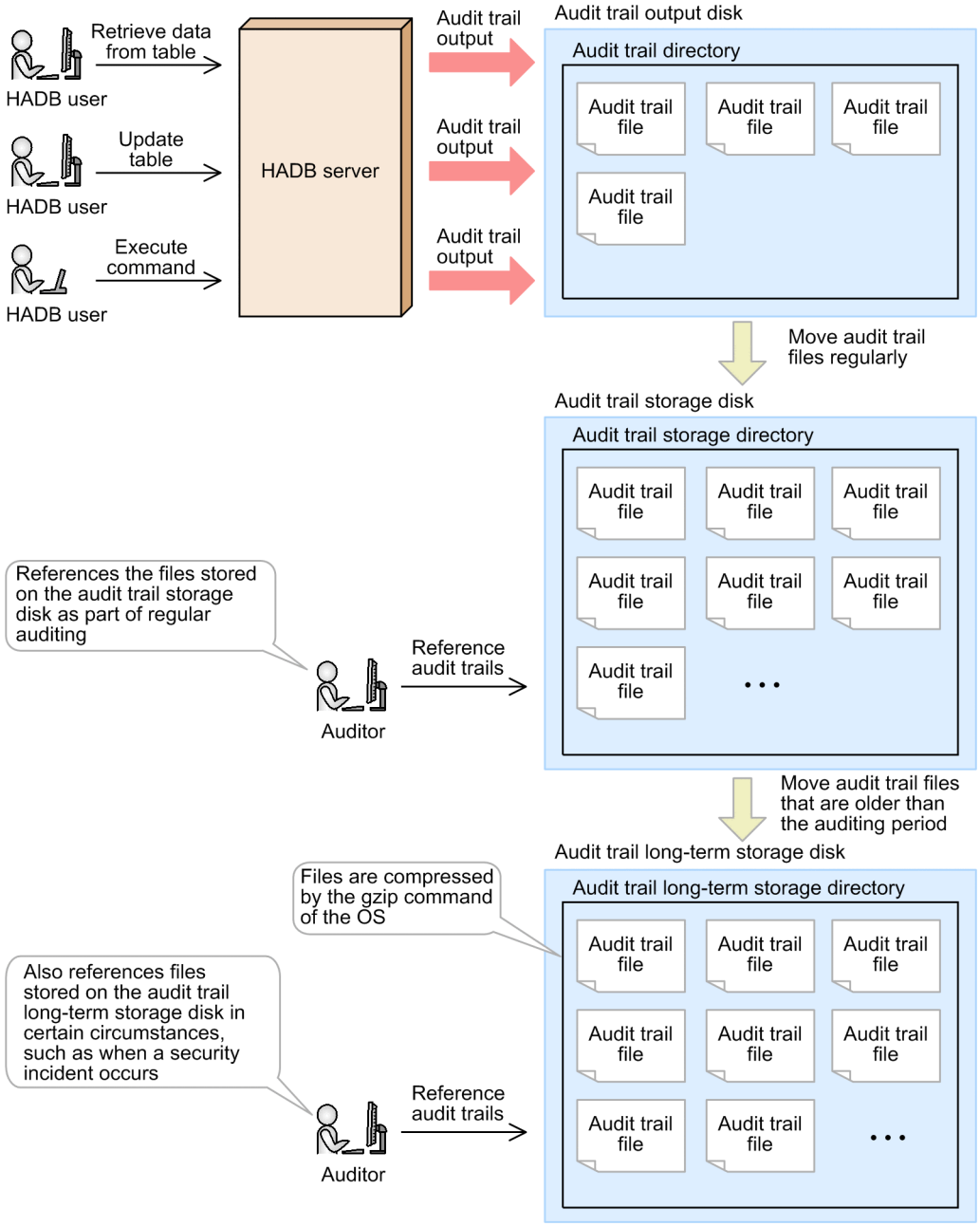
- Audit trail long-term storage disk

  The audit trail long-term storage disk stores the audit trail files from time periods that are no longer subject to auditing. For example, if an audit is conducted each year, audit trail files that are more than one year old are stored on the audit trail long-term storage disk.

Ordinarily, there is no reason to reference the audit trail information in the audit trail files stored on the audit trail long-term storage disk. The audit trail files on this disk might be used if a security incident occurs, or the need arises to audit a time period further in the past.

The following figure shows a recommended approach to operating the audit trail facility.

Figure 2-66: Flow of recommended audit trail facility operation



In the preceding figure, because the audit trail files stored on the audit trail long-term storage disk are accessed infrequently, we recommend that you compress them using the `gzip` command provided by the OS. Consider the size of the disk when deciding whether to compress the files. Because the audit trail files stored on the audit trail storage disk are referenced during auditing, we recommend that these files are stored uncompressed.

Important

A SELECT statement with the ADB_AUDITREAD function specified can reference audit trail information even if the audit trail file that contains the information is compressed.

2.18.9 Conversion of audit trail information (linkage with JP1/Audit Management - Manager)

This subsection describes conversion of audit trail information. To use JP1/Audit Management - Manager (JP1/Audit) to collect and manage the audit trail information about the HADB server, the audit trail information must be converted.

The adbconvertaudittrailfile command can be used to convert the audit trail information in an audit trail file into a common format that can be handled by JP1/Audit. This conversion is called *conversion of audit trail information*.

The converted audit trail information is output to a file in the directory that is specified when the adbconvertaudittrailfile command is executed (the output-directory for common format audit trails). This file is called a *common format audit trail file*. JP1/Audit can collect and manage the audit trail information in common format audit trail files.

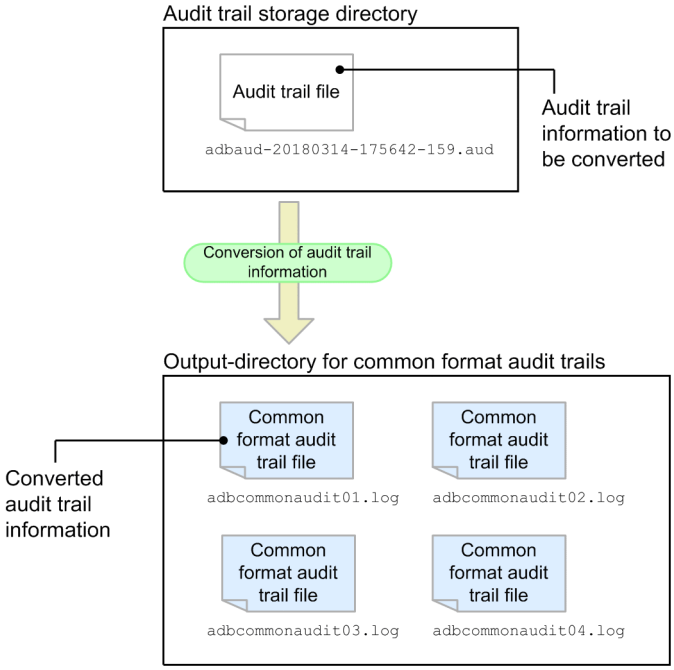
As explained in 2.18.6 Referencing audit trails, audit trail information is output in binary format to audit trail files. Because JP1/Audit cannot handle audit trail information in this format, it is impossible to collect or manage audit trail information even if the HADB server is linked with JP1/Audit. Therefore, to collect and manage audit trail information by using JP1/Audit, it is necessary to convert the audit trail information by using the adbconvertaudittrailfile command.

Note

If JP1/Audit can collect and manage audit trail information of the HADB server, it becomes possible to centrally manage the audit trail information of HADB and other products.

The following figure shows an overview of conversion of audit trail information.

Figure 2-67: Overview of conversion of audit trail information



Explanation

To convert audit trail information by using the `adbconvertauditrailfile` command, specify *an audit trail file that contains audit trail information to be converted* and *the output-directory for common format audit trails* in the command.

The audit trail information converted by the `adbconvertauditrailfile` command is output to common format audit trail files that are stored in the output-directory for common format audit trails.

If JP1/Audit is linked with the HADB server, JP1/Audit can collect and manage the converted audit trail information that has been output to common format audit trail files.

For details about how to link the audit trail facility with JP1/Audit, see 12.8 Linkage between the audit trail facility and JP1/Audit Management - Manager.

For details about the `adbconvertauditrailfile` command, see *adbconvertauditrailfile (Convert the Audit Trail File)* in the manual *HADB Command Reference*.

Important

Only HADB users having the audit viewer privilege can execute the `adbconvertauditrailfile` command.

Note

The output-directory for common format audit trails can contain a maximum of four common format audit trail files. For details, see (1) Environment settings for the HADB server in 12.8.3 Environment settings for linking the audit trail facility with JP1/Audit.

2.19 Multi-node function

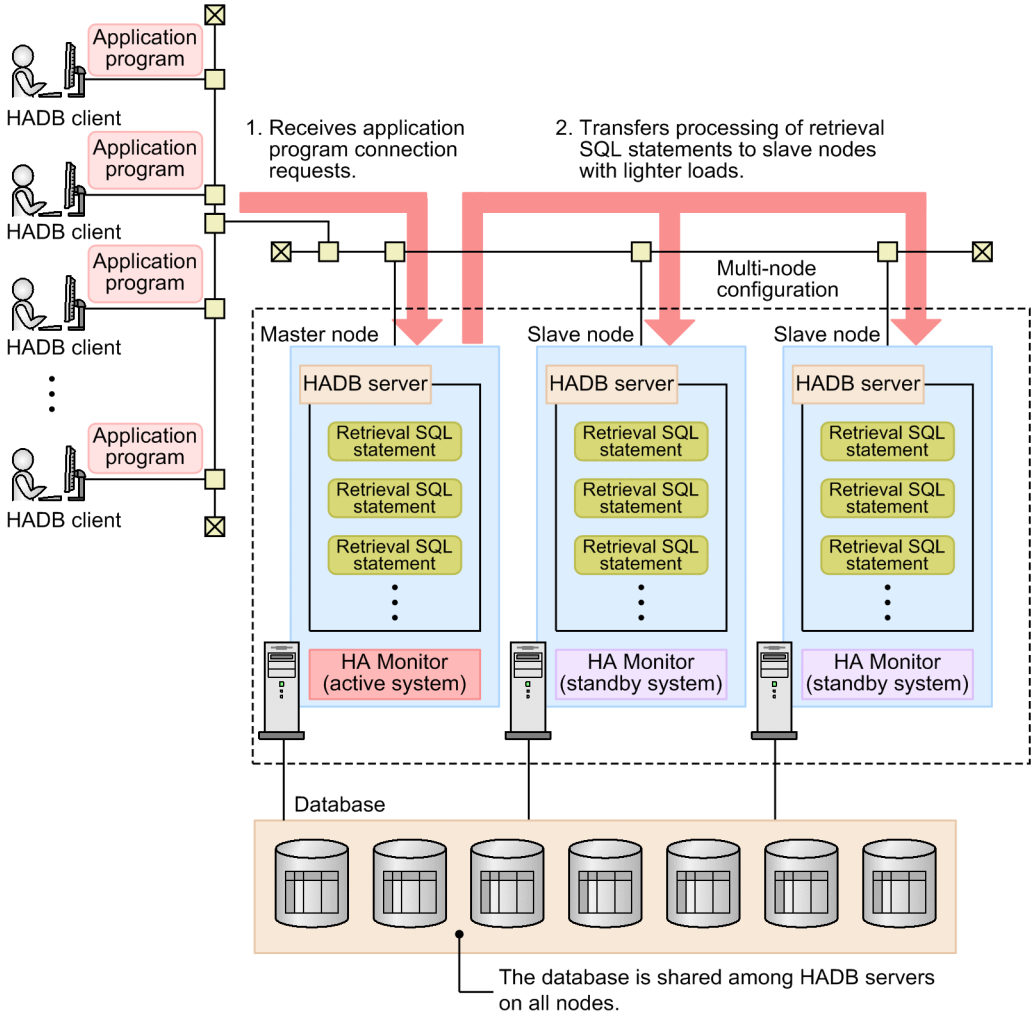
This section explains the features of the multi-node function and the system configuration necessary for using it.

2.19.1 About the multi-node function

The multi-node function realizes the load distribution of retrieval SQL statement processing by coordinating multiple HADB servers (scale-out based load distribution).

The following figure shows an example of using the multi-node function to distribute the processing load.

Figure 2-68: Example of using the multi-node function to distribute the processing load



Explanation

In the example above, three HADB servers are used to execute the retrieval SQL statement, enabling load distribution.

1. Application program connection requests from HADB clients are received by the HADB server on the master node.
2. The HADB server on the master node controls load distribution. It does this by selecting a node with a lighter load, and executing retrieval SQL statements on that node. Load distribution is achieved by allocating processing of retrieval SQL statements to the individual nodes.

- **Master node**

  The node containing the HADB server that controls the multi-node function is called the *master node*. Update SQL statements and commands such as the `adbimport` command are processed on the master node.

  One master node is required.

  Application program or command connection requests are received by the HADB server on the master node.

- **Slave node**

  A node containing an HADB server that processes retrieval SQL statements is called a *slave node*. When the multi-node function is used, the maximum number of nodes is four. Therefore, you can set up between one and three slave nodes.

  Update SQL statements and some commands, such as the `adbimport` command, cannot be executed on a slave node.

- **Multi-node configuration**

  A system configuration consisting of multiple HADB servers that are using the multi-node function is called a *multi-node configuration*.

- **HA Monitor**

  This program product monitors node statuses and separates any node on which a failure occurs. For details about the functionality of HA Monitor, see 2.19.3 Managing nodes with HA Monitor.

  The master node operates as the active system of HA Monitor, and the slave nodes operate as the standby systems of HA Monitor.

> **Note**
>
> A server machine on which an HADB server is installed, including the operating system, HADB server, and HA Monitor installed on that server machine, is referred to as a *node*.

Processing performance when the multi-node function is used depends on the hardware environment and the amount of data being processed. Note that using the multi-node function might increase the time required to execute SQL statements and commands because the overhead for communication or synchronization between the master and slave nodes increases.

2.19.2 Nodes on which transactions and commands are executed

(1) Node on which a transaction is executed

The node on which a transaction is executed is determined by the specifications of the transaction access mode and transaction isolation level. A transaction that meets all of the following conditions is executed on a slave node:

- The transaction access mode is read-only mode.
- The transaction isolation level is `READ COMMITTED`.

However, depending on the situation such as the number of idle threads on the slave nodes, the transaction might be executed on the master node even when the preceding conditions are met. The HADB server on the master node selects the least loaded node, and the transaction is executed on that node.

Important

If you execute an application program that performs only search, do so in the following manners in order to effectively use the resources of the slave nodes:

- Set the transaction access mode to read-only mode.
- Set the transaction isolation level to `READ COMMITTED`.

For details about transaction access modes, see 2.9.5 Transaction access modes.

For details about transaction isolation levels, see 2.9.2 Transaction isolation levels supported by HADB.

Note that while a transaction in read/write mode is being executed on the master node, all transactions are executed on the master node (no transactions are executed on slave nodes). Transactions whose transaction access mode is read-only mode and transaction isolation level is `READ COMMITTED` are not exceptions. Therefore, to effectively use the resources of the slave nodes, make sure that a transaction in read/write mode is committed as frequently as possible.

Important

If the multi-node function is used, a transaction that meets all of the following conditions automatically terminates normally when the cursor is closed:

- The transaction access mode is read-only mode.
- The transaction isolation level is `READ COMMITTED`.

If cursors are concurrently opened with multiple statement handles, the transaction automatically terminates normally when all open cursors are closed.

(2) Nodes on which SQL statements are executed

The following table shows the nodes on which SQL statements are executed.

Table 2-22: Nodes on which SQL statements are executed

No.	SQL statement type			Node on which SQL statements are executed
1	Definition SQL statements			Master node
2	Data manipulation SQL statements	Update SQL statements	`INSERT` statement	Master node
3			`UPDATE` statement	
4			`DELETE` statement	
5			`TRUNCATE TABLE` statement	
6			`PURGE CHUNK` statement	
7		Retrieval SQL statements	`SELECT` statement	Master node or slave node

■ Notes on executing a definition SQL statement, the TRUNCATE TABLE statement, or the PURGE CHUNK statement

While transactions are being executed on slave nodes, a definition SQL statement, the `TRUNCATE TABLE` statement, or the `PURGE CHUNK` statement is not executed immediately (and is placed in WAIT status). The

definition SQL statement, TRUNCATE TABLE statement, or PURGE CHUNK statement is executed after all transactions running on the slave nodes terminate.

Also, while a definition SQL statement, the TRUNCATE TABLE statement, or the PURGE CHUNK statement is being executed, all transactions are executed on the master node (no transactions are executed on slave nodes).

▪ Notes on executing the INSERT, UPDATE, or DELETE statement

If either of the following conditions is met, the INSERT, UPDATE, or DELETE statement is not executed immediately (and is placed in WAIT status):

- A transaction that accesses the DB area that contains the table to be processed by the INSERT, UPDATE, or DELETE statement is being executed on a slave node.
- A transaction that accesses the DB area that contains the indexes of the table to be processed by the INSERT, UPDATE, or DELETE statement is being executed on a slave node.

After the transaction that meets either of the preceding conditions terminates, the INSERT, UPDATE, or DELETE statement is executed.

When the INSERT, UPDATE, or DELETE statement is executed, the processing of an SQL statement that is running on a slave node and meets either of the following conditions is switched to the master node:

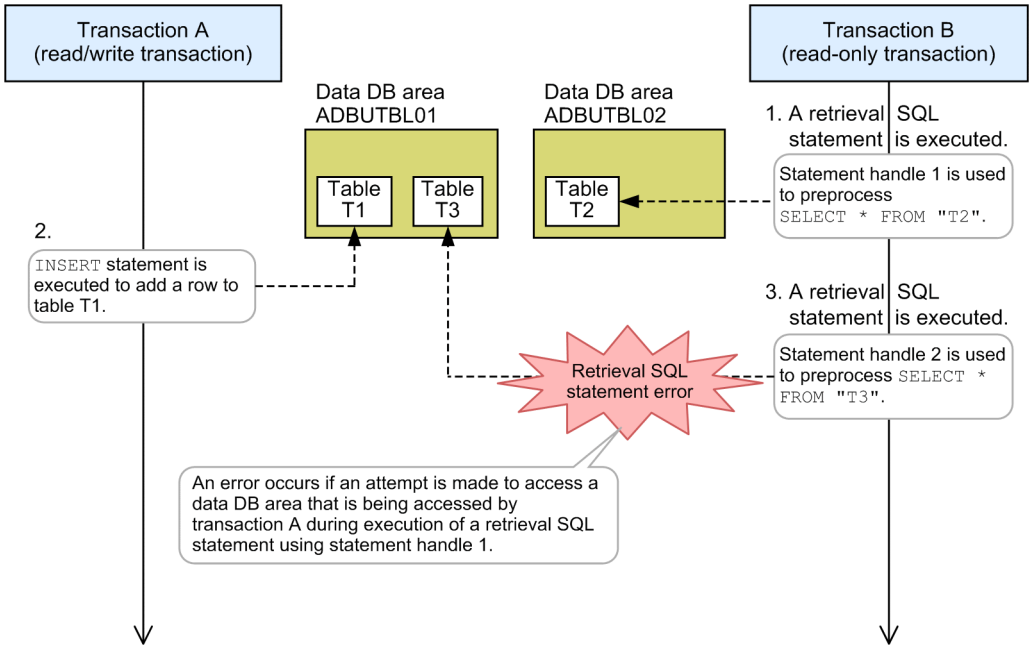
- An SQL statement that accesses the DB area that contains the table to be processed by the INSERT, UPDATE, or DELETE statement
- An SQL statement that accesses the DB area that contains the indexes of the table to be processed by the INSERT, UPDATE, or DELETE statement

▪ Cases in which retrieval SQL statements result in an error

If the multi-node function is used, retrieval SQL statements that meet all of the following conditions cannot be executed:

1. An attempt is made to concurrently perform retrieval SQL statements by using multiple statement handles in the same transaction.
2. One of the retrieval SQL statements is accessing the data DB area.
3. A retrieval SQL statement (that is not the one in condition 2) attempts to access the data DB area that is being accessed by an update SQL statement (the retrieval SQL statement results in an error).

Example:



Explanation

Assume that Transaction A, which is a read/write transaction, and Transaction B, which is a read-only transaction, are being executed concurrently.

In Transaction B, an attempt is made to simultaneously perform retrieval SQL statements by using two statement handles (1 and 3 in the figure).

1. The preprocessing of the retrieval SQL statement for table `T2` is being executed by using the statement handle 1.
2. In Transaction A, the `INSERT` statement is being executed for table `T1`.
3. After the `INSERT` statement in 2 is executed (Transaction A is not committed), Transaction B uses statement handle 2 (that is not statement handle 1) to execute the preprocessing of the retrieval SQL statement for table `T3`.

When the processing in 1 and the processing in 3 are executed concurrently, the retrieval SQL statement in 3 results in an error. At this time, the `KFAA31898-E` message is output. Take action as instructed in the `KFAA31898-E` message.

(3) Nodes on which commands can be executed

The nodes on which commands can be executed differ depending on the commands. For details about the nodes on which different commands can be executed, see *Nodes on which commands can be executed when the multi-node function is used* in the manual *HADB Command Reference*.

(4) Restrictions on simultaneously executing commands with transactions

There are some commands that cannot be simultaneously executed with transactions. The following table shows whether the command can be executed simultaneously with transactions.

Table 2-23: Whether the command can be executed simultaneously with transactions

No.	Command name		Whether simultaneous execution is possible
1	`adbarchivechunk`		B
2	`adbchgchunkcomment`		B
3	`adbchgchunkstatus`		B
4	`adbgetcst`		B
5	`adbidxrebuild`		B
6	`adbimport`	background import	A
7		Other types of import	B
8	`adbinit`		C
9	`adbmergechunk`		B
10	`adbmodarea`		C
11	`adbreorgsystemdata`		B
12	`adbsql`		A[#]
13	`adbunarchivechunk`		B

No.	Command name	Whether simultaneous execution is possible
14	Other commands	A

Legend:

A: Command that can be executed simultaneously with transactions

B: Command that can be executed simultaneously with transactions conditionally. For details, see **▪ Commands that can be executed simultaneously with transactions conditionally** described later in this manual.

C: Command that cannot be executed simultaneously with transactions

#

If `READ_ONLY` is specified for the `adb_clt_trn_access_mode` operand in the client definition, the command can be executed simultaneously with transactions that are being executed on slave nodes.

▪ Commands that can be executed simultaneously with transactions conditionally

For commands with B in the Whether simultaneous execution is possible column in Table 2-23: Whether the command can be executed simultaneously with transactions, the following restrictions apply:

- While a B-type command is being executed, all transactions are executed on the master node (no transactions are executed on slave nodes). Transactions whose transaction access mode is read-only mode and transaction isolation level is `READ COMMITTED` are not exceptions.
- While transactions are being executed on slave nodes, a B-type command is not executed immediately (and is placed in WAIT status). The B-type command is executed after all transactions that are being executed on slave nodes terminate.

2.19.3 Managing nodes with HA Monitor

HA Monitor is required to use the multi-node function. HA Monitor is a program product that monitors node statuses and separates any node on which a failure occurs.

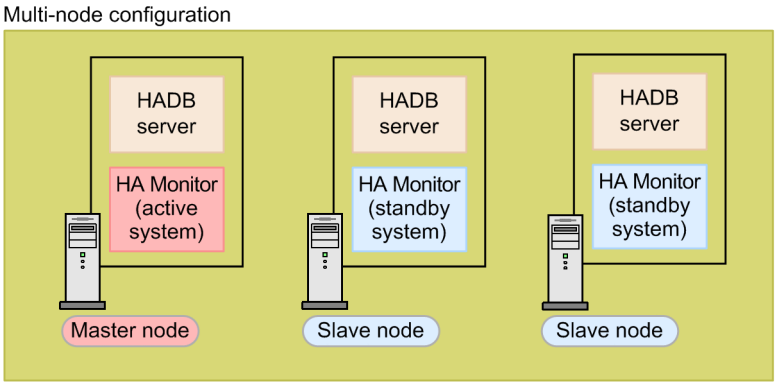
This subsection explains the roles (functions) of HA Monitor in the multi-node function. For details about HA Monitor, see the manual *HA Monitor for Linux(R) (x86)*.

(1) Determining node types

HA Monitor determines which node is to be used as the master node, based on the specification in the `servers` file for defining the HA Monitor environment settings. The specification of the `initial` operand for each node in the `servers` file governs the determination. If `online` is specified for that operand, the node becomes the master node (active system of HA Monitor). If `standby` is specified for that operand, the node becomes a slave node (standby system of HA Monitor).

The following figure shows the relationship among the master node, slave nodes, and the active system and standby system of HA Monitor.

Figure 2-69: Relationship among the master node, slave nodes, and the active system and standby system of HA Monitor



When the HADB servers in a multi-node configuration are started, the master node and slave nodes are determined according to the specification in the `initial` operand.

> **Note**
>
> To start HADB servers in a multi-node configuration, execute the `adbstart` command, and then execute HA Monitor's `monbegin` command. During this process, you must execute the `monbegin` command before processing of the `adbstart` command is completed.

(2) Monitoring for failures

HA Monitor monitors for the failures described in the table below. When any of the following failures is detected, the node on which the failure occurred is separated from the multi-node configuration.

Table 2-24: Failures monitored by HA Monitor

No.	Failures monitored by HA Monitor
1	Hardware failure
2	Hardware power failure
3	Kernel failure
4	HA Monitor abnormal termination
5	HADB server abnormal termination (server process abnormal termination)

The failures listed in the above table are called *node failures*.

■ Key points when reading the manual *HA Monitor for Linux(R) (x86)*

The failures monitored by HA Monitor are host failures (failures 1 through 4 in the above table) and server failures (failure 5 in the above table). Since the multi-node function works according to HA Monitor's monitor mode, you must create a server-monitoring command if you want to detect server failures. For details about how to create a server-monitoring command, see (e) Creating a server-monitoring command in (4) Creating environment variable definitions for commands, and creating commands under 16.3.4 Setting up an HA Monitor environment.

(3) Separating a node when a failure occurs

If a node failure occurs as described in (2) Monitoring for failures, HA Monitor functionality is used to separate the node on which the failure occurred from the multi-node configuration. This allows operations to continue by using the remaining nodes.

The node separation performed in the event of a node failure is managed by the HA Monitor. For details, see 2.19.4 Separating a node from a multi-node configuration.

2.19.4 Separating a node from a multi-node configuration

If a node failure occurs, the node on which the failure occurred is separated from the multi-node configuration so as to allow operations to continue by using the remaining nodes.

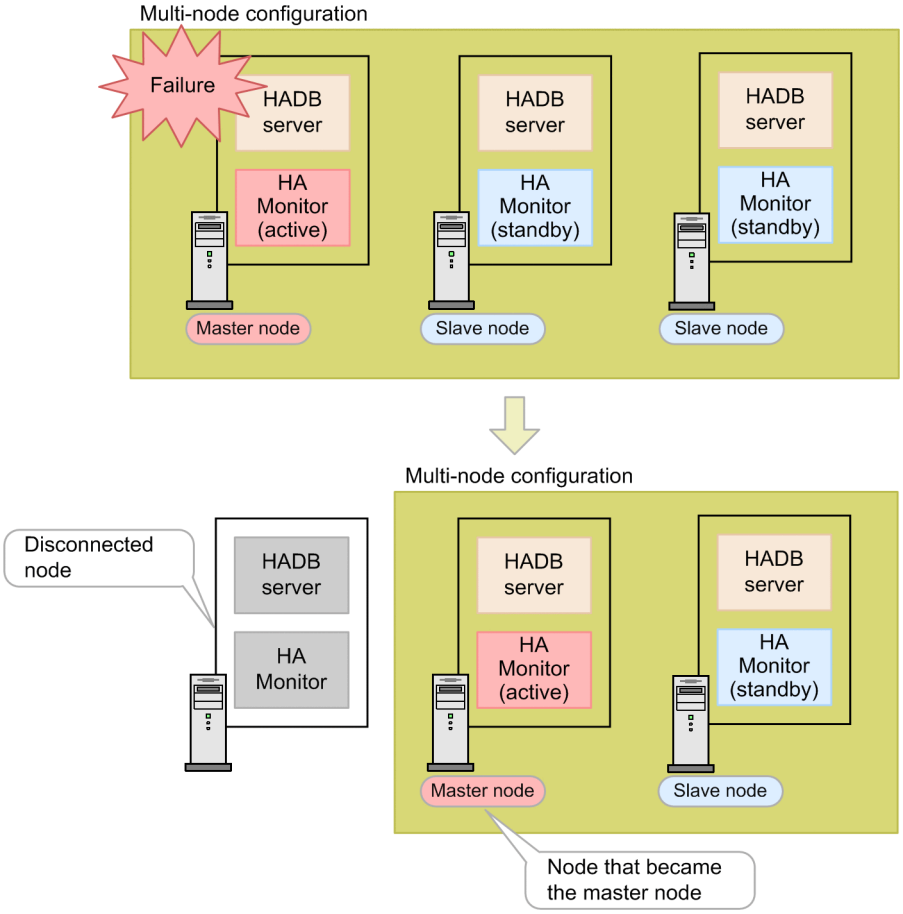
> **Note**
>
> HA Monitor's reduced operation mode refers to separating the node on which a failure occurred from the multi-node configuration so as to continue processing by using the remaining nodes.

(1) When a node failure occurs on the master node

If a node failure occurs on the master node, the master node is separated from the multi-node configuration. Then, one of the slave nodes becomes the master node, and processing continues with the remaining nodes.

The following figure shows the process of separating a node if a node failure occurs on the master node.

Figure 2-70: Process of separating a node if a node failure occurs on the master node



Explanation

- The master node on which a node failure occurred is separated from the multi-node configuration, and one of the slave nodes is switched over to the master node. This is called master node switchover.
- The SQL statements that were being executed on the master node on which a node failure occurred terminate in an error, and the application programs that were running transactions are disconnected from the HADB server. The application programs that were connected to the HADB server and were not running any transactions when the node failure occurred are also disconnected from the server.

Note

On the slave node that will become the new master node, the update processing that was running on the current master node, on which a failure occurred, is restored (rolled back). After restoration finishes, master node switchover takes place.

HA Monitor manages node separation and master node switchover (by using the hot standby function) in the event of a node failure. The master node operates as the active system of HA Monitor, and the slave nodes operate as the standby systems of HA Monitor.

When switching one of the slave nodes to the master node, HA Monitor also switches the shared disk, and adds and deletes the alias IP address.

Note

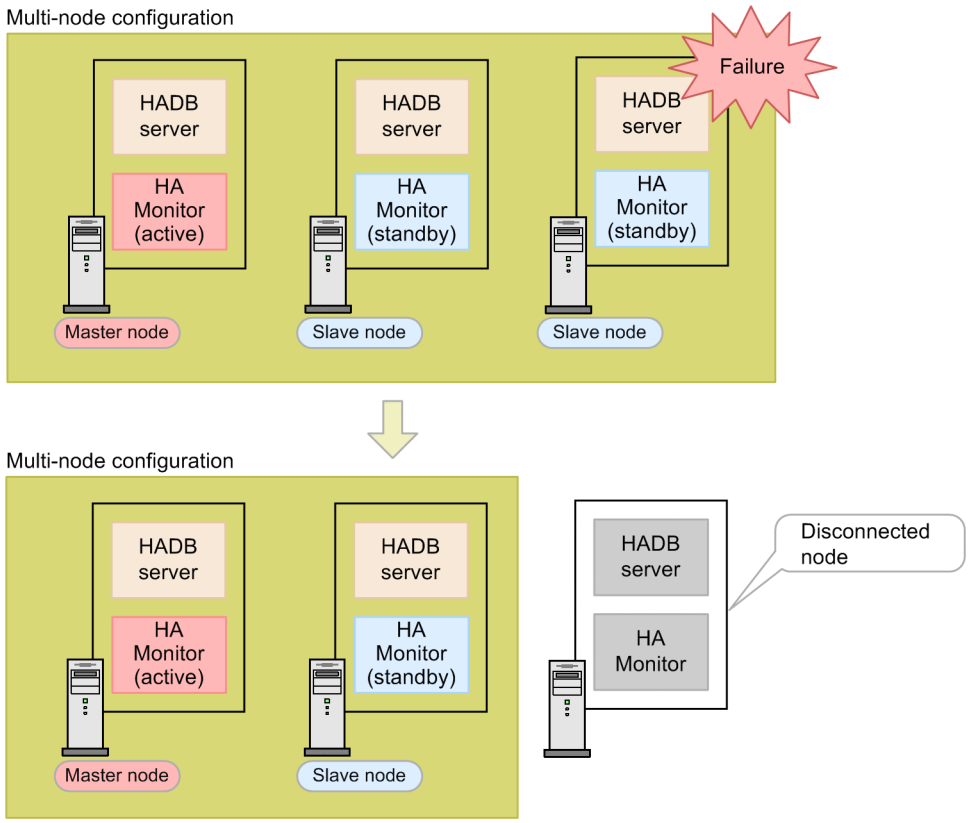
The slave node that the hot standby function of HA Monitor assigns as the new active system becomes the new master node. Which slave node becomes the new master node depends on the specifications of HA Monitor.

(2) When a node failure occurs on a slave node

If a node failure occurs on a slave node, the node is separated from the multi-node configuration, and processing continues with the remaining nodes.

The following figure shows the process of separating a slave node on which a node failure occurred.

Figure 2-71: Process of separating a slave node on which a node failure occurred



Explanation

The SQL statements that were being executed on the slave node on which a node failure occurred terminate in an error, and the application programs that were performing transactions are disconnected from the HADB server.

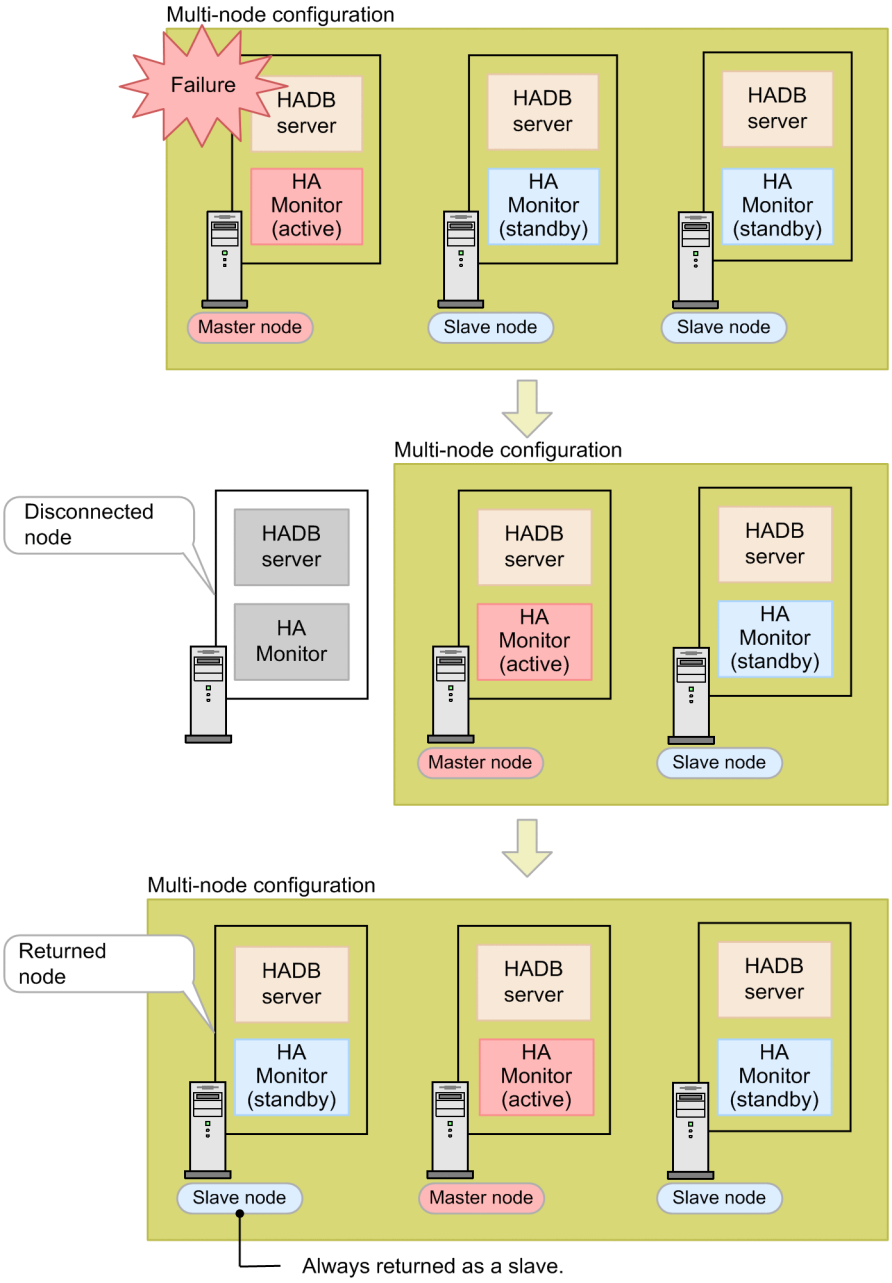
Note

The transactions that were being executed on the failed slave node are rolled back on the master node.

2.19.5 Returning a node to the multi-node configuration

A node that is separated from the multi-node configuration due to a node failure can be returned to the multi-node configuration after the failure has been corrected. At this time, it is not necessary to stop the HADB server in the multi-node configuration. The following figure shows an example of the case in which a node is returned to the multi-node configuration.

Figure 2-72: Example of when a node is returned to the multi-node configuration



Explanation

When a node is returned to a multi-node configuration, its node type is always slave node.

For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

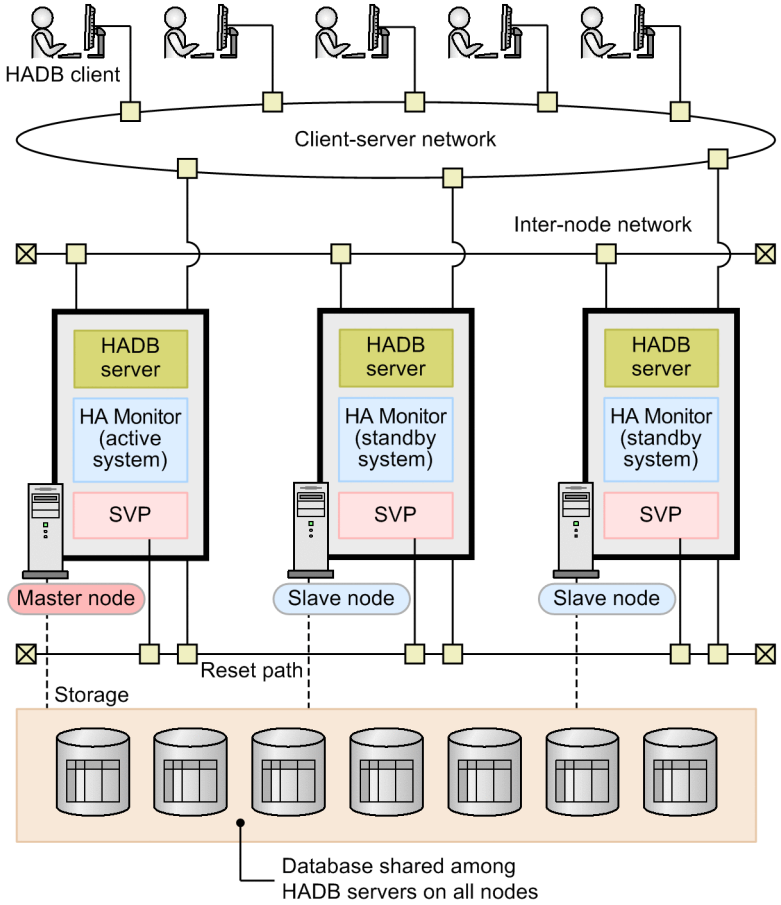
Note

Nodes can be manually disconnected from the multi-node configuration by executing an HA Monitor command for maintenance of the server machine. After maintenance of the server machine finishes, the nodes can be returned to the multi-node configuration.

2.19.6 System configuration example that uses the multi-node function

The following figure shows a system configuration example that uses the multi-node function.

Figure 2-73: System configuration example that uses the multi-node function



Explanation

- Program products required on each node
  You must install HADB server and HA Monitor on each node.
- Required network
  The multi-node function uses the three networks described below. These networks must be physically separated from one another.
  - Client-server network
    This network is used for communication between HADB clients and HADB servers.
  - Inter-node network

This network is used for communication between HADB servers, and as HA Monitor's monitoring path.

- Reset path

  When HA Monitor detects a failure, this network is used for resetting the input/output path of the node on which the failure occurred. When using the host reset function of HA Monitor, a reset path is required.

- Hardware required on each node

  When using the host reset function, each node requires an SVP. An SVP is a failure management processor. A failure management processor is hardware that can set up system configurations and control the CPU.

For details about the monitoring path, reset path, SVPs, and failure management processors, see *Required hardware* in the manual *HA Monitor for Linux(R) (x86)*.

Important

You must select one of the following as the shared disk data protection method used by HA Monitor:

- Host reset
- SCSI reservation for shared disk

We recommend that you select host reset. When using host reset, an SVP must be installed in the server machines on which each node operates.

When using the SCSI reservation for shared disk method, neither an SVP nor a reset path are required. If SVPs are not installed in the server machines of each node, select the SCSI reservation for shared disk method instead.

For details about host reset and SCSI reservation for shared disk, see *Host reset* and *SCSI reservation for shared disk* in the manual *HA Monitor for Linux(R) (x86)*.

2.20 Memory structure of the HADB server

This section explains the structure of the memory used by the HADB server and the relationship to the server definition operands.

2.20.1 Memory used by the HADB server

The HADB server uses shared memory and process memory. The following bullet points explain the memory used by the HADB server.

■ **Shared memory**

The following types of memory are allocated in shared memory:

- **Process common memory**

  Process common memory stores the information needed for starting and controlling the HADB server processes.

- **Memory for managing shared memory**

  Memory for managing shared memory stores management information for the shared memory that is used by the HADB server.

- **Memory for global buffers**

  Memory for global buffers stores the following buffers:

  • Global buffers used for database input/output processing

  • Global buffers for global work tables that are used for global work table input/output processing

  > **Note**
  >
  > When multiple processing real threads are used to process one SQL statement, a global work table is created so that the data in the work table can be shared among those processing real threads.

- **Real thread private memory**

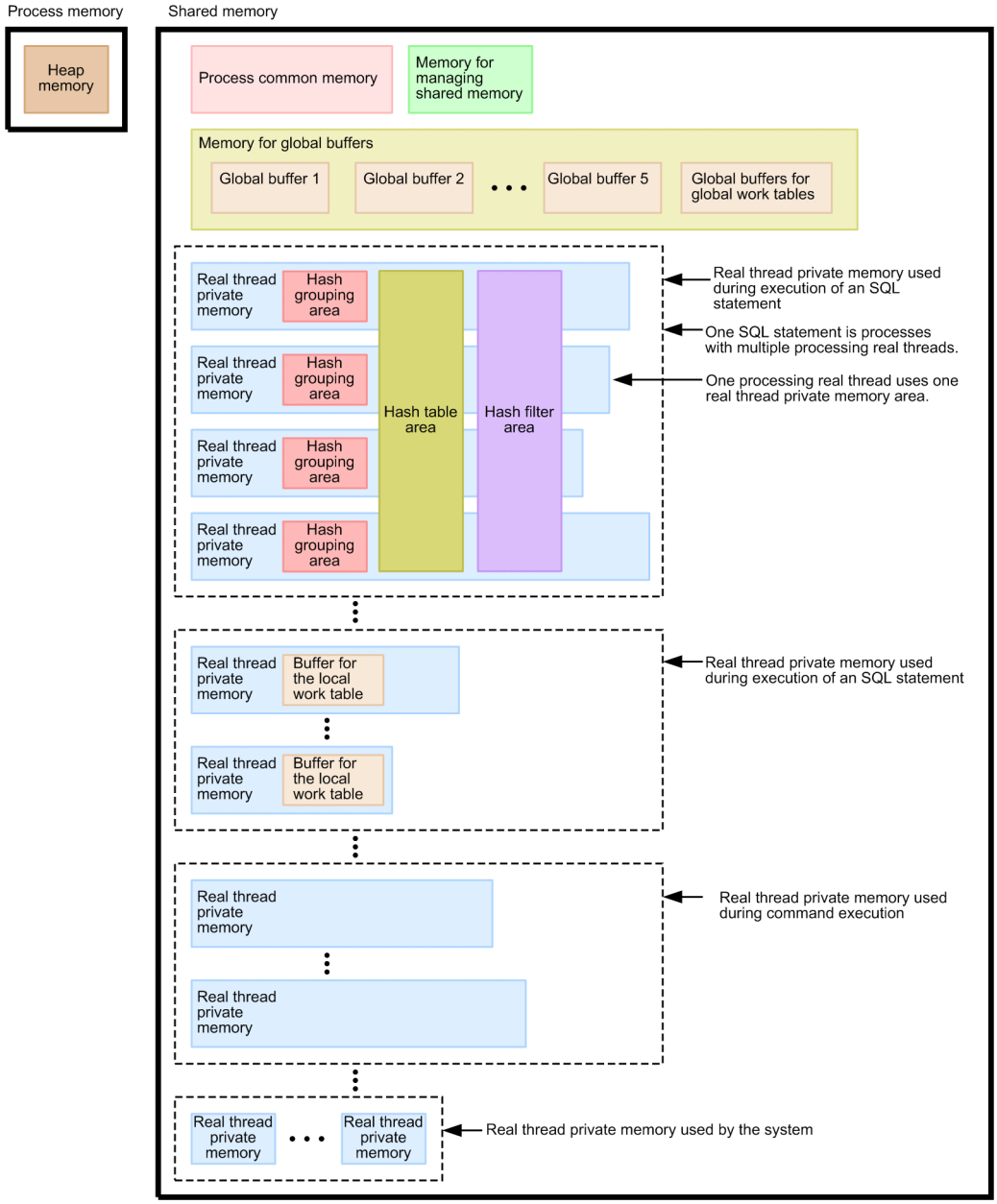
  Real thread private memory stores the information needed by the processing real threads that process an SQL statement or a command. One real thread private memory area is allocated to each processing real thread. Multiple processing real threads are run within an HADB server process, and the information required for each of those processing real threads is stored in real thread private memory.

■ **Process memory**

Heap memory is allocated in process memory. The HADB server uses heap memory during command execution.

The following figure shows the memory structure of the HADB server.

Figure 2-74: Memory structure of HADB servers



Explanation

- Multiple real thread private memory areas are allocated within shared memory.
- Multiple processing real threads are used to process one SQL statement or one command. Each of these processing real threads uses one real thread private memory.
- In addition to the real thread private memory areas that are used when SQL statements and commands are executed, one real thread private memory area is allocated for use by the system.

The following bullet points explain the areas and buffers that are allocated in real thread private memory.

- **Hash grouping area**

  A hash grouping area stores hash tables that are used for local hash grouping. When an SQL statement to which local hash grouping is applied is executed, a hash grouping area is allocated in each real thread private memory area.
- **Hash table area**

A hash table area stores the hash tables that are shared among multiple processing real threads. A hash table area is allocated if any of the following types of processing occur when an SQL statement is executed:

- Hash join as a table joining method
- Global hash grouping as a grouping method
- Hash execution as a method for processing subqueries
- Hash execution as a method for processing `SELECT DISTINCT`
- Hash execution as a method for processing the set operation

- **Hash filter area**

  A hash filter area stores the hash filters that are used for hash retrieval and are shared among multiple processing real threads. A hash filter area is allocated if any of the following types of processing occur when an SQL statement is executed:

  - Hash join as a table joining method
  - Hash execution as a method for processing subqueries

  > **Note**
  >
  > For details about hash filters, see *Table joining methods* or *How to process subqueries* in the *HADB Application Development Guide*.

- **Buffer for the local work table**

  A buffer for the local work table is used for input/output processing on local work tables. A local work table is a work table created for each processing real thread. A local work table is created if the `ORDER BY` clause is specified or when local hash grouping is performed.

> **Note**
>
> For details about the following types of processing, see the relevant sections in the *HADB Application Development Guide*.
>
> - Local hash grouping and global hash grouping: *Grouping methods*
> - Hash join: *Table joining methods*
> - Hash execution as a method for processing subqueries: *How to process subqueries*
> - Hash execution as a method for processing `SELECT DISTINCT`: *Method for processing `SELECT DISTINCT`*
> - Hash execution as a method for processing the set operation: *Methods for processing set operations*

2.20.2 Relationship with server definition operands

The various memory types and the sizes of the areas used by the HADB server are specified in server definition operands. This subsection explains the relationship between the memory types used by the HADB server and the server definition operands.

The two methods described below are provided for specifying in the server definition the maximum memory size to be used by the HADB server. We recommend use of method 1.

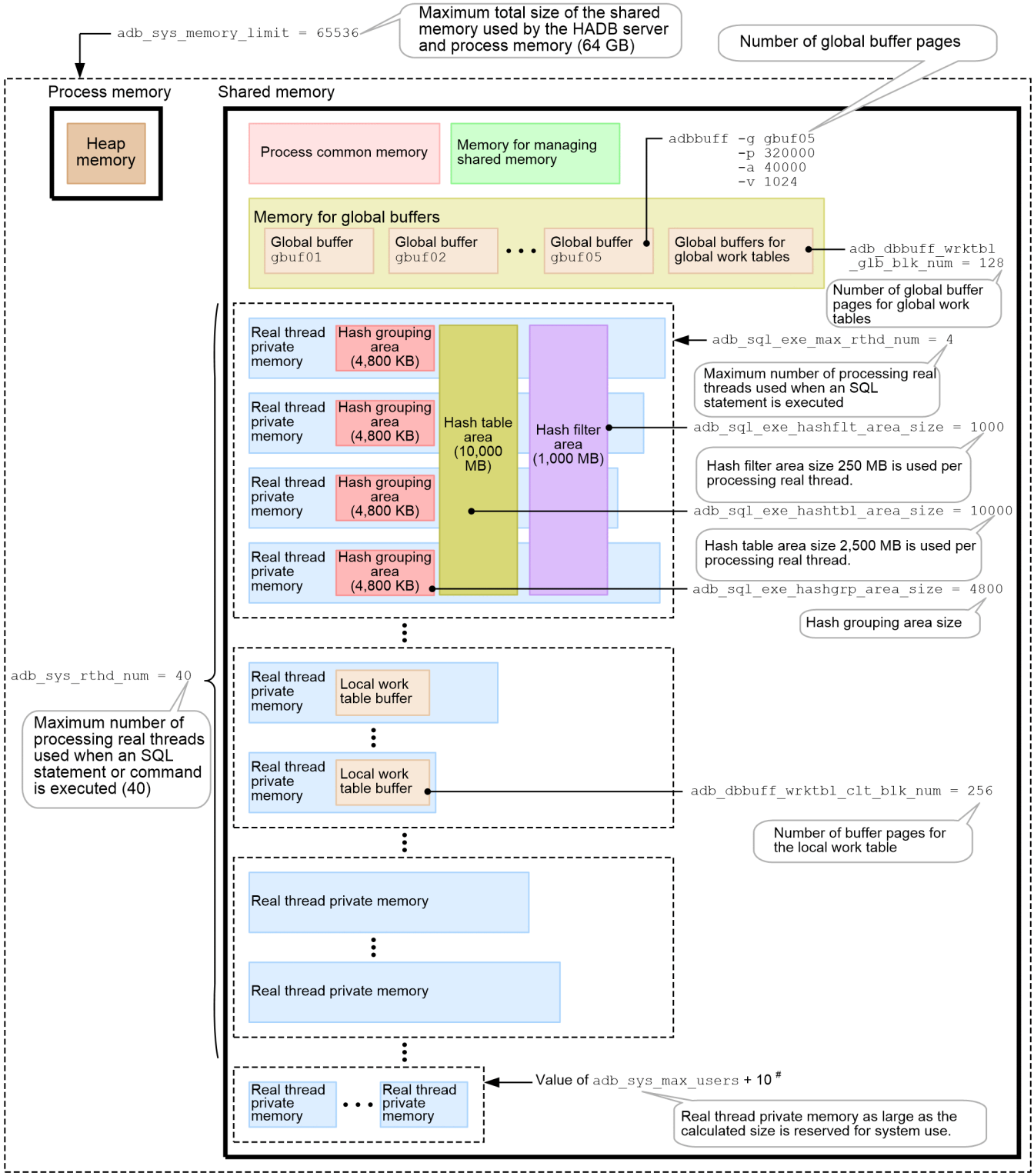
1. Specify in the `adb_sys_memory_limit` operand the maximum memory size to be used by the HADB server (maximum value for the sum of the shared memory size and the process memory size).
2. Specify in the `adb_sys_proc_area_max` operand the maximum value for the process common memory and specify in the `adb_sys_rthd_area_max` operand the maximum value for the real thread private memory.

The relationships between the memory used by the HADB server and the server definition operands are explained below for both of these methods.

(1) Method 1 (specifying the adb_sys_memory_limit operand)

The following figure shows the relationship between the memory used by the HADB server and the server definition operands.

Figure 2-75: Relationship between the memory used by the HADB server and the server definition operands (when the adb_sys_memory_limit operand is specified)



Note

The values of the server definition operands shown in the figure are examples. Specify the actual values that you have estimated.

#

When the multi-node function is used, the number of real thread private memory areas that are allocated is equal to *value-of-adb_sys_max_users* + *number-of-nodes* × 2 + 13.

Important

Specify in the `adb_sys_memory_limit` operand in the server definition the maximum size of the memory to be used by the HADB server (maximum value for the sum of the shared memory size and the process memory size).

The following table describes the relationships between the server definition operands and the individual memory types and areas shown in the figure.

Table 2-25: Explanation of individual memory types and areas used by the HADB server (when the adb_sys_memory_limit operand is specified)

Type of memory or area			Related server definition operand	Section describing the estimation method	Method of checking memory usage during HADB server operation
Shared memory	Process common memory		--	(3) Determining the process common memory requirement in 6.3.1 Determining the shared memory requirement.	The value of `PROCESS_USE` output when the `adbls -d mem` command#2 is executed with the `-a` option specified.
	Memory for managing shared memory		--	(1) Determining the shared memory management area requirement in 6.3.1 Determining the shared memory requirement.	The value of `OTHER_USE` output when the `adbls -d mem` command#2 is executed with the `-a` option specified.
	Memory for global buffer		• `adbbuff` Specifies information such as the number of global buffer pages. • `adb_dbbuff_wrktbl_glb_blk_num` Specifies the number of global buffer pages used for global work tables.	• (2) Determining the global buffer page requirement in 6.3.1 Determining the shared memory requirement. • 6.25.1 Estimating the number of pages in the global buffer for global work tables	The value of `GBUFFER_USE` output when the `adbls -d mem` command#2 is executed with the `-a` option specified.
	Real thread private memory	--	• `adb_sys_rthd_num` Specifies the maximum number of processing real threads that can be used when SQL statements and commands are executed. One real thread private memory area is allocated for each processing real thread. • `adb_sql_exe_max_rthd_num`#1 Specifies the maximum number of processing real threads that can be used when one SQL statement is executed.	(4) Determining the real thread private memory requirement in 6.3.1 Determining the shared memory requirement.	The value of `THREAD_USE` output when the `adbls -d mem` command#2 is executed with the `-a` option specified.

Type of memory or area			Related server definition operand	Section describing the estimation method	Method of checking memory usage during HADB server operation
		Hash grouping area	• `adb_sql_exe_hashgrp_area_size`[#1] Specifies the size of one hash grouping area.	Description of the adb_sql_exe_hashgrp_area_size operand in 7.2.2 Operands related to performance (set format)	
		Hash table area	• `adb_sql_exe_hashtbl_area_size`[#1] Specifies the size of a hash table area.	Description of the adb_sql_exe_hashtbl_area_size operand in 7.2.2 Operands related to performance (set format).	
		Hash filter area	• `adb_sql_exe_hashflt_area_size`[#1] Specifies the size of a hash filter area.	Description of the adb_sql_exe_hashflt_area_size operand in 7.2.2 Operands related to performance (set format)	
		Local work table buffer	• `adb_dbbuff_wrktbl_clt_blk_num`[#1] Specifies the number of pages in the buffer used for local work tables.	6.25.2 Estimating the number of pages in the buffer for local work tables	
Process memory	Heap memory		--	6.3.2 Determining the process memory requirement	The value of `HEAP_USE` output when the `adbls -d mem` command[#2] is executed with the `-a` option specified.

Legend:

--: Not applicable

#1

You can use the same operand in the client definition to change the value that is specified in the server definition.

#2

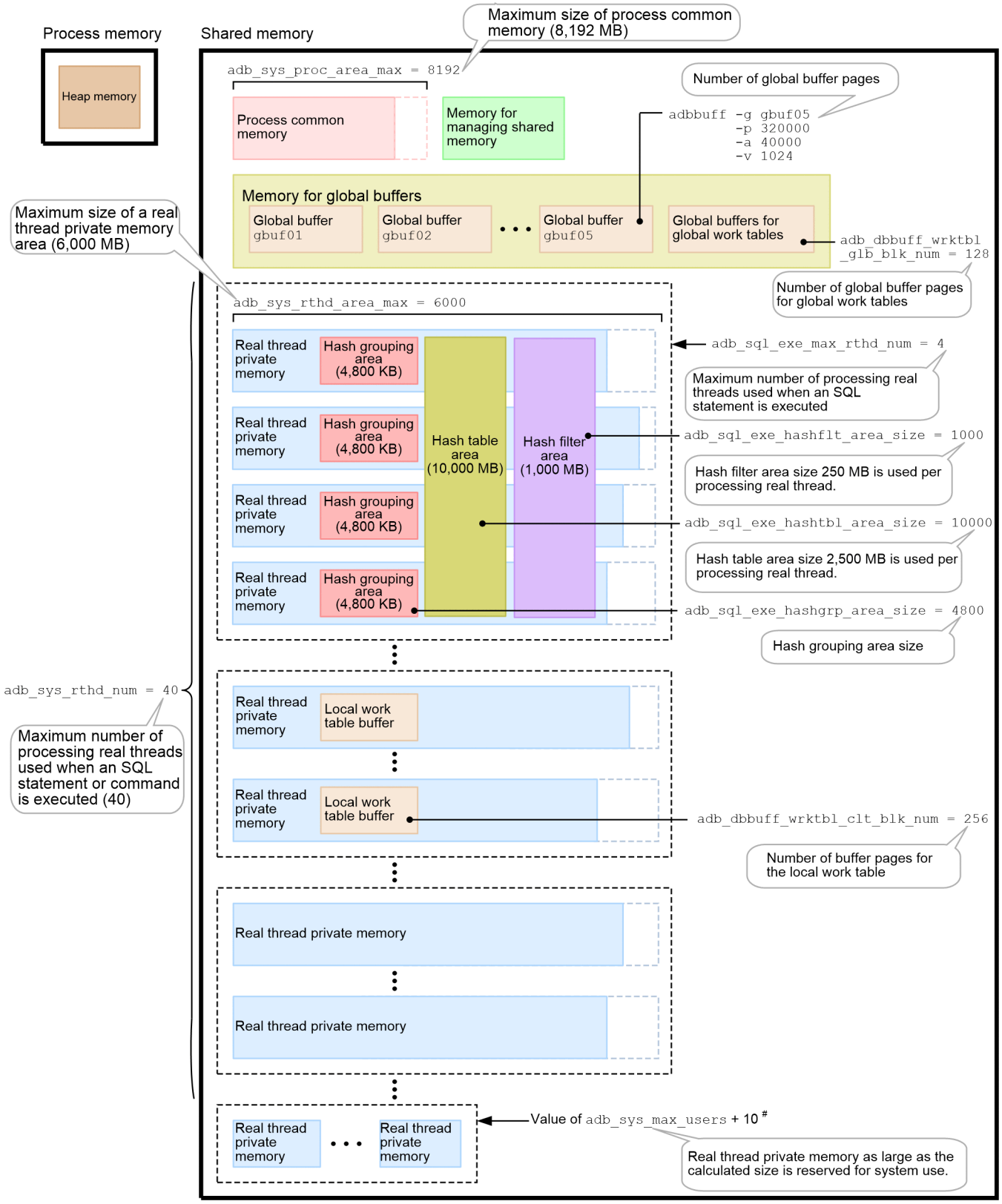
For details about the items output by the `adbls -d mem` command when the `-a` option is specified, see *adbls -d mem (Display the Memory Usage Status)* in the manual *HADB Command Reference*.

> **Note**
>
> For details about how to estimate the size of memory used by the HADB server, see the following sections:
>
> - 4.3 Estimating memory requirements
> - 6.3 Estimating the HADB server's memory requirement

(2) Method 2 (specifying the adb_sys_proc_area_max and adb_sys_rthd_area_max operands)

The following figure shows the relationship between the memory used by the HADB server and the server definition operands.

Figure 2-76: Relationship between the memory used by the HADB server and the server definition operands (when the adb_sys_proc_area_max and the adb_sys_rthd_area_max operands are specified)



Note

The values of the server definition operands shown in the figure are examples. Specify the actual values that you have estimated.

#

When the multi-node function is used, the number of real thread private memory areas that are allocated is equal to *value-of-adb_sys_max_users* + *number-of-nodes* × 2 + 13.

The following table describes the relationships between the server definition operands and the individual memory types and areas shown in the figure.

Table 2-26: Explanation of individual memory types and areas used by the HADB server (when the adb_sys_proc_area_max and adb_sys_rthd_area_max operands are specified)

Type of memory or area			Related server definition operand	Section describing the estimation method	Method of checking memory usage during HADB server operation
Shared memory	Process common memory[#1]		• `adb_sys_proc_area_max` Specifies the maximum size of process common memory.	(3) Determining the process common memory requirement in 6.3.1 Determining the shared memory requirement.	The value of `PROCESS_USE` output when the `adbls -d mem` command[#3] is executed with the `-a` option specified.
	Memory for managing shared memory		--	(1) Determining the shared memory management area requirement in 6.3.1 Determining the shared memory requirement.	The value of `OTHER_USE` output when the `adbls -d mem` command[#3] is executed with the `-a` option specified.
	Memory for global buffer		• `adbbuff` Specifies information such as the number of global buffer pages. • `adb_dbbuff_wrktbl_glb_blk_num` Specifies the number of global buffer pages used for global work tables.	• (2) Determining the global buffer page requirement in 6.3.1 Determining the shared memory requirement. • 6.25.1 Estimating the number of pages in the global buffer for global work tables	The value of `GBUFFER_USE` output when the `adbls -d mem` command[#3] is executed with the `-a` option specified.
	Real thread private memory[#1]	--	• `adb_sys_rthd_area_max` Specifies the maximum size of one real thread private memory area. • `adb_sys_rthd_num` Specifies the maximum number of processing real threads that can be used when SQL statements and commands are executed. One real thread private memory area is allocated for each processing real thread. • `adb_sql_exe_max_rthd_num`[#2] Specifies the maximum number of processing real threads that can be used	(4) Determining the real thread private memory requirement in 6.3.1 Determining the shared memory requirement.	The value of `THREAD_USE` output when the `adbls -d mem` command[#3] is executed with the `-a` option specified.

Type of memory or area			Related server definition operand	Section describing the estimation method	Method of checking memory usage during HADB server operation
			when one SQL statement is executed.		
		Hash grouping area	• adb_sql_exe_hashgrp_area_size[#2] Specifies the size of one hash grouping area.	Description of the adb_sql_exe_hashgrp_area_size operand in 7.2.2 Operands related to performance (set format)	
		Hash table area	• adb_sql_exe_hashtbl_area_size[#2] Specifies the size of a hash table area.	Description of the adb_sql_exe_hashtbl_area_size operand in 7.2.2 Operands related to performance (set format).	
		Hash filter area	• adb_sql_exe_hashflt_area_size[#2] Specifies the size of a hash filter area.	Description of the adb_sql_exe_hashflt_area_size operand in 7.2.2 Operands related to performance (set format)	
		Local work table buffer	• adb_dbbuff_wrktbl_clt_blk_num[#2] Specifies the number of pages in the buffer used for local work tables.	6.25.2 Estimating the number of pages in the buffer for local work tables	
Process memory	Heap memory		--	6.3.2 Determining the process memory requirement	The value of HEAP_USE output when the adbls -d mem command[#3] is executed with the -a option specified.

Legend:

--: Not applicable

#1

This part differs from (1) Method 1 (specifying the adb_sys_memory_limit operand).

If you omit the adb_sys_memory_limit operand, which specifies the maximum size of the memory used by the HADB server, you must specify the maximum size of the process common memory (adb_sys_proc_area_max operand) and the maximum size of one real thread private memory area (adb_sys_rthd_area_max operand).

#2

You can use the same operand in the client definition to change the value that is specified in the server definition.

#3

For details about the items output by the adbls -d mem command when the -a option is specified, see *adbls -d mem (Display the Memory Usage Status)* in the manual *HADB Command Reference*.

Note

For details about estimating the size of the memory to be used by the HADB server, see the following sections:

- 4.3 Estimating memory requirements

- 6.3 Estimating the HADB server's memory requirement

3 Guide for Building a Hands-on Environment

This chapter explains how to build a hands-on environment that will allow users to gain first-hand experience in installing and operating HADB.

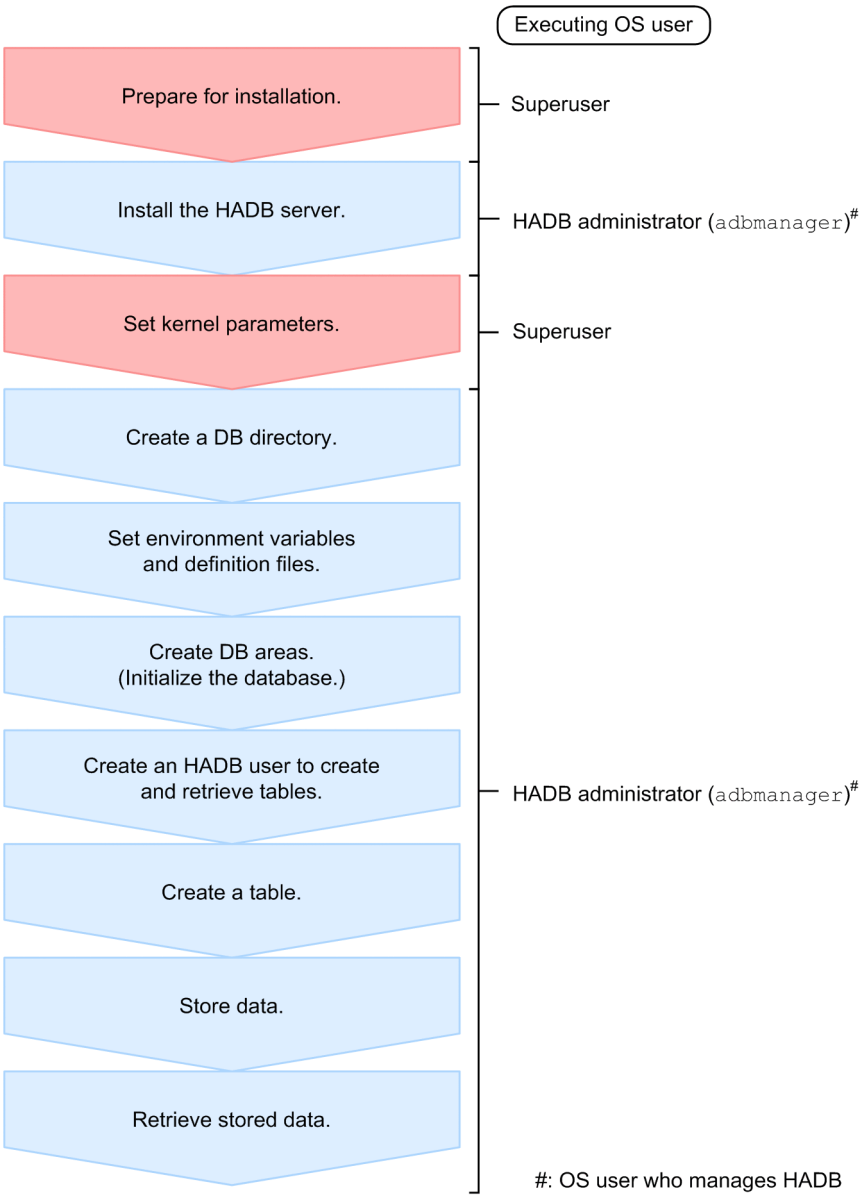
To fully understand the concepts involved in building an HADB server, we recommend that you build an environment on an actual machine while you read this chapter. This exercise will take between 1 and 2 hours.

Note that the procedures explained in this chapter and the specification values that are used are specifically for the purpose of configuring a hands-on environment. They are not appropriate for building an environment that will be used in actual operations. When you configure an environment that will be used for actual operations, first estimate the required values based on the explanation in *Part 3. Design* of this manual, and then configure your environment based on the explanation in *Part 4. Setup* of this manual.

3.1 General procedure for building a hands-on environment

The figure below shows the general procedure for building a hands-on environment for an HADB server. Details about how to build a hands-on environment are provided in the subsections that follow.

Figure 3-1: General procedure for building a hands-on environment



3.1.1 System configuration of the HADB server to be built

The machine environment for building a hands-on environment for an HADB server must have the following capacities:

- Size of installed memory: 8 gigabytes
- Available disk space: At least 2 gigabytes
- OS versions
  Either of the following OSs must be installed on the server machine:

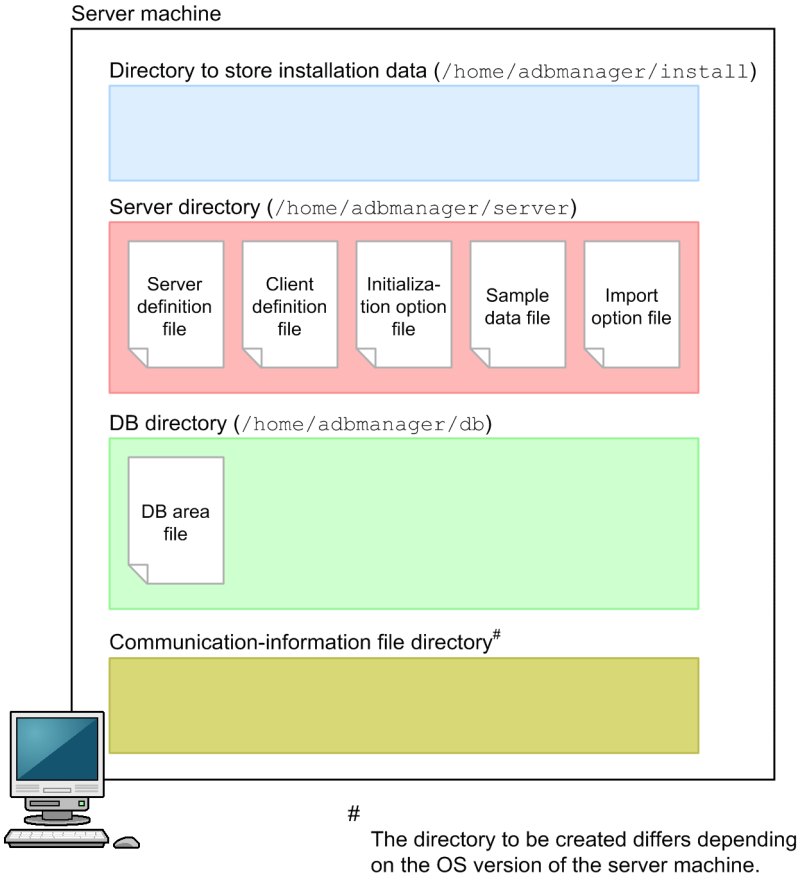
- Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  The HADB server runs on version 6.2 or later.
- Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  The HADB server runs on version 7.1 or later.

The OS must include the libraries and user commands that are prerequisite for the HADB server to run correctly. For details about the prerequisite libraries and user commands, see (1) Checking the prerequisite libraries and user commands in 8.2.1 Tasks that must be performed before installation.

Note that HADB runs only on the Intel 64 architecture. It does not run on the AMD 64 architecture.

The following figure shows the system configuration of the HADB server that will be built in the hands-on environment.

Figure 3-2: System configuration of the HADB server to be built



(1) Directory that stores the installation data

This is the directory to which you copy the HADB server installation data stored on the file system CD-ROM. The directory that stores the installation data stores the following two types of data:

- Installation command (`adbinstall` command)
- Installation data (`hitachi_advanced_data_binder_server-$VR.tar.gz` file)

Note:

`$VR` indicates the HADB version and release number.

(2) Server directory

The server directory is a directory that stores the definition files and various types of commands that are necessary for operating the HADB server.

During the installation of the HADB server, use the installation command (`adbinstall` command) to specify the directory to be used as the server directory (`/home/adbmanager/server`). The installation data (`hitachi_advanced_data_binder_server-$VR.tar.gz` file) is expanded under the server directory, which stores directories and files.

The following table lists and describes the files that are used to build a hands-on environment.

Table 3-1: Files in the server directory that are used in a hands-on environment

No.	File name	Description
1	Server definition file	Definitions necessary for operating the HADB server are specified in this file.
2	Client definition file	Definitions necessary for executing applications and commands are specified in this file.
3	Initialization option file	Options necessary for creating DB areas are specified in this file.
4	Sample data file	The data to be stored in the tables in the hands-on environment is entered into this text file.
5	Import option file	Options necessary for storing data in tables are specified in this file.

(3) DB directory

The DB directory is where the DB areas are stored when they are created during database initialization. Before initializing the database, the user must create the DB directory (`/home/adbmanager/db`).

When the database is initialized, the five DB areas shown in the table below are created. The only DB area that the user explicitly creates is the *data DB area*. Other DB areas are created automatically.

Table 3-2: List of DB areas that are created

No.	DB area name	Description
1	Data DB area	This DB area stores tables and indexes.
2	Work table DB area	This DB area stores the work tables created by HADB when executing SQL statements.
3	Master directory DB area	This DB area stores the system's internal information.
4	Dictionary DB area	This DB area stores dictionary tables and B-tree indexes of dictionary tables.
5	System-table DB area	This DB area stores system tables and B-tree indexes of system tables.

(4) Directories for storing communication-information files

Files of communication information that is necessary for operating the HADB server are stored in the directories for storing communication-information files.

Before the HADB server can be installed, a user must manually create directories that will store communication-information files. The directories that will store communication-information files differ depending on the OS version of the server machine.

If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)

- `/dev/HADB/pth`

- /lib/udev/devices/HADB/pth

If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)

- /dev/HADB/pth

Note that if the OS is Red Hat Enterprise Linux Server 7, a user must also create the following configuration file:

- /etc/tmpfiles.d/dev-HADB-pth.conf

3.1.2 Data stored in a table in the hands-on environment

The data to be stored in a table in the hands-on environment is the data (sample data) provided with the HADB server. The sample data contains the state codes, state names, ZIP codes, addresses of state capitol buildings, and area sizes of the 50 U.S. states. The following figure shows what the sample data stored in the table looks like.

Figure 3-3: Sample data stored in a table in the hands-on environment

STATECODE	STATENAME	ZIPCODE	ADDRESS	AREA
1	Alabama	36130-2751	State Capitol N-104 600 Dexter Avenue Montgomery	135765000000
2	Alaska	99811	State Capitol Juneau	1717854000000
3	Arizona	85007	State Capitol West Wing 1700 W. Washington, 9th Fl. Phoenix	295254000000
:	:	:	:	:

Legend:
STATECODE: State code
STATENAME: State name
ZIPCODE: ZIP code
ADDRESS: Address of the state capitol
AREA: Area of state in square meters

3.2 Installing the HADB server

This section explains the tasks to be performed to install the HADB server.

3.2.1 Preparations necessary for installation

Before installing the HADB server, you must set up on the OS the OS user who will manage HADB (*HADB administrator*) and the group to which the HADB administrator belongs (*HADB administrators group*).

This user must also create directories for storing the communication-information files necessary for operating the HADB server.

(1) Setting up the HADB administrator and HADB administrators group

You can assign any names you wish to the HADB administrator and HADB administrators group. The following names are used in this manual:

- HADB administrator: `adbmanager`
- HADB administrators group: `adbgroup`

1. Log on to the OS as a superuser.
   Once you have logged on, open a terminal window for entering commands.
2. Set up the HADB administrators group (`adbgroup`) on the OS.
   Enter the following operating system command and press **Enter**:

   ```
   groupadd adbgroup
   ```

3. Set up the HADB administrator (`adbmanager`) on the OS.
   Enter the operating system command below and press **Enter**:
   For the `-g` option, specify the HADB administrators group (`adbgroup`) set up in step 2.

   ```
   useradd -g adbgroup adbmanager
   ```

4. Set up a password for the HADB administrator on the OS.
   Enter the following operating system command and press **Enter**:

   ```
   passwd adbmanager
   ```

   When you execute the `passwd` command, the system asks you to enter the new password for `adbmanager` twice. Specify any password.

(2) Verifying that the HADB administrator and HADB administrators group have been set up correctly

Verify that the HADB administrator and HADB administrators group that have been set up on the OS are correct.

As a superuser, enter the following operating system command and press **Enter**:

```
id adbmanager
```

Executing the `id` command lets you check the information that has been specified for `adbmanager`. The following shows an example of the execution result.

■ Execution result example

```
uid=501(adbmanager) gid=501(adbgroup) group=501(adbgroup)
```

If `adbmanager` is shown for `uid`, and `adbgroup` is shown for `gid` and `group`, the settings are correct.

> **Note**
>
> Setting up the HADB administrator as an OS user belonging to the HADB administrators group on the OS gives the HADB administrator access privileges as the owner of the HADB server's various files and directories. As a result, the HADB administrator can prevent other users from overwriting crucial data, and can enhance the level of security. The HADB administrator also has the privileges needed to execute all HADB commands.

(3) Creating the directories for storing communication-information files

The directories that will store communication-information files differ depending on the OS version of the server machine.

- If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  See (a) Red Hat Enterprise Linux Server 6.
- If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  See (b) Red Hat Enterprise Linux Server 7.

(a) Red Hat Enterprise Linux Server 6

To store communication-information files, create the following directories:

- `/dev/HADB/pth`
- `/lib/udev/devices/HADB/pth`

The permission to be assigned to these directories is `777`.

1. Create the `/dev/HADB/pth` directory.
   As a superuser, enter the following operating system command, and then press **Enter**:
   ```
   mkdir -p -v -m 777 /dev/HADB/pth
   ```
2. Create the `/lib/udev/devices/HADB/pth` directory.
   As a superuser, enter the following operating system command, and then press **Enter**:
   ```
   mkdir -p -v -m 777 /lib/udev/devices/HADB/pth
   ```

(b) Red Hat Enterprise Linux Server 7

To store communication-information files, create the following directory and file:

- `/dev/HADB/pth`
  The permission to be assigned to the directory is `777`.
- `/etc/tmpfiles.d/dev-HADB-pth.conf`

The permission to be assigned to the configuration file is `644` (default). Because this is the default permission, you do not need to specify this permission.

1. Create the `/dev/HADB/pth`.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   mkdir -p -v -m 777 /dev/HADB/pth
   ```

2. Create the configuration file (`dev-HADB-pth.conf`).

   As a superuser, enter the following operating system command, and then press **Enter**: The configuration file is created.

   ```
   vi /etc/tmpfiles.d/dev-HADB-pth.conf
   ```

3. Edit the configuration file.

   Press the **I** key to place the editor in edit mode. When the editor is placed in edit mode, enter the following settings:

   ```
   # Type Path Mode UID  GID  Age Argument
   d /dev/HADB/pth 0777 root root - -
   ```

4. Terminate editing the configuration file.

   When entry finishes, press the **Esc** key to place the editor in command mode. When the editor is placed in command mode, enter the following command, and then press the **Enter** key.

   ```
   :wq
   ```

   The settings specified in edit mode are saved. Creation of the configuration file is completed when this procedure finishes.

(4) Checking the created directories for storing communication-information files

Check that the directories for storing communication-information files have been created correctly.

- If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  See (a) Red Hat Enterprise Linux Server 6.
- If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  See (b) Red Hat Enterprise Linux Server 7.

(a) Red Hat Enterprise Linux Server 6

Use the following procedure to check whether the two directories for storing communication-information files have been created correctly.

1. Check whether the `/dev/HADB/pth` directory has been created.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -d -l /dev/HADB/pth
   ```

   From the execution result of the `ls` command, you can check whether the `/dev/HADB/pth` directory has been created and whether the proper permission has been assigned. The following shows an example of the execution result.

   ▪ Execution result example

   ```
   drwxrwxrwx 2 root root 40  Aug 13 19:28 2013 /dev/HADB/pth
   ```

Verify that the result begins with `drwxrwxrwx` and ends with `/dev/HADB/pth`.

2. Check whether the `/lib/udev/devices/HADB/pth` directory has been created.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -d -l /lib/udev/devices/HADB/pth
   ```

   From the execution result of the `ls` command, you can check whether the `/lib/udev/devices/HADB/pth` directory has been created and whether the proper permission has been assigned. The following shows an example of the execution result.

   ▪ **Execution result example**

   ```
   drwxrwxrwx 2 root root 4096  Aug 13 19:29 2013 /lib/udev/devices/HADB/pth
   ```

   Verify that the result begins with `drwxrwxrwx` and ends with `/lib/udev/devices/HADB/pth`.

(b) Red Hat Enterprise Linux Server 7

Use the following procedure to check whether the directory for storing communication-information files has been created correctly and whether the configuration file has been created correctly.

1. Check whether the `/dev/HADB/pth` directory has been created.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -d -l /dev/HADB/pth
   ```

   From the execution result of the `ls` command, you can check whether the `/dev/HADB/pth` directory has been created and whether the proper permission has been assigned. The following shows an example of the execution result.

   ▪ **Execution result example**

   ```
   drwxrwxrwx 2 root root 40  Aug 13 19:28 2013 /dev/HADB/pth
   ```

   Verify that the result begins with `drwxrwxrwx` and ends with `/dev/HADB/pth`.

2. Check whether the configuration file (`dev-HADB-pth.conf`) has been created.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -l /etc/tmpfiles.d/dev-HADB-pth.conf
   ```

   From the execution result of the `ls` command, you can check whether the `/etc/tmpfiles.d/dev-HADB-pth.conf` file has been created correctly and whether the default permission has been assigned. The following shows an example of the execution result.

   ▪ **Execution result example**

   ```
   -rw-r--r-- 2 root root 40  Aug 13 19:29 2013 /etc/tmpfiles.d/dev-HADB-pth.conf
   ```

   Verify that the result begins with `-rw-r--r--` and ends with `/etc/tmpfiles.d/dev-HADB-pth.conf`.

(5) Related item

8.2.1 Tasks that must be performed before installation

3.2.2 Installation

After you have set up the HADB administrator and HADB administrators group under the operating system, and have created directories for storing communication-information files, you are ready to install the HADB server.

Install the HADB server as the HADB administrator (`adbmanager`). If you are logged on as a superuser, log off and then log on as the HADB administrator (`adbmanager`).

(1) Creating the directory for storing the installation data

Create a directory to store the HADB server installation data. The path name of the directory to be created is as follows:

- Absolute path of the directory for storing the installation data: `/home/adbmanager/install`

As the HADB administrator (`adbmanager`), enter the following operating system command and press **Enter**:

```
mkdir /home/adbmanager/install
```

(2) Checking the created directory for storing the installation data

Enter the following operating system command to verify that the directory for storing the installation data has been created correctly by the `mkdir` command, and press **Enter**:

```
ls -d /home/adbmanager/install
```

Executing the `ls` command allows you to check that the directory for storing the installation data has been created correctly. The following shows an example of the execution result.

■ Execution result example

```
/home/adbmanager/install
```

If the directory shown above is displayed, the directory for storing the installation data has been created.

(3) Granting write permission to the directory that stores installation data

Grant write permission to the `/home/adbmanager/install` directory so that the HADB administrator can store the installation data in that directory. To do this, enter the following command and press **Enter**:

```
chmod 755 /home/adbmanager/install
```

Write permission has now been granted to the directory in which the installation data is to be stored.

(4) Mounting the file system CD-ROM

Allow the file system CD-ROM containing the installation command (`adbinstall` command) and the installation data (`tar.gz` file) for the HADB server to mount automatically.

If the file system CD-ROM does not mount automatically, you must mount it manually. To do so, enter the following command and press **Enter**:

```
mount /dev/cdrom /media
```

The underscored portion shows the name of the mount directory for the file system CD-ROM. It will vary depending on the environment.

> **Important**
>
> Depending on the machine being used, the directory names and file names indicated on the CD-ROM might be different from those shown above. Enter the directory name that is displayed exactly as shown when you execute the `ls` operating system command.

(5) Copying the installation command (adbinstall command)

Copy the HADB server installation command (`adbinstall` command) that is stored in the mounted file system CD-ROM to the `/home/adbmanager/install` directory.

Enter the following operating system command and press **Enter**:

```
cp /media/adbinstall /home/adbmanager/install
```

The underlined portion is the name of the mount directory for the file system CD-ROM. It varies depending on the environment.

(6) Copying the installation data (tar.gz file)

Copy to the `/home/adbmanager/install` directory the installation data (`tar.gz` file) for the HADB server that is stored on the mounted file system CD-ROM.

Enter the following operating system command and press **Enter**:

```
cp /media/hitachi_advanced_data_binder_server-$VR.tar.gz /home/adbmanager/install
```

The underlined portion is the name of the mount directory for the file system CD-ROM. It varies depending on the environment. `$VR` indicates the HADB version and release number.

(7) Checking the copied installation command (adbinstall command) and installation data (tar.gz file)

Verify that the `adbinstall` command and `tar.gz` file have been copied to the `/home/adbmanager/install` directory. Enter the following operating system command and press **Enter**:

```
ls /home/adbmanager/install
```

Executing the `ls` command allows you to verify that the `adbinstall` command and `tar.gz` file have been copied to the `/home/adbmanager/install` directory. The following shows an example of the execution result.

■ Execution result example

```
adbinstall hitachi_advanced_data_binder_server-$VR.tar.gz
```

Note:

`$VR` indicates the HADB version and release number.

(8) Assigning execution privileges to the installation command (adbinstall command)

Assign execution privileges to the HADB server installation command (`adbinstall` command) that was copied to the `/home/adbmanager/install` directory, so that the HADB administrator has execution privileges. Enter the following operating system command and press **Enter**:

```
chmod 777 /home/adbmanager/install/adbinstall
```

The execution privilege has now been assigned to the installation command (`adbinstall` command).

(9) Installing the HADB server

Install the HADB server by executing the HADB server installation command (`adbinstall` command) to which execution privileges were assigned.

Enter the following HADB command and press **Enter**:

```
/home/adbmanager/install/adbinstall -s /home/adbmanager/server
```

`-s /home/adbmanager/server`

When you use the `adbinstall` command to install the HADB server, specify the directory to be used as the server directory in the `-s` option. Here, specify the `/home/adbmanager/server` directory.

Executing the `adbinstall` command expands the installation data (`tar.gz` file) and stores the directories and files that comprise the HADB server in the `/home/adbmanager/server` directory.

> **Important**
>
> If you execute the `adbinstall` command as superuser (`root`) rather than as HADB administrator (`adbmanager`), the `KFAA91558-W` message is output. In this case, reply `n` or `N` to the `KFAA91559-Q` message that is output after the `KFAA91558-W` message. After that, log out, and then log in again as HADB administrator (`adbmanager`). Then, execute the `adbinstall` command as HADB administrator (`adbmanager`).

(10) Checking that the HADB server has been installed

Check that the HADB server has been installed correctly by entering the following operating system command and pressing **Enter**:

```
ls /home/adbmanager/server
```

Executing the `ls` command enables you to check that the HADB server has been installed. The following shows an example of the execution result.

■ Execution result example

```
bin  client  conf  include  lib  sample  spool
```

If the directory shown above is displayed, the HADB server has been installed.

(11) Related item

8.2.2 Installation procedure

3.3 Setting the kernel parameters

This section explains how to set the kernel parameters (Linux operating system parameters) that are required for configuring HADB servers.

3.3.1 Setting and checking the kernel parameters

Kernel parameters are set in the following files:

- `sysctl.conf` file
- `limits.conf` file

Kernel parameters are set by a superuser. If you are logged on as the HADB administrator (`adbmanager`), log off and then log on again as a superuser.

> **Note**
>
> This section explains how to use the `vi` command to set the kernel parameters in each file. You can also use a text editor to open and edit the individual files.

(1) Setting the kernel parameters in the sysctl.conf file

First, set the kernel parameters in the `sysctl.conf` file. The `sysctl.conf` file is stored at the following location:

- `sysctl.conf` file: `/etc/sysctl.conf`

1. Open the `sysctl.conf` file.

   As a superuser, enter the operating system command below and press **Enter**. Executing the `vi` command opens the `sysctl.conf` file in the edit mode:

   ```
   vi /etc/sysctl.conf
   ```

2. Specify the kernel parameters.

   Press the **I** key to go into the insert mode. Once you are in the insert mode, use the cursor movement keys to move the cursor to the location where the kernel parameters are to be inserted and enter the following kernel parameters:

   ```
   net.core.rmem_default = 33554432
   net.core.rmem_max = 33554432
   net.core.wmem_default = 33554432
   net.core.wmem_max = 33554432
   fs.aio-max-nr = 65536
   ```

   > **Note**
   >
   > If these kernel parameters are already specified in the `sysctl.conf` file, make their existing lines into comments so that their settings can be restored. To make a line into a comment, enter a hash mark (#) at the beginning of the line.
   >
   > **Example**
   >
   > Regard `#net.core.rmem_default = 10000` as a comment and ignore its value:

```
#net.core.rmem_default = 10000
net.core.rmem_default = 33554432
```

The following table provides the details of the kernel parameters that are to be specified.

Table 3-3: List of kernel parameters to be specified in the sysctl.conf file

No.	Kernel parameter	Description
1	net.core.rmem_default	Specifies the default window size for receive operations. Specify 33554432.
2	net.core.rmem_max	Specifies the maximum window size for receive operations. Specify 33554432.
3	net.core.wmem_default	Specifies the default window size for send operations. Specify 33554432.
4	net.core.wmem_max	Specifies the maximum window size for send operations. Specify 33554432.
5	fs.aio-max-nr	Specify 65536.

3. End setup of the kernel parameters.

   When you have completed entering the required kernel parameters in the insert mode, press **Esc** to go into the command mode. When you are in the command mode, enter the following command and press **Enter**:

```
:wq
```

   The entries you made in the insert mode are saved. Kernel parameters have now been set in the sysctl.conf file.

(2) Verifying the kernel parameters that were set in the sysctl.conf file

To verify that the kernel parameters have been set correctly in the sysctl.conf file, enter the command shown below and press **Enter**. You can check the kernel parameters that have been set in the sysctl.conf file.

```
sysctl -p
```

Executing the sysctl -p command displays a list of the kernel parameters that are set in the sysctl.conf file. Because the list of kernel parameters that are displayed is extremely long, the following shows only an excerpt as an example of the execution results.

■ Execution result example

```
          :
net.core.rmem_default = 33554432
net.core.rmem_max = 33554432
net.core.wmem_default = 33554432
net.core.wmem_max = 33554432
fs.aio-max-nr = 65536
          :
```

(3) Setting the kernel parameters in the limits.conf file

Set the kernel parameters in the limits.conf file. The limits.conf file is stored at the following location:

- limits.conf file: /etc/security/limits.conf

1. Open the limits.conf file.

Enter the following OS command and press **Enter**: By executing the `vi` command, you can edit the `limits.conf` file.

```
vi /etc/security/limits.conf
```

2. Specify the kernel parameters.

Press the **I** key to go into the insert mode. Once you are in the insert mode, use the cursor movement keys to move the cursor to the location where kernel parameters are to be inserted and enter the following kernel parameters:

```
adbmanager   soft   nofile    6400
adbmanager   hard   nofile    6400
adbmanager   soft   memlock   unlimited
adbmanager   hard   memlock   unlimited
```

Note

If these kernel parameters are already specified in the `limits.conf` file, make their existing lines into comments so that their settings can be restored. To change a line to a comment, enter a hash mark (#) at the beginning of the line.

Example

```
#adbmanager soft nofile 1000
adbmanager soft nofile 6400
```

The following table provides the details of the kernel parameters that are to be specified.

Table 3-4: List of kernel parameters to be specified in the limits.conf file

No.	Kernel parameter	Description
1	`soft nofile`	Specifies the number of file descriptors that can be opened. Specify `soft nofile` for the HADB administrator (`adbmanager`). Specify `6400` for `soft nofile`.
2	`hard nofile`	Specifies the number of file descriptors that can be opened. Specify `hard nofile` for the HADB administrator (`adbmanager`). Specify `6400` for `hard nofile`.
3	`soft memlock`	Specifies the upper limit of the memory lock for the shared memory. Specify `soft memlock` for the HADB administrator (`adbmanager`). Specify `unlimited` for `soft memlock`.
4	`hard memlock`	Specifies the upper limit of the memory lock for the shared memory. Specify `hard memlock` for the HADB administrator (`adbmanager`). Specify `unlimited` for `hard memlock`.

Important

The value `6400` specified in `soft nofile` and `hard nofile` is for building the hands-on environment. When configuring an environment that will be used for actual operations and not a hands-on environment, first estimate the required values according to 6.2 Estimating the kernel parameters, and then specify those values.

When you configure an environment that will be used for actual operations, there are other kernel parameters that will need to be specified in addition to those specified in the `sysctl.conf` and `limits.conf` files in this chapter.

3. End setup of the kernel parameters.

   When you have completed entering the required kernel parameters in the insert mode, press **Esc** to go into the command mode. When you are in the command mode, enter the following command and press **Enter**:

   ```
   :wq
   ```

   The entries you made in the insert mode are saved. Kernel parameters have now been set in the `limits.conf` file.

(4) Verifying the kernel parameters that were set in the limits.conf file

To verify that the kernel parameters have been set correctly in the `limits.conf` file, enter the OS command shown below and press **Enter**. By changing from the superuser to the HADB administrator (`adbmanager`), you can check the kernel parameters that have been set in the `limits.conf` file.

```
su adbmanager
```

The user is temporarily changed to the HADB administrator (`adbmanager`). As the HADB administrator, enter the following OS command and press **Enter**:

```
ulimit -a
```

Executing the `ulimit -a` command displays a list of the kernel parameters that are set for the HADB administrator (`adbmanager`). Because the list of kernel parameters that are displayed is extremely long, the following shows only an excerpt as an example of the execution results.

■ Execution result example

```
             :
max locked memory       (kbytes, -l) unlimited
open files                      (-n) 6400
             :
```

After checking the execution result example, change the user back from the HADB administrator (`adbmanager`) to the superuser by entering the following OS command and pressing **Enter**:

```
exit
```

(5) Enabling the kernel parameters

You must restart the operating system to apply the kernel parameters that you have set in the `sysctl.conf` and `limits.conf` files. Enter the following OS command and press **Enter**:

```
reboot
```

(6) Related items

6.2 Estimating the kernel parameters

3.4 Creating the DB directory

This section explains how to create the DB directory for storing DB areas.

3.4.1 Creating and checking the DB directory

In this section, you create the *DB directory* in which DB areas are stored. A DB area is a logical area for storing tables. A DB area is created when a database is initialized and stored in the DB directory. You create a DB area by following the procedure in 3.6 Creating DB areas (initializing the database).

The DB directory to be created here can have any name. The following directory name is used in this manual:

- Absolute path of the DB directory to be created: `/home/adbmanager/db`

The DB directory is created by the *HADB administrator (`adbmanager`)*. If you are logged on as a superuser, log off once and log on again as the HADB administrator (`adbmanager`).

(1) Executing the DB directory creation command

As the HADB administrator (`adbmanager`), enter the following operating system command and press **Enter**:

```
mkdir /home/adbmanager/db
```

(2) Confirming the created DB directory

To verify that the DB directory was created by the `mkdir` command, enter the following operating system command and press **Enter**:

```
ls -d /home/adbmanager/db
```

By specifying the `-d` option in the `ls` command, you can verify that the specified directory was created. The following shows an example of the execution result.

■ Execution result example

```
/home/adbmanager/db
```

Once you have confirmed that the DB directory exists, creation of the DB directory is complete.

3.5 Setting environment variables and definition files

This section explains how to set the environment variables and definition files that are necessary for operating the HADB server.

Environment variables and definition files are set by the *HADB administrator (adbmanager)*.

3.5.1 Setting environment variables

The environment variables that are necessary for operating the HADB server are set in the `.bashrc` file. This file is stored in the following location:

- `.bashrc` file: `/home/adbmanager/.bashrc`

This subsection explains how to use the `vi` command to set environment variables in the `.bashrc` file.

> **Note**
>
> You can also edit the `.bashrc` file using other text editors.

(1) Setting environment variables in the .bashrc file

1. Open the `.bashrc` file.

   As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. When you execute the `vi` command, the `.bashrc` file is opened in edit mode.

   ```
   vi /home/adbmanager/.bashrc
   ```

2. Set environment variables.

   Press the **I** key to shift to insert mode. Once you are in insert mode, use the cursor keys to move the cursor to the location where environment variables go and enter the following environment variables:

   ```
   export ADBDIR=/home/adbmanager/server
   export ADBCLTDIR=$ADBDIR
   export LD_LIBRARY_PATH=$ADBDIR/lib:$ADBDIR/client/lib
   export PATH=$PATH:$ADBDIR/bin:$ADBDIR/client/bin
   export ADBLANG=UTF8
   export ADBCLTLANG=UTF8
   ```

   The following table shows the details of the environment variables that you are entering.

   Table 3-5: List of environment variables to be specified

No.	Environment variable	Value to be specified	Description
1	`ADBDIR`	`/home/adbmanager/server`	Specifies the absolute path to the server directory.
2	`ADBCLTDIR`	`$ADBDIR`	Specifies the same path as `ADBDIR` in order to execute applications on the HADB server. Specify the value shown here.
3	`LD_LIBRARY_PATH`	• `$ADBDIR/lib` • `$ADBDIR/client/lib`	• Specifies the `lib` directory (`$ADBDIR/lib`) under the server directory.

No.	Environment variable	Value to be specified	Description
			• To execute applications on the HADB server, specify the `client/lib` directory (`$ADBDIR/client/lib`) under the server directory.
4	`PATH`	• `$PATH` • `$ADBDIR/bin` • `$ADBDIR/client/bin`	• To retain the path setting valid for the OS, specify `$PATH`. • Specify the `bin` directory (`$ADBDIR/bin`) under the server directory. • To execute applications on the HADB server, specify the `client/bin` directory (`$ADBDIR/client/bin`).
5	`ADBLANG`	`UTF8`	Specifies the character encoding to be used.
6	`ADBCLTLANG`	`UTF8`	

3. Finish setting the environment variables.

   After entering the required environment variables in insert mode, press **Esc** to shift to command mode. Once you are in command mode, enter the following command and press **Enter**:

   ```
   :wq
   ```

   The content that was specified in edit mode is saved. The environment variables have now been set in the `.bashrc` file.

4. Enable the environment variables that have been set.

   Enter the following operating system command and press **Enter**:

   ```
   source /home/adbmanager/.bashrc
   ```

   Executing the `source` command enables the environment variables that were set in the `.bashrc` file.

(2) Verifying the environment variables that were set in the .bashrc file

To verify that the environment variables have been correctly set, enter the following operating system command and press **Enter**:

```
env
```

Executing the `env` command displays a list of the environment variables that are valid for the OS. Because the list of environment variables to be displayed is extremely long, the following shows only an excerpt as an example of the execution results.

■ Execution result example

```
          :
LD_LIBRARY_PATH=/home/adbmanager/server/lib:/home/adbmanager/server/client/lib
ADBCLTDIR=/home/adbmanager/server
PATH=/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:
/sbin:/home/adbmanager/bin:/home/adbmanager/server/bin:/home/adbmanager/server/cli
ent/bin
ADBLANG=UTF8
ADBDIR=/home/adbmanager/server
ADBCLTLANG=UTF8
          :
```

(3) Related item

8.4 Setting environment variables

3.5.2 Editing the definition files

Edit the *server definition files* necessary for operating the HADB server and the *client definition files* necessary for executing applications and commands.

A file template is available for each definition file. Therefore, copy the file template and edit it.

The templates for the definition files are stored in the following locations:

■ Storage locations of definition file templates

- Server definition file template: `/home/adbmanager/server/sample/conf/server.def`
- Client definition file template: `/home/adbmanager/server/sample/conf/client.def`

The definition files (`server.def` and `client.def`) created by copying and editing the templates are stored in the following location:

- Storage location of the definition files: `/home/adbmanager/server/conf/`
  By storing the edited definition files in the above location, you can activate the HADB server.

This subsection explains how to use the `vi` command to edit each definition file.

> **Note**
>
> You can also edit each definition file using other text editors.

(1) Editing the server definition file

1. Copy the server definition file template.
   As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. The server definition file template is copied.

   ```
   cp /home/adbmanager/server/sample/conf/server.def /home/adbmanager/server/conf/
   ```

2. Open the copied server definition file template.
   Enter the operating system command below and press **Enter**. When you execute the `vi` command, the server definition file is opened in edit mode.

   ```
   vi /home/adbmanager/server/conf/server.def
   ```

3. Specify the DB directory in which to store the database.
   Press the **I** key to shift to insert mode. Once you are in insert mode, use the cursor keys to move the cursor to the following operand:

   ```
   set adb_db_path = XXXXX
   ```

   After you have moved the cursor, delete `XXXXX`. Then, enter the absolute path to the DB directory (`/home/adbmanager/db`).

■ **Input result**

```
set adb_db_path = /home/adbmanager/db
```

4. Change values specified for the `TBLBUF01` global buffer.

   Use the cursor keys to move the cursor to the following operand:

```
adbbuff -g TBLBUF01 \
        -n ADBUTBL01 \
        -p 1000000 \
        -v 1024
```

   Note

   Interpret \ as a half-width backslash.

   After repositioning the cursor, delete `1000000` from the `-p` option. Then, enter `1000`.

   Next, delete `1024` from the `-v` option. Then, enter `64`.

   ■ **Input result**

```
adbbuff -g TBLBUF01 \
        -n ADBUTBL01 \
        -p 1000 \
        -v 64
```

5. Change a value specified for the `IDXBUF01` global buffer.

   Use the cursor keys to move the cursor to the following operand:

```
adbbuff -g IDXBUF01 \
        -n ADBUIDX01 \
        -p 2500000
```

   Note

   Interpret \ as a half-width backslash.

   After repositioning the cursor, delete `2500000` from the `-p` option. Then, enter `250`.

   ■ **Input result**

```
adbbuff -g IDXBUF01 \
        -n ADBUIDX01 \
        -p 250
```

> **Note**
>
> A global buffer is the area that is used for inputting data to, and outputting data from, tables stored in a DB area.

6. Finish editing the server definition file.

   For this environment, there is no need to edit operands other than those described above. Press **Esc** to shift to command mode. Once you are in command mode, enter the following command and press **Enter**:

```
:wq
```

   The content that was specified in edit mode is saved. The information necessary for operating the HADB server is now set in the server definition file.

(2) Editing the client definition file

1. Copy the client definition file template.

Enter the operating system command below and press **Enter**. The client definition file template is copied.

```
cp /home/adbmanager/server/sample/conf/client.def /home/adbmanager/server/conf/
```

2. Open the copied client definition file template.

   Enter the operating system command described below and press **Enter**. When you execute the `vi` command, the client definition file is opened in edit mode.

   ```
   vi /home/adbmanager/server/conf/client.def
   ```

3. Specify the host name of the HADB server.

   Use the cursor keys to move the cursor to the following operand:

   ```
   set adb_clt_rpc_srv_host = XXXXX
   ```

   After you have moved the cursor, delete `XXXXX`. Then, enter `localhost`.

   ■ Input result

   ```
   set adb_clt_rpc_srv_host = localhost
   ```

4. Finish editing the client definition file.

   For this environment, there is no need to edit operands other than those described above. After you have entered the host name in edit mode, press **Esc** to shift to command mode. Once you are in command mode, enter the following command and press **Enter**:

   ```
   :wq
   ```

   The content that was specified in edit mode is saved. The information necessary for executing applications and commands is now set in the client definition file.

   > **Note**
   >
   > The client definition file need to be defined on the HADB server that executes applications or on the HADB client. In this case, in order to execute applications on the HADB server, the client definition file located on the HADB server is edited.

(3) Related items

- 7. Designing the Server Definition
- 8.5 Creating and modifying a server definition
- *Creating a client definition* in *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*

3.6 Creating DB areas (initializing the database)

DB areas are created when you initialize the database.

A DB area is a logical area in which the tables and indexes are stored. DB areas are stored in the DB directory you create in 3.4 Creating the DB directory.

An *initialization option file* is required when creating a DB area. It specifies the type of DB area to be created. This subsection explains how to edit the initialization option file and how to create DB areas.

DB areas are created by the *HADB administrator (adbmanager)*.

Note

This section explains how to create DB areas by initializing the database. To initialize the database, you use the DB area creation command (`adbinit` command). Use this command when you first create DB areas after installing the HADB server.

You cannot use the DB area creation command (`adbinit` command) to add or modify DB areas after the database has been initialized. In this situation, you use the command for adding and changing DB areas (`adbmodarea` command). For details about the command for adding and changing DB areas (`adbmodarea` command), see *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

3.6.1 Editing the initialization option file

This subsection explains how to use the `vi` command to edit the initialization option file.

A file template is available for the initialization option file. Therefore, copy and edit the file template.

The template for the initialization option file is stored in the following location:

- Initialization option file template: `/home/adbmanager/server/sample/conf/adbinit.opt`

The initialization option file created by editing the template is stored in the following location:

- Storage location of the initialization option file: `/home/adbmanager/server/conf/`
  By storing the edited initialization option file in the above location, you can create a DB area.

  Note

  You can also edit the initialization option file using other text editors.

(1) Editing the DB area page size

1. Copy the initialization option file template.
   As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. The initialization option file template is copied.

```
cp /home/adbmanager/server/sample/conf/adbinit.opt /home/adbmanager/server/conf/
```

2. Open the copied initialization option file template.

   As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. When you execute the `vi` command, the initialization option file is opened in edit mode.

   ```
   vi /home/adbmanager/server/conf/adbinit.opt
   ```

3. Edit the page size of the DB area to be created.

   Press the **I** key to shift to insert mode. Once you are in insert mode, use the cursor keys to move the cursor to the following operand:

   ```
   adbinitdbarea -n ADBUIDX01 -p 8
   ```

   Once you have positioned the cursor, replace `8` in the `-p` option (which has been set as the default value) with `4`.

   ■ Input result

   ```
   adbinitdbarea -n ADBUIDX01 -p 4
   ```

4. Finish editing the initialization option file.

   For this environment, there is no need to edit operands other than the one described above. After you have entered the page size of the DB area to be created in edit mode, press **Esc** to shift to command mode. Once you are in command mode, enter the following command and press **Enter**:

   ```
   :wq
   ```

   The content that was specified in edit mode is saved. The information necessary for creating DB areas is now set up in the initialization option file.

(2) Related item

Format of initialization options in *Specification format for the adbinit command* under *adbinit (Initialize the Database)* in the manual *HADB Command Reference*

3.6.2 Creating a DB area

After you have edited the initialization option file, initialize the database and create DB areas.

> **Note**
>
> This section explains how to create DB areas by initializing the database. To initialize the database, you use the DB area creation command `adbinit` command). Use this command when you first create DB areas after installing the HADB server.
>
> You cannot use the DB area creation command (`adbinit` command) to add or modify DB areas after the database has been initialized. In this situation, you use the command for adding and changing DB areas (`adbmodarea` command). For details about the command for adding and changing DB areas (`adbmodarea` command), see *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

(1) Executing the DB area creation command

As the HADB administrator (`adbmanager`), enter the HADB command below and press **Enter**. A DB area is automatically created.

```
adbinit -u ADBUSER01 -p '#HelloHADB_01' /home/adbmanager/server/conf/adbinit.opt /hom
e/adbmanager/db
```

Note

If the message `bash: adbinit: command not found` is output during execution of the `adbinit` command, the environment variables that have been set might be invalid. Verify that the environment variables are set correctly based on the explanation in (2) Verifying the environment variables that were set in the .bashrc file under 3.5.1 Setting environment variables.

The following describes the options that are specified for the `adbinit` command:

- `-u ADBUSER01`

  Specify the HADB user's user ID (authorization identifier). When a DB area is created, the HADB user specified in the `-u` option is also created. Here, specify `ADBUSER01` as the HADB user's user ID (authorization identifier).

  The HADB user (`ADBUSER01`) created here is used to create the HADB user necessary for creating and retrieving tables.

  Note

  To explain how to create an HADB user, this chapter provides the steps for user `ADBUSER01` to create a new HADB user (`ADBUSER02`) needed to create and retrieve tables. For details, see 3.7 Creating an HADB user for creating and retrieving tables.

- `-p '#HelloHADB_01'`

  Specify a password for the user ID (`ADBUSER01`) specified in the `-u` option. Here, specify `#HelloHADB_01` as the password.

- `/home/adbmanager/server/conf/adbinit.opt`

  Specifies the absolute path to the location of the initialization option file.

- `/home/adbmanager/db`

  Specifies the absolute path to the DB directory in which the DB areas that are created will be stored. Note that the DB directory name specified here must be the same as the DB directory name specified in the `adb_db_path` operand of the server definition.

Messages are output when DB areas are created. The following shows an example of the output messages.

■ Example of the output messages

```
KFAA90000-I adbinit processing started.
KFAA96201-I Database initialization started.
KFAA96204-I The DB directory "/home/adbmanager/db" is initialized.
KFAA96233-I Initialization of file "ADBUIDX01" is complete. (size = 5456 KB, infor
mation = 7fd0dc15-f7b8-45ca-a178-1a1292c80fcc)
KFAA96233-I Initialization of file "ADBUTBL01" is complete. (size = 5440 KB, infor
mation = df7978f1-9f7a-4bcf-a477-61056297a71c)
KFAA96202-I Database initialization ended. (return code = 0)
KFAA90001-I adbinit processing ended. (return code = 0)
```

(2) Confirming the created DB areas

To verify that DB areas were created by the `adbinit` command, enter the following operating system command and press **Enter**:

```
ls /home/adbmanager/db
```

Executing the `ls` command, you can verify that the directories and files have been created under the specified directory. The following shows an example of the execution result.

■ Execution result example

```
ADBDIC  ADBMST  ADBSTBL  ADBSYS  ADBUIDX01  ADBUTBL01  ADBWORK  ADBWRK  SPOOL
```

Once you have confirmed that DB areas exist, creation of DB areas is complete.

(3) Related item

■ When creating DB areas by initializing the database

The related items when using the DB area creation command (`adbinit` command) are as follows:

- *adbinit (Initialize the Database)* in the manual *HADB Command Reference*
- 9.2 Initializing a database (creating data DB areas)

■ When adding or modifying DB areas after initializing the database

The related items when using the command for adding and changing DB areas (`adbmodarea` command) are as follows:

- *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*
- 11.10 Handling data DB areas

3.7 Creating an HADB user for creating and retrieving tables

After you have created a DB area by initializing a database, create an HADB user for creating and retrieving tables.

This section explains how to use the HADB user (ADBUSER01) created during DB area creation to create an HADB user (ADBUSER02) necessary for creating and retrieving tables. It also explains how to grant privileges to the created HADB user (ADBUSER02).

Because ADBUSER01 has two privileges (the DBA privilege and the CONNECT privilege), it can create HADB users and grant privileges.

When creating an HADB user and granting privileges, the OS user uses the *HADB administrator (adbmanager)*.

> **Note**
>
> To explain how to create an HADB user, this chapter provides the steps for user ADBUSER01 to create a new HADB user (ADBUSER02) needed to create and retrieve tables.
>
> It is also possible to grant the privileges needed to create and retrieve tables (schema definition privileges) to ADBUSER01 instead of creating a new HADB user (ADBUSER02).
>
> For details about privileges, see 2.7 Privileges.

3.7.1 Starting the HADB server

To create an HADB user by executing an SQL statement, start the HADB server.

(1) Executing the command to start the HADB server

As the HADB administrator (adbmanager), enter the HADB command below and press **Enter**. This starts the HADB server.

```
adbstart
```

Messages are output when the HADB server starts. The following shows an example of the output messages.

■ Example of the output messages

```
KFAA90000-I adbstart processing started.
KFAA91105-I The HADB system was started normally. (HADB server operation mode = "N
ORMAL")
KFAA90001-I adbstart processing ended. (return code = 0)
```

(2) Related item

- 10.2.1 Starting the HADB server
- *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*

3.7.2 Connecting to the HADB server

After the HADB server has started, use the client program for creating HADB users (`adbsql` command) to connect to the HADB server.

(1) Executing the connection command

1. Execute the `adbsql` command.

   As the HADB administrator (`adbmanager`), enter the following HADB command and press **Enter**:

   ```
   adbsql
   ```

2. Enter a user ID (authorization identifier) in response to the system request.

   When you execute the `adbsql` command, the system asks you to enter an HADB user's user ID (authorization identifier). The user ID (authorization identifier) specified here is used to connect to the HADB server. In this case, specify `ADBUSER01` and press **Enter**.

   ```
     USER-ID?
   ADBUSER01
   ```

3. Enter a password in response to the system request.

   When you enter a user ID (authorization identifier), the system asks you to enter a password. In this case, specify `#HelloHADB_01` and press **Enter**.

   ```
     PASSWORD?
   #HelloHADB_01    The entered password is not displayed.
   ```

(2) Confirming that the client program has connected to the HADB server

When you specify a password and press **Enter**, the client program connects to the HADB server, allowing you to enter SQL statements. An example of the execution result follows.

■ Execution result example

```
COMMAND ?   +----2----+----3----+----4----+----5----+----6----+----7----+
```

If the above execution result is displayed, you can enter SQL statements.

> **Important**
>
> If the entered HADB user's user ID (authorization identifier) or password is invalid, the attempt to connect to the HADB server fails and the `KFAA30561-E` message is output. If this occurs, see 3.12.4 The KFAA30561-E message is output while attempting to connect to the HADB server.

(3) Related item

adbsql (Execute SQL Statements) in the manual *HADB Command Reference*

3.7.3 Creating an HADB user

Once you are able to enter SQL statements, create the HADB user necessary for creating and retrieving tables.

(1) Executing the SQL statement for creating an HADB user

To create an HADB user, you use the CREATE USER definition SQL statement. Here, the following HADB user is created as the HADB user for creating and retrieving tables:

- User ID (authorization identifier): ADBUSER02
- Password: #HelloHADB_02

Enter the SQL statement described below and press **Enter**.

Make sure that the last character you enter is a semicolon (;), or the SQL statement will not execute.

```
CREATE USER "ADBUSER02" IDENTIFIED BY '#HelloHADB_02';
```

This SQL statement creates the HADB user (ADBUSER02). The created user ADBUSER02 does not have the two privileges (the CONNECT privilege and the schema definition privilege) necessary for creating and retrieving tables. Therefore, you must grant the necessary privileges.

(2) Confirming the HADB user creation result

The message shown below is output when the HADB user (ADBUSER02) is created. An example of the output message follows.

■ Example of the output message

```
KFAA96403-I SQL processing completed.
```

After you have confirmed that the HADB user has been created, HADB user creation is complete.

(3) Related item

9.4 Creating an HADB user to define base tables

3.7.4 Granting privileges to the created HADB user

This subsection explains how to grant to the HADB user (ADBUSER02) the privileges necessary for creating and retrieving tables.

(1) Executing the SQL statement for granting privileges

To grant privileges to an HADB user, you use the GRANT definition SQL statement. Here, the following two privileges are granted to the HADB user (ADBUSER02) as the privileges necessary for creating and retrieving tables:

- CONNECT privilege
- Schema definition privilege

Enter the SQL statement described below and press **Enter**.

Make sure that the last character you enter is a semicolon (;), or the SQL statement will not execute.

```
GRANT CONNECT,SCHEMA TO "ADBUSER02";
```

The CONNECT privilege and the schema definition privilege have now been granted to the HADB user (ADBUSER02).

(2) Confirming the results of granting privileges

The message shown below is output when privileges have been granted to the HADB user (ADBUSER02). An example of the output message follows.

■ Example of the output messages

```
KFAA96403-I SQL processing completed.
```

After you have confirmed that privileges have been granted, granting of privileges is complete.

(3) Terminating the connection to the HADB server

After you have created an HADB user and granted privileges, terminate the adbsql command. Enter the adbsql subcommand shown below and press **Enter**.

Make sure that the last character you enter is a semicolon (;), or the SQL statement will not execute.

```
#EXIT;
```

The adbsql command terminates and other commands can now be entered. An example of the execution result follows.

■ Execution result example

```
[adbmanager@localhost ~]$
```

(4) Related item

9.4 Creating an HADB user to define base tables

3.8 Creating tables

After you have created the HADB user (ADBUSER02) for creating and retrieving tables, you need to create a table to store the sample data.

3.8.1 Connecting to the HADB server

After the HADB server has started, use the client program for creating tables (adbsql command) to connect to the HADB server.

(1) Executing the connection command

1. Execute the adbsql command.

   As the HADB administrator (adbmanager), enter the following HADB command and press **Enter**:

   ```
   adbsql
   ```

2. Enter a user ID (authorization identifier).

   When you execute the adbsql command, the system asks you to enter the HADB user's user ID (authorization identifier). This user ID (authorization identifier) is used to connect to the HADB server. Here, specify ADBUSER02 and press **Enter**.

   ```
     USER-ID?
   ADBUSER02
   ```

3. Enter a password.

   When you enter a user ID (authorization identifier), the system asks you to enter a password. Here, specify #HelloHADB_02 and press **Enter**.

   ```
     PASSWORD?
   #HelloHADB_02    The entered password is not displayed.
   ```

(2) Confirming that the client program has connected to the HADB server

When you specify a password and press **Enter**, the client program connects to the HADB server, allowing you to enter SQL statements. The following shows an example of the execution result.

■ Execution result example

```
COMMAND ?   +----2----+----3----+----4----+----5----+----6----+----7----+
```

If the above execution result is displayed, you can enter SQL statements.

(3) Related item

adbsql (Execute SQL Statements) in the manual *HADB Command Reference*

3.8.2 Defining a schema

Once you are able to enter SQL statements, define the schemas. Schemas are a logical concept that includes tables. Unless a schema is defined, you cannot create tables.

(1) Executing an SQL statement for defining a schema

To define a schema, use the CREATE SCHEMA definition SQL statement. Enter the SQL statement described below and press **Enter**.

Be sure that a semicolon (;) is the last character you enter, or the SQL statement will not be executed.

```
CREATE SCHEMA ADBUSER02;
```

This defines a schema for the user ADBUSER02. The name of the schema defined here must be the same as the user ID (authorization identifier) specified in the adbsql command.

(2) Confirming the schema definition result

A message is displayed when the schema ADBUSER02 is defined. The following shows an example of the output message.

■ Example of the output message

```
KFAA96403-I SQL processing completed.
```

When you have confirmed that a schema exists, schema definition is complete.

(3) Related item

CREATE SCHEMA (define a schema) in *Definition SQL* in the manual *HADB SQL Reference*

3.8.3 Creating tables

After creating a schema, you will create a table with the name SAMPLE for storing the sample data.

> **Note**
>
> The authorization identifier (SAMPLE) for the sample application program ($ADBDIR/sample) and the authorization identifier explained in this section (ADBUSER02) are different. If using the sample application program, be sure to change the authorization identifier first.

(1) Executing an SQL statement for creating a table

To create a table, use the CREATE TABLE definition SQL statement. Enter the following SQL statement and press **Enter**:

```
CREATE TABLE "SAMPLE" ("STATECODE" SMALLINT, "STATENAME" VARCHAR(15),
```

NEXT ? will then be displayed. Continue by entering the following SQL statement and pressing **Enter**:

```
"ZIPCODE" CHAR(15), "ADDRESS" VARCHAR(100), "AREA" DECIMAL(19))
```

NEXT ? will then be displayed. Continue by entering the following SQL statement and pressing **Enter**:

```
IN ADBUTBL01;
```

The following table shows the details of the CREATE TABLE statement entered.

Table 3-6: Details of the CREATE TABLE statement specified

No.	Input content	Description	
1	"SAMPLE"	Specifies the name of the table to be created.	
2	"STATECODE" SMALLINT	Specifies column names and data types for the table.	Specifies STATECODE (state code) as the column name. Since a number between 1 and 50 is to be stored in the STATECODE column, SMALLINT is specified as the data type.
3	"STATENAME" VARCHAR(15)		Specifies STATENAME (state name) as the column name. Since a variable-length character string of from 1 to 15 bytes is to be stored in the STATENAME column, VARCHAR is specified as the data type.
4	"ZIPCODE" CHAR(15)		Specifies ZIPCODE (postal zip code) as the column name. Since a 15-byte, fixed-length character string is to be stored in the ZIPCODE column, CHAR is specified as the data type.
5	"ADDRESS" VARCHAR(100)		Specifies ADDRESS (address of state capitol building) as the column name. Since a variable-length character string of from 1 to 100 bytes is to be stored in the ADDRESS column, VARCHAR is specified as the data type.
6	"AREA" DECIMAL(19)		Specifies AREA (area size) as the column name. Since a 19-digit number is to be stored in the AREA column, DECIMAL is specified as the data type.
7	IN ADBUTBL01	Specifies the name of the DB area for storing the table.	

Note

You can also create a table using an SQL statement stored in a file. If the length of the SQL statement (CREATE TABLE statement) makes it difficult to enter the adbsql command without making a mistake, follow the procedure in (3) Ending the connection to the HADB server, and then see 3.12.2 I cannot create a table.

(2) Confirming the result of table creation

A message is output when the table SAMPLE is created. The following shows an example of the output message.

■ Example of the output message

```
KFAA96403-I SQL processing completed.
```

When you have confirmed that the table exists, table creation is complete.

(3) Ending the connection to the HADB server

Once you have created the table, terminate the `adbsql` command. Enter the following `adbsql` subcommand and press **Enter**:

Make sure that the last character you enter is a semicolon (`;`), or the SQL statement will not execute.

```
#EXIT;
```

The `adbsql` command is terminated and another command can now be entered. The following shows an example of the execution result.

■ Execution result example

```
[adbmanager@localhost ~]$
```

(4) Related item

CREATE TABLE (define a table) in *Definition SQL* in the manual *HADB SQL Reference*

3.9 Storing data

Once the table SAMPLE is created, store the sample data in the table.

To store the sample data in the table, execute the adbimport command. When doing so, it is best to specify a text file (*input data path file*) that points to the location where the sample data is stored, rather than specifying the sample data itself. Therefore, before storing the data, you must create an input data path file.

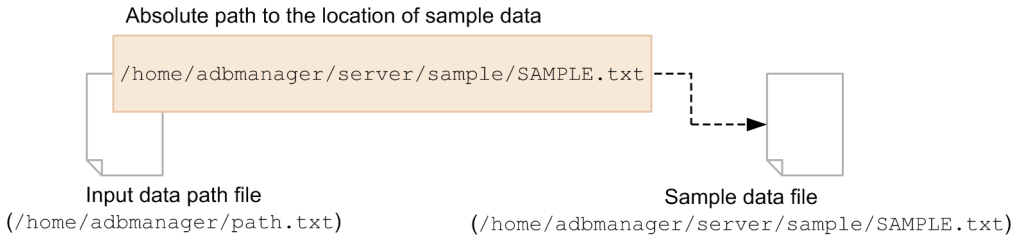
This section explains how to create an input data path file and how to store data in the table.

3.9.1 Preparation for storing data

In preparation for storing data in the table, create a text file (*input data path file*) that points to the location where the sample data is stored.

The following figure shows the details of an input data path file.

Figure 3-4: Details of an input data path file



This subsection explains how to use the vi command to create an input data path file.

(1) Creating an input data path file

1. Create an input data path file.

   As the HADB administrator (adbmanager), enter the operating system command below and press **Enter**. This creates a file with the file name path.txt under the home directory of adbmanager.

   ```
   vi /home/adbmanager/path.txt
   ```

2. Enter the sample data storage location.

   Press the **I** key, and then enter the absolute path shown below. This absolute path identifies the sample data storage location.

   ```
   /home/adbmanager/server/sample/SAMPLE.txt
   ```

3. Finish creating an input data path file.

   Press **Esc**, enter the following command, and then press **Enter**:

   ```
   :wq
   ```

   An input data path file has now been created.

(2) Confirming the input data path file that has been created

To check whether an input data path file was created by the `vi` command, enter the following operating system command and press **Enter**:

```
ls /home/adbmanager/path.txt
```

Using the `ls` command, you can verify that the specified file was created. The following shows an example of the execution result.

■ Execution result example

```
/home/adbmanager/path.txt
```

When you have confirmed that an input data path file exists, creation of the input data path file is complete.

(3) Related item

adbimport (Import Data) in the manual *HADB Command Reference*

3.9.2 Storing data in the table

Once the input data path file has been created, store the sample data in the table.

When storing the sample data, use the *input data path file* that you just created and the *import option file*.

A file template is available for the import option file. Therefore, copy and use the template. The template is stored in the following location:

- Import option file template: `/home/adbmanager/server/sample/conf/adbimport.opt`

The created input data path file and the import option file created by copying the template are stored in the following locations:

- Input data path file: `/home/adbmanager/path.txt`
- Import option file: `/home/adbmanager/server/conf/`
  Use the import option file with its default settings.

(1) Copying the import option file template

To store the sample data in a table, copy the import option file template.

1. Copy the import option file template.
   Enter the operating system command below and press **Enter**. The import option file template is copied.
   ```
   cp /home/adbmanager/server/sample/conf/adbimport.opt /home/adbmanager/server/conf/
   ```
2. Check the copied import option file template.
   Enter the operating system command below and press **Enter**. This checks whether the import option file template has been copied.
   ```
   ls /home/adbmanager/server/conf/
   ```

■ Execution result example

```
adbimport.opt  adbinit.opt  client.def  server.def
```

If the import option file template has been copied, executing the `ls` command displays `adbimport.opt`.

(2) Executing the command for storing data

As the HADB administrator (`adbmanager`), enter the HADB command below and press **Enter**.

In the command example below, a line feed has been entered before `SAMPLE` to improve readability. When entering the actual command, a line feed is not necessary. Enter a space after `adbimport.opt` and continue by entering `SAMPLE` and the information following it.

```
adbimport -u ADBUSER02 -p '#HelloHADB_02' -z /home/adbmanager/server/conf/adbimport.o
pt
          SAMPLE /home/adbmanager/path.txt
```

Note

The `adbimport` command can be executed only when the HADB server is running. If the `KFAA90004-E` message is output, start the HADB server by following the explanation in (1) Executing the command to start the HADB server under 3.7.1 Starting the HADB server.

The following table shows the options that are specified for the `adbimport` command.

Table 3-7: Details of the options specified for the `adbimport` command

No.	Option	Description
1	`-u ADBUSER02`	Specifies the user ID (authorization identifier) of the HADB user who is executing the `adbimport` command. Here, `ADBUSER02` is specified.
2	`-p '#HelloHADB_02'`	Specifies the password of the HADB user specified in the `-u` option. Here, `#HelloHADB_02` is specified.
3	`-z /home/adbmanager/server/conf/adbimport.opt`	Specifies the import option file if you are specifying options when storing data. When storing the sample data, you must specify the import option file set to the default contents. Otherwise, an error will occur.
4	`SAMPLE`	Specifies the name of the table in which the data is to be stored. Since the sample data is to be stored in the table `SAMPLE`, specify `SAMPLE`.
5	`/home/adbmanager/path.txt`	Specifies the absolute path to the input data path file.

(3) Confirming the data storage result

Messages are displayed when the `adbimport` command is executed. The following shows an example of the displayed results.

■ Example of the displayed results

```
KFAA90000-I adbimport processing started.
KFAA80202-I Import processing started. The table is "ADBUSER02"."SAMPLE".
KFAA80203-I 50 rows loaded.
```

```
KFAA80204-I Import processing ended. (return code = 0)
KFAA90001-I adbimport processing ended. (return code = 0)
```

The sample data has now been stored in the table.

(4) Related item

adbimport (Import Data) in the manual *HADB Command Reference*

3.10 Retrieving the stored data

Once the sample data has been loaded into the table, you can search through it. This section explains how to connect to the HADB server and execute the SQL statement (`SELECT` statement) to retrieve the stored data.

Retrieving the stored data is performed by the *HADB administrator (adbmanager)*.

3.10.1 Connecting to the HADB server and retrieving data

(1) Executing the connection command

1. Execute the `adbsql` command.

   As the HADB administrator (`adbmanager`), enter the following HADB command and press **Enter**:

```
adbsql
```

2. Enter a user ID (authorization identifier) in response to the system request.

   Here, enter `ADBUSER02` and press **Enter**.

```
  USER-ID?
ADBUSER02
```

3. Enter a password in response to the system request.

   Here, specify `#HelloHADB_02` and press **Enter**.

```
  PASSWORD?
#HelloHADB_02    The entered password is not displayed.
```

When you specify a password and press **Enter**, the client program connects to the HADB server, allowing you to enter SQL statements. The following shows an example of the execution result.

■ Execution result example

```
COMMAND ?   +----2----+----3----+----4----+----5----+----6----+----7----+
```

If the above execution result is displayed, you can enter SQL statements.

(2) Retrieving all of the stored data

Enter the SQL statement (`SELECT` statement) to retrieve all rows of the table storing the sample data and press **Enter**.

Make sure that the last character you enter is a semicolon (`;`), or the SQL statement will not execute.

```
SELECT * FROM "SAMPLE";
```

Retrieving all rows of the table outputs the state codes, state names, ZIP codes, addresses of state capitol buildings, and area sizes of the 50 U.S. states.

A message is output when the retrieval is completed. The following shows an example of the output message.

■ Example of the output message

```
KFAA96404-I 50 rows were selected.
```

(3) Terminating the connection to the HADB server

Enter the `adbsql` subcommand shown below and press **Enter**. This terminates the `adbsql` command.

Make sure that the last character you enter is a semicolon (;), or the SQL statement will not execute.

```
#EXIT;
```

The `adbsql` command terminates and another command can now be entered. The following shows an example of the execution result.

■ Execution result example

```
[adbmanager@localhost ~]$
```

(4) Terminating the HADB server

Enter the HADB command shown below and press **Enter**. This stops the HADB server.

```
adbstop
```

When the `adbstop` command is executed, messages are output and the HADB server stops. The following shows an example of the output messages.

■ **Example of the output messages**

```
KFAA90000-I adbstop processing started.
KFAA91154-I The HADB system was terminated normally.
KFAA90001-I adbstop processing ended. (return code = 0)
```

This completes the trial operation of an HADB server using the hands-on environment. If you no longer need the hands-on environment, uninstall the HADB server in the hands-on environment by following the procedure in 3.11 Uninstalling the HADB server.

> **Important**
>
> Note that the hands-on environment that was built according to the instructions in this chapter is used specifically for gaining first-hand experience in installing and operating an HADB server. It is not an environment that can be used in actual operations. Therefore, if you execute commands or SQL statements not explained in this chapter in the hands-on environment, an error might occur and the HADB server might not operate normally.
>
> It you want to build an environment for use in actual operations, first estimate the required values based on the explanation in *Part 3. Design* of this manual, and then build your environment based on the explanation in *Part 4. Setup* of this manual.

(5) Related items

- *SELECT Statement Examples* in the manual *HADB SQL Reference*

- *adbsql subcommands* under *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*
- 10.2.2 Terminating the HADB server
- *adbstop (Terminate the HADB Server)* in the manual *HADB Command Reference*

3.11 Uninstalling the HADB server

This section explains how to uninstall the HADB server. When you no longer need the HADB server hands-on environment that was built, uninstall it using the procedures provided below.

You can uninstall the HADB server either as a superuser or as the HADB administrator (`adbmanager`).

Important

Before uninstalling the HADB server, make sure it is stopped. The HADB server cannot be uninstalled while it is active.

3.11.1 Deleting the server directory

To uninstall the HADB server, you must delete the server directory.

(1) Executing the server directory deletion command

As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. This deletes the server directory.

```
rm -rf /home/adbmanager/server
```

(2) Checking that the server directory has been deleted

As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. This enables you to check that the server directory has been deleted correctly.

```
ls -d /home/adbmanager/server
```

■ Execution result example

```
ls: cannot access /home/adbmanager/server: No such file or directory
```

If the execution result shown above is displayed, the server directory has been deleted.

3.11.2 Deleting the directory that stores the installation data

Delete the directory that stores the HADB server installation data.

(1) Executing the command to delete the directory that stores the installation data

As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. The directory that stores the installation data is deleted.

```
rm -rf /home/adbmanager/install
```

(2) Confirming that the directory that stores the installation data has been deleted

As the HADB administrator (`adbmanager`), enter the operating system command below and press **Enter**. You can check whether the directory that stores the installation data has been deleted correctly.

```
ls -d /home/adbmanager/install
```

■ Execution result example

```
ls: cannot access /home/adbmanager/install/: No such file or directory
```

If the execution result shown above is displayed, the directory that stores the installation data has been deleted.

3.11.3 Deleting the directories that store communication-information files

Once you have uninstalled the HADB server, delete the directories that store communication-information files.

(1) Executing the command for deleting the directories that store communication-information files

The communication-information file directories to be deleted differ depending on the OS version of the server machine.

- If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  See (a) Red Hat Enterprise Linux Server 6.
- If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  See (b) Red Hat Enterprise Linux Server 7.

(a) Red Hat Enterprise Linux Server 6

Delete the following directories:

- `/dev/HADB/pth`
- `/lib/udev/devices/HADB/pth`

1. Delete the `/dev/HADB/pth` directory.
   As a superuser, enter the following operating system command, and then press the **Enter** key. The `/dev/HADB/pth` directory is deleted.

   ```
   rm -rf /dev/HADB/pth
   ```

2. Delete the `/lib/udev/devices/HADB/pth` directory.
   As a superuser, enter the following operating system command, and then press the **Enter** key. The `/lib/udev/devices/HADB/pth` directory is deleted.

   ```
   rm -rf /lib/udev/devices/HADB/pth
   ```

(b) Red Hat Enterprise Linux Server 7

Delete the following directory and file:

- /dev/HADB/pth
- /etc/tmpfiles.d/dev-HADB-pth.conf

1. Delete the /dev/HADB/pth directory.

   As a superuser, enter the following operating system command, and then press the **Enter** key. The /dev/HADB/pth directory is deleted.

   ```
   rm -rf /dev/HADB/pth
   ```

2. Delete the /etc/tmpfiles.d/dev-HADB-pth.conf file.

   As a superuser, enter the following operating system command, and then press the **Enter** key. The /etc/tmpfiles.d/dev-HADB-pth.conf file is deleted.

   ```
   rm -f /etc/tmpfiles.d/dev-HADB-pth.conf
   ```

(2) Checking that the directories for storing communication-information files have been deleted

Check that the directories for storing communication-information files have been deleted correctly.

The items to be checked differ depending on the OS version of the server machine.

- If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  See (a) Red Hat Enterprise Linux Server 6.
- If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  See (b) Red Hat Enterprise Linux Server 7.

(a) Red Hat Enterprise Linux Server 6

1. Check whether the /dev/HADB/pth directory has been deleted.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -d -l /dev/HADB/pth
   ```

   From the execution result of the ls command, you can check whether the /dev/HADB/pth directory has been deleted. The following shows an example of the execution result.

   ▪ **Execution result example**

   ```
   ls: cannot access /dev/HADB/pth: No such file or directory
   ```

   If the execution result such as the preceding one is displayed, the /dev/HADB/pth directory has been deleted.

2. Check whether the /lib/udev/devices/HADB/pth directory has been deleted.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -d -l /lib/udev/devices/HADB/pth
   ```

   From the execution result of the ls command, you can check whether the /lib/udev/devices/HADB/pth directory has been deleted. The following shows an example of the execution result.

   ▪ **Execution result example**

   ```
   ls: cannot access /lib/udev/devices/HADB/pth: No such file or directory
   ```

If the execution result such as the preceding one is displayed, the `/lib/udev/devices/HADB/pth` directory has been deleted.

(b) Red Hat Enterprise Linux Server 7

1. Check whether the `/dev/HADB/pth` directory has been deleted.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -d -l /dev/HADB/pth
   ```

   From the execution result of the `ls` command, you can check whether the `/dev/HADB/pth` directory has been deleted. The following shows an example of the execution result.

   ▪ **Execution result example**

   ```
   ls: cannot access /dev/HADB/pth: No such file or directory
   ```

   If the execution result such as the preceding one is displayed, the `/dev/HADB/pth` directory has been deleted.

2. Check whether the `/etc/tmpfiles.d/dev-HADB-pth.conf` file has been deleted.

   As a superuser, enter the following operating system command, and then press **Enter**:

   ```
   ls -l /etc/tmpfiles.d/dev-HADB-pth.conf
   ```

   From the execution result of the `ls` command, you can check whether the `/etc/tmpfiles.d/dev-HADB-pth.conf` file has been deleted. The following shows an example of the execution result.

   ▪ **Execution result example**

   ```
   ls: cannot access /etc/tmpfiles.d/dev-HADB-pth.conf: No such file or directory
   ```

   If the execution result such as the preceding one is displayed, the `/etc/tmpfiles.d/dev-HADB-pth.conf` file has been deleted.

3.11.4 Deleting the HADB administrator and HADB administrators group

During the process of installing the HADB server, the OS user who was to manage HADB (*HADB administrator*) and the group to which the HADB administrator belongs (*HADB administrators group*) were set up on the OS. This subsection explains how to delete these.

The following HADB administrator and HADB administrators group must be deleted:

- HADB administrator: `adbmanager`
- HADB administrators group: `adbgroup`

(1) Deleting the HADB administrator

Enter the following operating system command and press **Enter**:

```
userdel -r adbmanager
```

The following describes the details of the options for the `userdel` command:

- `-r`

When this option is specified, deleting a user also deletes the user's home directory. Therefore, the home directory (`/home/adbmanager/`) of `adbmanager` will be deleted.

- `adbmanager`
  Specifies the name of the user to be deleted.

The HADB administrator has now been deleted.

(2) Deleting the HADB administrators group

As a superuser, enter the following operating system command and press **Enter**:

```
groupdel adbgroup
```

The following describes the details of the option for the `groupdel` command:

- `adbgroup`
  Specifies the name of the group to be deleted.

The HADB administrators group has now been deleted.

3.12 Frequently asked questions and corrective actions

This section covers some questions that are frequently asked when building a hands-on HADB server environment, along with the appropriate corrective action to take.

3.12.1 I created the DB directory as a superuser by mistake

If you created the DB directory (`/home/adbmanager/db`) as a superuser instead of as the HADB administrator (`adbmanager`), delete the DB directory that was created.

To delete the DB directory:

1. Log on as a superuser.
   If you are logged on as the HADB administrator (`adbmanager`), log off, and then log on again as a superuser.
2. Delete the DB directory.
   Enter the following operating system command and press **Enter**:

```
rmdir /home/adbmanager/db
```

You have now deleted the DB directory. Now log on as the HADB administrator (`adbmanager`) and create a new DB directory.

If you cannot delete the DB directory using the `rmdir` command, enter the following operating system command and press **Enter**:

```
rm -rf /home/adbmanager/db
```

3.12.2 I cannot create a table

If the SQL statement (`CREATE TABLE` statement) is long and you make a mistake in entering the `adbsql` command, try the procedure described below to create a table.

The following describes the procedure for executing an SQL statement saved in a file.

> **Note**
>
> You can also use other text editors to create the file (`infile`) that is being created in steps 1 through 3.

1. Create the file (`infile`) that is to contain the SQL statement.
   Enter the operating system command shown below and press **Enter**. Create the file under the home directory (`/home/adbmanager`) and use the file name `infile`.

```
vi /home/adbmanager/infile
```

2. Enter the SQL statement to be executed.
   Press the **I** key and enter the SQL statement shown below.

Enter ADBUSER02 in the first row and #HelloHADB_02 in the second row. Enter the CREATE TABLE statement beginning on the third row.

```
ADBUSER02
#HelloHADB_02
CREATE TABLE "SAMPLE" ("STATECODE" SMALLINT, "STATENAME" VARCHAR(15),
"ZIPCODE" CHAR(15), "ADDRESS" VARCHAR(100), "AREA" DECIMAL(19))
IN ADBUTBL01;
```

3. Finish creating the file (infile) containing the SQL statement.

   Press **Esc**, enter the following command, and then press **Enter**:

   ```
   :wq
   ```

   The file (infile) containing the SQL statement has now been created.

4. Execute the file (infile) containing the SQL statement.

   With the HADB server started, execute the following HADB command:

   ```
   adbsql -V < infile
   ```

   When the above command is executed, the SQL statement entered in step 2 is executed. The following shows an example of the execution result.

   ■ Execution result example

   ```
     USER-ID ?
   ADBUSER02

     PASSWORD ?
   #HelloHADB_02

   COMMAND ? +----2----+----3----+----4----+----5----+----6----+----7----+
   CREATE TABLE "SAMPLE" ("STATECODE" SMALLINT, "STATENAME" VARCHAR(15),
   NEXT ? +----2----+----3----+----4----+----5----+----6----+----7----+
   "ZIPCODE" CHAR(15), "ADDRESS" VARCHAR(100), "AREA" DECIMAL(19))
   NEXT ? +----2----+----3----+----4----+----5----+----6----+----7----+
   IN ADBUTBL01;
   KFAA96403-I SQL processing completed.
   ```

   A table has been created that contains this information.

3.12.3 The KFAA91104-Q message is output during startup of the HADB server

This subsection explains the action to take if the KFAA91104-Q message is output when the HADB server startup command (adbstart command) is executed.

The KFAA91104-Q message (asking whether you want to delete the troubleshooting information) is output if the adbstart command is executed after the OS was shut down without the HADB server termination command (adbstop command) being executed while the HADB server was running.

■ KFAA91104-Q message output example

```
KFAA90000-I adbstart processing started. KFAA91104-Q Are you sure you want to disc
ard trouble shoot information? (y/N)
```

The HADB server terminated abnormally because it was not terminated according to the correct procedure, and as a result, troubleshooting information has been output.

Because the troubleshooting information that was output is not necessary in the hands-on environment built in this chapter, delete it.

> **Important**
>
> If the KFAA91104-Q message is output during execution of the adbstart command in an environment used in actual operations instead of in a hands-on environment, check the appropriate location to collect troubleshooting information. For details, see the notes under *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

Click **Y** in response to the KFAA91104-Q message. Clicking **Y** deletes the troubleshooting information, and then the HADB server starts.

■ **Example of a message output after Y is clicked**

```
KFAA91105-I The HADB system was started normally. (HADB server operation mode = "N
ORMAL")
KFAA90001-I adbstart processing ended. (return code = 0)
```

If the return code is 0, the HADB server has started.

3.12.4 The KFAA30561-E message is output while attempting to connect to the HADB server

This subsection explains the action to take if the KFAA30561-E message is output after an invalid user ID (authorization identifier) or password was entered by mistake during execution of the HADB server connection command (adbsql command).

■ **KFAA30561-E message output example**

```
KFAA30561-E The specified authorization identifier or password is invalid.
```

In this case, the attempt to connect to the HADB server has failed.

If the attempt to connect to the HADB server failed, try to connect again. Enter the adbsql subcommand shown below and press **Enter**.

Make sure that the last character you enter is a semicolon (;). If not, the SQL statement will not execute.

```
#CONNECT;
```

When the system asks you to enter a user ID (authorization identifier) and password, enter the correct user ID (authorization identifier) and password. If the entries are valid, you can re-connect to the HADB server.

4 System Design

This chapter explains the flow of designing a system (database, resource, and server definition design) when building a DBMS.

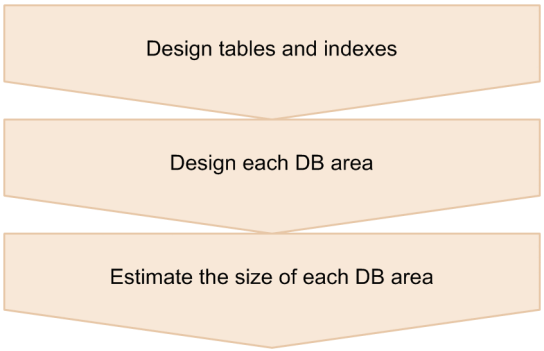
This chapter also explains how to estimate resource and memory requirements based on the amount of data to be stored in the HADB server (input data).

4.1 System design flow

This section provides figures that illustrate the following aspects of system design:

- Database design
- Resource design
- Server definition design

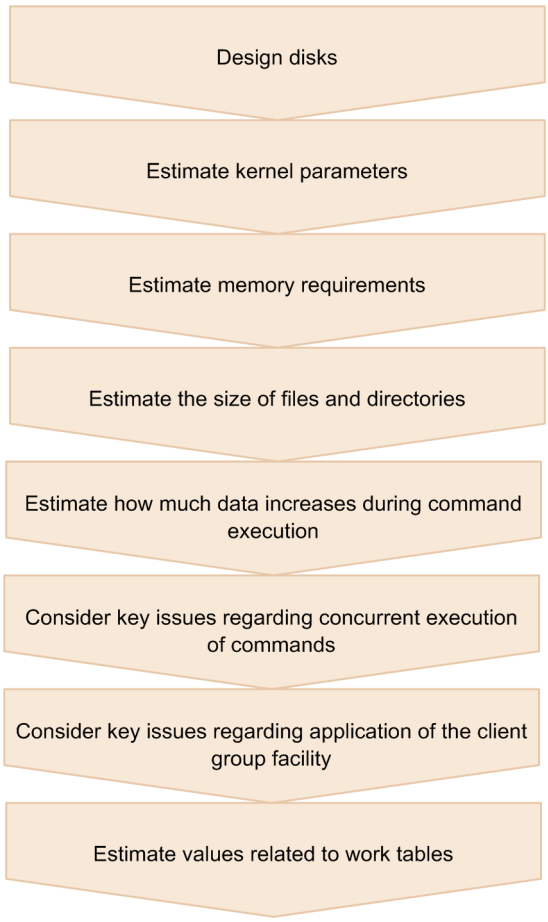
Figure 4-1: System design flow (database design)



For details about the database design to be performed in the system design stage, see 5. Designing a Database.

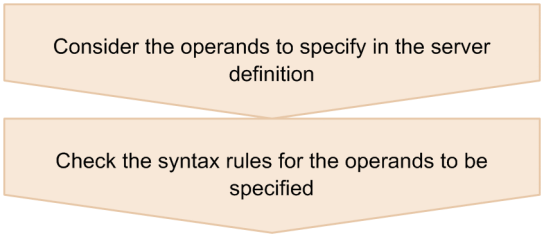
> **Note**
>
> Before designing a database, you can estimate its size and memory requirements based on the amount of data to be stored on the HADB server (input data). For details, see the following topics:
>
> - 4.2 Estimating resource requirements
> - 4.3 Estimating memory requirements

Figure 4-2: System design flow (resource design)



For details about the resource design to be performed in the system design stage, see 6. Preparing Resources.

Figure 4-3: System design flow (server definition design)



For details about the server definition design to be performed in the system design stage, see 7. Designing the Server Definition.

4.2 Estimating resource requirements

This section explains how to estimate the sizes of various resources based on the amount of data to be stored in the HADB server (input data). The resources whose sizes are to be estimated here are the files and directories that are contained in the following directories:

- DB directory
- Server directory
- Archive directory
- Unload file directory
- Synonym dictionary file directory
- Multi-node synonym dictionary storage directory

If you wish to estimate the approximate size of resources based on the input data before designing a database, use the formulas described in the subsections below. Note, however, that the values obtained are only rough estimates since specific sizes cannot be estimated.

If you want to estimate sizes more precisely, estimate the sizes after first designing the database as explained in 5. Designing a Database.

4.2.1 Estimating the size of the DB directory

The following table shows the formulas for estimating the sizes of directories and files in the DB directory based on input data. The sum total of the values estimated for rows 1 through 14 will be the total size of the DB directory.

Based on the size estimated here, use a disk that has a sufficient free space.

Table 4-1: Formula (DB directory size)

No.	Directory name and file name	Description	Formula (gigabytes)
1	$DBDIR/ADBMST	Master directory DB area file	0.01
2	$DBDIR/ADBDIC	Dictionary DB area file	0.5
3	$DBDIR/ADBSTBL	System-table DB area file	0.5
4	$DBDIR/ADBWRK	Work table DB area file#12	*org_data*#1 x 1#2
5	$DBDIR/DBAREA#3	Data DB area file for storing row store tables	*org_data*#1 x 2
6		Data DB area file for storing column store tables	*org_data*#1
7		Data DB area file for storing B-tree indexes	*org_data*#1 x 1#4
8		Data DB area file for storing text indexes	*org_data*#1 x 1.2#5
9		Data DB area file for storing range indexes	*org_data*#1 x 0.00002 + 0.2#6
10	$DBDIR/ADBSYS/ADBSLG	Directory for system log files	1#7, #8
11	$DBDIR/ADBSYS/ADBSTS	Directory for status files	
12	$DBDIR/ADBSYS/ADBUTL	Directory for the command status file	
13	$DBDIR/ADBWORK#9	Work directory#12	*db_idx* x 4#10

No.	Directory name and file name	Description	Formula (gigabytes)
14	`$DBDIR/SPOOL`[#11]	Error information (core file) output directory[#12]	2

Legend:

org_data: Size of the input data file (gigabytes)

db_idx: Value obtained for *Data DB area file for storing B-tree indexes* in row 7 (gigabytes)

#1

If the input data file has been compressed, substitute the size of the decompressed data.

#2

Because this value varies considerably depending on the results of the SQL statements that use work tables, a larger size than that estimated by this formula might be required.

#3

The name specified by the HADB administrator in the `adbinit` command.

#4

If you know the data types of the columns for which B-tree indexes are to be defined and the number of data items, using the following formula can yield a more precise estimate. Calculate only the number of B-tree indexes to be stored in the data DB area. Add the calculation result to the value obtained for *Data DB area file for storing B-tree indexes* in Table 4-1: Formula (DB directory size).

Formula (gigabytes)

$$db\_idx = \left\lceil \frac{(20 + KEYSZ) \times row\_num \times \frac{ENT\_NUM}{ENT\_NUM - 1}}{1{,}073{,}741{,}824} \right\rceil$$

Explanation of variables

- *KEYSZ*
  Size of the column for which a B-tree index is to be defined (bytes)
  For details, see 5.8.4 Determining the key length (KEYSZ) of a B-tree index.
- *row_num*
  Number of data items in the table in which B-tree indexes are defined
- *ENT_NUM*
  Use the following formula to calculate a value for this variable.
  Formula

$$ENT\_NUM = \mathrm{MAX}\left\{2, \left\lfloor 4{,}000 \div (20 + KEYSZ) \right\rfloor\right\}$$

#5

If notation-correction-search text-index specification or text-index-word-context search specification was specified when defining the text index, use the following formula to determine the size.

Formula (gigabytes)

- If notation-correction-search text-index specification was specified

$$org\_data \times 2.4$$

- If text-index-word-context search specification was specified

$org\_data \times 2.8$

- If notation-correction-search text-index specification and text-index-word-context search specification were both specified

$org\_data \times 4$

#6

If the target range index has been defined in a multi-chunk table, use the following formula to determine the size.

Formula (gigabytes)

$$\sum_{i=1}^{chunk\_num} (chunk\_data_{(i)} \times 0.00002 + 0.2)$$

Explanation of variables

- *chunk_num*
  Total number of chunks
- *chunk_data(i)*
  Amount of data in the i-th chunk (gigabytes)

#7

Because this value greatly depends on the volume of data handled by the following SQL statements and commands, a larger size than that estimated by the formula might be required depending on the data:

- Update SQL statement
- `adbimport` command
- `adbidxrebuild` command
- `adbmergechunk` command

#8

For rows 10 through 12 combined, use an estimate of 1 gigabyte.

#9

This directory is used when the `-w` option is not specified for the following commands:

- `adbimport` command
- `adbidxrebuild` command
- `adbmergechunk` command
- `adbunarchivechunk` command
- `adbreorgsystemdata` command

If the `-w` option is specified for the above commands, the specified directory is used as the work directory.

#10

Used only when executing the following commands:

- `adbimport` command
- `adbidxrebuild` command
- `adbmergechunk` command
- `adbunarchivechunk` command

- adbreorgsystemdata command

Because the value greatly depends on the data types of the columns for which B-tree indexes are defined and the number of B-tree indexes defined, a larger size than that estimated by the formula might be required depending on the data.

If a text index is defined, use the following formula to determine the size.

Formula (gigabytes)

org_data × 5

#11

The error information (core file) is output when the adb_core_path operand in the server definition is not specified.

#12

When using the multi-node function, the size of this directory or file must be estimated for all nodes.

4.2.2 Estimating the size of the server directory

The table below shows the estimate sizes of the files under the server directory based on the input data. The sum total of the values estimated for rows from row 1 to row 6 will be the total size of the server directory.

Table 4-2: Estimated values (server directory size)

No.	Directory name and file name	Description	Estimated value (gigabytes)
1	$ADBDIR/spool/adbmessage*XX*.log[#1]	Server message log file[#5]	0.064
2	$ADBDIR/spool/adbstatlog*XX*[#1, #2]	Statistics log file[#5]	16
3	$ADBDIR/spool/adbdump*YYYYMMDDhhmmss*.*server-process-process-ID*[#3]	HADB dump file[#5]	100
4	$ADBDIR/spool/adbsqltrc*XX*.log[#4]	SQL trace file[#5]	2
5	$ADBDIR/spool/adbcolumnize01	Log file used by the updated-row columnizing facility[#5]	0.1
6	$ADBDIR/spool/adboptlog*XX*[#6]	Access path search information log file[#5]	0.2

#1

XX in the file name is a sequential number between 01 and 04.

#2

If the adb_sta_log_path operand is specified in the server definition, data is output to the directory specified for the adb_sta_log_path operand in the server definition.

For details about the adb_sta_log_path operand in the server definition, see the description of the adb_sta_log_path operand in 7.2.7 Operands related to statistical information (set format).

#3

YYYYMMDDhhmmss in the file name indicates the time when the file was generated.

#4

XX in the file name is a sequential number between 01 and 08.

#5

When using the multi-node function, the size of this file must be estimated for all nodes.

#6

XX in the file name is a sequential number that is either `01` or `02`.

4.2.3 Estimating the size of the archive directory

The following shows the calculation formula for estimating the size of the archive files that will be stored in the archive directory. The estimated value will be the total size of the archive directory.

Use the following calculation formula if you define an archivable multi-chunk table and use the `adbarchivechunk` command to archive the data in the chunk.

Formula (gigabytes)

$$\text{Estimated size of archive directory} = \sum_{i=1}^{archive\_chunk\_num} (archive\_chunk\_data(i) \times 0.5)^{\#}$$

#

The compression rate of archived data differs depending on the data stored in the archivable multi-chunk table. Therefore, more capacity than estimated by using the formula might be required.

Explanation of the variables

archive_chunk_num

The number of chunks to be archived

archive_chunk_data(i)

The size of the original data that is stored in the i-th chunk to be archived (gigabytes)

4.2.4 Estimating the size of the unload file directory

The following shows the estimated size of unload files that will be stored in the unload file directory.

An unload file is created when a system table is reorganized by using the `adbreorgsystemdata` command.

Value (gigabytes)

$$0.5^{\#}$$

#

The actual size of an unload file varies according to the size of the system table to be reorganized.

4.2.5 Estimating the size of the synonym dictionary file directory

The following shows the calculation formula for estimating the size of the synonym dictionary files that will be stored in the synonym dictionary file directory. The estimated value will be the total size of the synonym dictionary file directory.

Before you use the `adbsyndict` command to create synonym dictionaries, use the following formula to estimate the size.

Formula (gigabytes)

$$\text{Estimated size of the synonym dictionary file directory} = \sum_{i=1}^{syndict\_num} (syndict\_data(i) \times 24)$$

Explanation of the variables

syndict_num
Number of synonym dictionaries that will be created

syndict_data(i)
Size of the synonym list definition file of each synonym dictionary (gigabytes)

> **Note**
>
> The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

4.2.6 Estimating the size of the multi-node synonym dictionary storage directory

The formula for estimating the size of the multi-node synonym dictionary storage directory is the same as the formula in 4.2.5 Estimating the size of the synonym dictionary file directory. The estimated value will be the total size of the multi-node synonym dictionary storage directory.

You need to estimate the size of the multi-node synonym dictionary storage directory for all nodes.

> **Note**
>
> The multi-node synonym dictionary storage directory is the directory specified for the `adb_syndict_node_storage_path` operand in the server definition.

4.3 Estimating memory requirements

This section explains how to estimate the memory requirements based on the values obtained in 4.2 Estimating resource requirements. The memory used by the HADB server consists of *shared memory* and *process memory*. There are four types of shared memory:

- Shared memory management area
- Global buffer page
- Process common memory
- Real thread private memory

If you wish to estimate the approximate memory requirements before designing a database, use the formulas described in the subsections below. Note, however, that the values obtained are only rough estimates, since it is impossible to estimate memory requirements in any great detail.

The amount of memory used by the HADB server differs greatly depending on which of the following processing is performed.

■ Processing executed after the HADB server starts

- Execution of normal operations (database connection, SQL statement execution)
- Execution of the `adbimport` command to import data
- Execution of the `adbidxrebuild` command to rebuild indexes
- Execution of the `adbgetcst` command to collect cost information
- Execution of the `adbexport` command to export data
- Execution of the `adbmergechunk` command to merge multiple chunks
- Execution of the `adbarchivechunk` command to archive a chunk
- Execution of the `adbunarchivechunk` command to unarchive a chunk

We recommend that you determine the amount of memory that will be used for each of these types of processing, and then use the largest of the values that are obtained as the amount of memory the HADB server will use. For details about performing normal operations (database connection, SQL statement execution):

- **Execution of normal operations (database connection, SQL statement execution)**
  See 4.3.1 Estimating memory requirements during normal operation.
- **Execution of the `adbimport` command**
  See 4.3.2 Estimating memory requirements during execution of the adbimport command.
- **Execution of the `adbidxrebuild` command**
  See 4.3.3 Estimating memory requirements during execution of the adbidxrebuild command.
- **Execution of the `adbgetcst` command**
  See 4.3.4 Estimating memory requirements during execution of the adbgetcst command.
- **Execution of the `adbexport` command**
  See 4.3.5 Estimating memory requirements during execution of the adbexport command.
- **Execution of the `adbmergechunk` command**
  See 4.3.6 Estimating memory requirements during execution of the adbmergechunk command.

- **Execution of the `adbarchivechunk` command**

  See 4.3.7 Estimating memory requirements during execution of the adbarchivechunk command.

- **Execution of the `adbunarchivechunk` command**

  See 4.3.8 Estimating memory requirements during execution of the adbunarchivechunk command.

If you want to estimate memory requirements more precisely, see 6.3 Estimating the HADB server's memory requirement after you have designed the database and estimated the DB area size.

4.3.1 Estimating memory requirements during normal operation

The following table shows the formula for calculating the amount of memory required during normal operation.

Table 4-3: Memory requirement determinations (during normal operation)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	`100`	None
2		Global buffer page	*dbarea_num* × `1,024 +` (*db_idx* × `1,024`)	`adbbuff` operand
3		Process common memory	`1,024` + *max_users* + *GBUF*	• `adb_sys_proc_area_max` operand • `adbbuff` operand
4		Real thread private memory	*rthd_num* × 1,074	`adb_sys_rthd_area_max` operand
5	Process memory	Heap memory	`1,024`	None

Explanation of the variables

dbarea_num: Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

GBUF: Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

rthd_num: Total number of real threads

If this value cannot be estimated, use the number of CPU cores in the machine on which the HADB server was installed.

db_idx: Size of the data DB area for storing indexes (gigabytes)

Substitute the values determined for the following items in Table 4-1: Formula (DB directory size) in 4.2.1 Estimating the size of the DB directory.

- *Data DB area file for storing B-tree indexes*
- *Data DB area file for storing text indexes*
- *Data DB area file for storing range indexes*

4.3.2 Estimating memory requirements during execution of the adbimport command

The following table shows the formula for calculating the amount of memory required when the `adbimport` command is executed.

Table 4-4: Memory requirement determinations (during execution of the adbimport command)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	`100`	None
2		Global buffer page	*dbarea_num* × `1,024`	`adbbuff` operand
3		Process common memory	`1,024` + *max_users* + *GBUF*	• `adb_sys_proc_area_max` operand • `adbbuff` operand
4		Real thread private memory	`512` + *SORTBUF* + *BLKBUF*	`adb_sys_rthd_area_max` operand
5	Process memory	Heap memory	`1,024` + *sort_rthd* × `0.5`	None

Explanation of the variables

dbarea_num

Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

GBUF

Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

SORTBUF

Sort buffer

Use the following formula to determine the value.

Formula (megabytes)

SORTBUF = *sort_buff_size* × *sort_rthd*

sort_buff_size

Value specified for the import option `adb_import_sort_buff_size`

sort_rthd

Use the following formula to determine the value.

value-specified-for-import-option-adb_import_rthd_num `- 1`

BLKBUF

Buffer for creating indexes

Use the following formula to determine the value.

Formula (megabytes)

```
BLKBUF=
    ↑MAX(((buff_blk_size × buff_blk_num) × (import_rthd × 2) ÷ 1,024) ,
      (((buff_blk_size × buff_blk_num) ÷ 1,024 + 200 + txt_sort_buff_size)
      × (import_rthd × 2)))↑
```

buff_blk_size

Assume 4,096.

buff_blk_num

Value specified for the import option `adb_import_buff_blk_num`

import_rthd

Use the following formula to determine the value.

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

txt_sort_buff_size

Value specified for the import option `adb_import_txt_buff_size`

4.3.3 Estimating memory requirements during execution of the adbidxrebuild command

The following table shows the formula for calculating the amount of memory required when the `adbidxrebuild` command is executed.

Table 4-5: Memory requirement determinations (during execution of the adbidxrebuild command)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	`100`	None
2		Global buffer page	*dbarea_num* × `1,024`	`adbbuff` operand
3		Process common memory	`1,024` + *max_users* + *GBUF*	• `adb_sys_proc_area_max` operand • `adbbuff` operand
4		Real thread private memory	`512` + *SORTBUF* + *BLKBUF*	`adb_sys_rthd_area_max` operand
5	Process memory	Heap memory	`1,024` + *sort_rthd* × `0.5`	None

Explanation of the variables

dbarea_num

Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

GBUF

Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

SORTBUF

Sort buffer

Use the following formula to determine the value.

Formula (megabytes)

$$SORTBUF = sort\_buff\_size \times sort\_rthd$$

sort_buff_size

Value specified for the `adb_idxrebuild_sort_buff_size` index rebuild option

sort_rthd

Use the following formula to determine the value.

value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1

BLKBUF

Buffer for creating indexes

Use the following formula to determine the value.

Formula (megabytes)

$$BLKBUF = \uparrow MAX(((buff\_blk\_size \times buff\_blk\_num) \times (scan\_rthd + dividx\_rthd) \div 1{,}024), (((buff\_blk\_size \times buff\_blk\_num) \div 1{,}024 + 200 + txt\_sort\_buff\_size) \times (scan\_rthd + dividx\_rthd)))\uparrow$$

buff_blk_size

Assume 4,096.

buff_blk_num

Value specified for the `adb_idxrebuild_buff_blk_num` index rebuild option

scan_rthd

Use the following formula to determine the value.

↓(*value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num* - 1) ÷ 2↓

dividx_rthd

Use the following formula to determine the value.

value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1

txt_sort_buff_size

Value specified for the `adb_idxrebuild_txt_buff_size` index rebuild option

4.3.4 Estimating memory requirements during execution of the adbgetcst command

The following table shows the formula for calculating the amount of memory required when the `adbgetcst` command is executed.

Table 4-6: Memory requirement determinations (during execution of the adbgetcst command)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	100	None
2		Global buffer page	*dbarea_num* × 1,024	adbbuff operand
3		Process common memory	1,024 + *max_users* + *GBUF*	• adb_sys_proc_area_max operand • adbbuff operand
4		Real thread private memory	*rthd_num* × 512	adb_sys_rthd_area_max operand
5	Process memory	Heap memory	1,024	None

Explanation of the variables

dbarea_num: Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

max_users

Value specified for the adb_sys_max_users operand in the server definition

GBUF: Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

rthd_num

Value specified for the adb_getcst_rthd_num cost-information collection option

4.3.5 Estimating memory requirements during execution of the adbexport command

The following table shows the formula for calculating the amount of memory required when the adbexport command is executed.

Table 4-7: Memory requirement determinations (during execution of the adbexport command)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	100	None
2		Global buffer page	*dbarea_num* × 1,024 + (*db_idx* × 1,024)	adbbuff operand
3		Process common memory	1,024 + *max_users* + *GBUF* + (*scan_buff_size* x *rthd_num* x 4)	• adb_sys_proc_area_max operand • adbbuff operand
4		Real thread private memory	*rthd_num* × 512	adb_sys_rthd_area_max operand
5	Process memory	Heap memory	1,024	None

Explanation of the variables

dbarea_num: Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

GBUF: Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

scan_buff_size

Value specified for the `adb_export_scan_buff_size` export option

rthd_num

Value specified for the `adb_export_rthd_num` export option

db_idx: Size of the data DB area for storing indexes (gigabytes)

Substitute the values determined for the following items in Table 4-1: Formula (DB directory size) in 4.2.1 Estimating the size of the DB directory.

- *Data DB area file for storing B-tree indexes*
- *Data DB area file for storing text indexes*
- *Data DB area file for storing range indexes*

4.3.6 Estimating memory requirements during execution of the adbmergechunk command

The following table shows the formula for calculating the amount of memory required when the `adbmergechunk` command is executed.

Table 4-8: Memory requirement determinations (during execution of the adbmergechunk command)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	`100`	None
2		Global buffer page	*dbarea_num* × `1,024`	`adbbuff` operand
3		Process common memory	`1,024` + *max_users* + *GBUF*	• `adb_sys_proc_area_max` operand • `adbbuff` operand
4		Real thread private memory	`512` + *SORTBUF* + *BLKBUF*	`adb_sys_rthd_area_max` operand
5	Process memory	Heap memory	`1,024` + *sort_rthd* × `0.5`	None

Explanation of the variables

dbarea_num

Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

GBUF

Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

SORTBUF

Sort buffer

Use the following formula to determine the value.

Formula (megabytes)

$$SORTBUF = sort\_buff\_size \times sort\_rthd$$

sort_buff_size

Value specified for the `adb_mergechunk_sort_buff_size` merge chunk option

sort_rthd

Use the following formula to determine the value.

```
value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
```

BLKBUF

Buffer for creating indexes

Use the following formula to determine the value.

Formula (megabytes)

$$BLKBUF = \uparrow MAX(((buff\_blk\_size \times buff\_blk\_num) \times (scan\_rthd + dividx\_rthd) \div 1{,}024), (((buff\_blk\_size \times buff\_blk\_num) \div 1{,}024 + 200 + txt\_sort\_buff\_size) \times (scan\_rthd + dividx\_rthd)))\uparrow$$

buff_blk_size

Assume 4,096.

buff_blk_num

Value specified for the `adb_mergechunk_buff_blk_num` merge chunk option

scan_rthd

Use the following formula to determine the value.

```
↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
```

dividx_rthd

Use the following formula to determine the value.

```
value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
```

txt_sort_buff_size

Value specified for the `adb_mergechunk_txt_buff_size` merge chunk option

4.3.7 Estimating memory requirements during execution of the adbarchivechunk command

The following table shows the formula for calculating the amount of memory required when the `adbarchivechunk` command is executed.

Table 4-9: Memory requirement determinations (during execution of the adbarchivechunk command)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	`100`	None
2		Global buffer page	*dbarea_num* `x 1,024 +` (*db_idx* `x 1,024`)	`adbbuff` operand
3		Process common memory	`1,024 +` *max_users* `+` *GBUF* `+` (*scan_buff_size* `x` *rthd_num* `x 4`)	• `adb_sys_proc_area_max` operand • `adbbuff` operand
4		Real thread private memory	*rthd_num* `x 512`	`adb_sys_rthd_area_max` operand
5	Process memory	Heap memory	`1,024`	None

Explanation of the variables

dbarea_num: Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

db_idx: Size of the data DB area for storing indexes (gigabytes)

Substitute the values determined for the following items in Table 4-1: Formula (DB directory size) in 4.2.1 Estimating the size of the DB directory.

- *Data DB area file for storing B-tree indexes*
- *Data DB area file for storing text indexes*
- *Data DB area file for storing range indexes*

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

GBUF: Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

scan_buff_size

Value specified for the archive chunk option `adb_arcv_scan_buff_size`

rthd_num

Value specified for the archive chunk option `adb_arcv_rthd_num`

4.3.8 Estimating memory requirements during execution of the adbunarchivechunk command

The following table shows the formula for calculating the amount of memory required when the `adbunarchivechunk` command is executed.

Table 4-10: Memory requirement determinations (during execution of the adbunarchivechunk command)

No.	Memory type		Formula (megabytes)	Related server definitions
1	Shared memory	Shared memory management area	`100`	None
2		Global buffer page	*dbarea_num* `x 1,024`	`adbbuff` operand
3		Process common memory	`1,024 +` *max_users* `+` *GBUF*	• `adb_sys_proc_area_max` operand • `adbbuff` operand
4		Real thread private memory	`512 +` *SORTBUF* `+` *BLKBUF*	`adb_sys_rthd_area_max` operand
5	Process memory	Heap memory	`1,024 +` *sort_rthd* `x 0.5`	None

Explanation of variables

dbarea_num

Number of data DB areas

If this value cannot be estimated, use the combined total number of tables and indexes that will be defined.

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

GBUF

Global buffer

Assume 20 megabytes for each 400 megabytes of the global buffer page size calculated in No. 2.

SORTBUF

Sort buffer

Use the following formula to determine the value.

Formula (megabytes)

SORTBUF = *sort_buff_size* × *sort_rthd*

sort_buff_size

Value specified for the unarchive chunk option `adb_unarcv_sort_buff_size`

sort_rthd

Use the following formula to determine the value.

value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num `- 1`

BLKBUF

Buffer for creating indexes

Use the following formula to determine the value.

Formula (megabytes)

BLKBUF =
↑MAX(((*buff_blk_size* × *buff_blk_num*) × (*unarcv_rthd* × 2) ÷ 1,024) ,
((*buff_blk_size* × *buff_blk_num*) ÷ 1,024 + 200 + *txt_sort_buff_size*)
× (*unarcv_rthd* × 2)))↑

buff_blk_size

Assume 4,096.

buff_blk_num

Value specified for the unarchive chunk option `adb_unarcv_buff_blk_num`

unarcv_rthd

Use the following formula to determine the value.

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

txt_sort_buff_size

Value specified for the unarchive chunk option `adb_unarcv_txt_buff_size`

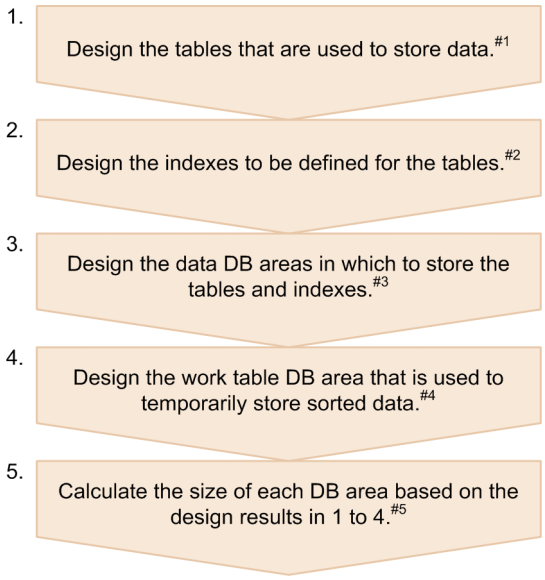
5 Designing a Database

This chapter explains how to design tables, indexes, and DB areas, as well as how to estimate the size of each DB area.

5.1 Database design procedure

The following figure shows the procedure for designing a database.

Figure 5-1: Database design procedure



#1

For details about the procedure, see 5.2 Designing a table.

#2

For details about the procedure, see the following sections:

- 5.3 Designing a B-tree index
- 5.4 Designing a text index
- 5.5 Designing a range index

#3

For details about the procedure, see 5.6 Designing a data DB area.

#4

For details about the procedure, see 5.7 Designing a work table DB area.

#5

For details about the procedure, see the following sections:

- 5.8 Estimating the size of the data DB area
- 5.9 Estimating the size of the work table DB area
- 5.10 Estimating the size of the master directory DB area
- 5.11 Estimating the size of the dictionary DB area
- 5.12 Estimating the size of the system-table DB area

5.2 Designing a table

This section explains the items that must be considered when designing a table.

5.2.1 Flow of table design

The following shows the general procedure for designing tables:

1. Determine whether to define the table as a row store table or a column store table.
   With reference to 5.2.2 Criteria for selecting row store tables and column store tables, determine whether to define the table as a row store table or a column store table.
2. Determine whether to define the table as a single-chunk table or a multi-chunk table.
   With reference to 5.2.3 Criteria for selecting single or multi-chunk tables, determine whether to define the table as a single-chunk table or a multi-chunk table.
3. When defining the table as a row store table, assess the normalization of the table.
   With reference to 5.2.6 Normalizing a table [Row store table], assess the normalization of the table.
4. Assess the items to specify when defining the table (assess the options to specify in the `CREATE TABLE` statement).
   With reference to the following topics, assess each of the items to be specified in the table definition:
   - 5.2.7 Fixing the row length (FIX specification) [Row store table]
   - 5.2.8 Setting a default value for a column (DEFAULT clause)
   - 5.2.9 Specifying a primary key (uniqueness constraint definition) (PRIMARY KEY) [Single-chunk table]
   - 5.2.10 Specifying a foreign key (FOREIGN KEY)
   - 5.2.11 Allocating an unused area inside the data page (PCTFREE) [Row store table]
   - 5.2.12 Branch specification for column data of variable-length data types (BRANCH) [Row store table]

Note

- The label [Row store table] indicates that the item needs to be assessed when defining the table as a row store table. You do not need to assess these items when defining the table as a column store table.
- The label [Single-chunk table] indicates that the item needs to be assessed when defining the table as a single-chunk table. You do not need to assess these items when defining the table as a multi-chunk table.

5.2.2 Criteria for selecting row store tables and column store tables

This section explains the criteria for selecting whether to define a table as a row store table, or as a column store table.

(1) Restrictions on column store tables

The following operational restrictions apply to column store tables. If operational issues could occur as a result of any of these restrictions, define the table as a row store table.

- You cannot define a text index for a column store table.

- You cannot define a column store table as an archivable multi-chunk table.
- You cannot define a column store table as a FIX table.

(2) Criteria for selecting whether to define a table as a row store table or column store table

Use the following criteria to determine whether to define a table as a row store table or a column store table:

Table 5-1: Criteria for selecting row store tables and column store tables

Manner of data retrieval from table or salient characteristic of table being defined		Suitable table type
Manner of data retrieval from table	Data is retrieved in the following manner: • Data retrieval that accesses data at the row level is common This applies to situations where an asterisk (*) is specified in the selection expression of a `SELECT` statement, and where the selection expression specifies almost every column. • Operations are common that retrieve data by narrowing down the search range using B-tree indexes	Row store table
	Data is retrieved in the following manner: • Searches often target data in a specific column in its entirety without significantly narrowing the search range • The data in a specific column within a specific range (such as a specific month or year) is frequently accessed • There are frequent calculations (such as computing averages and totals) in relation to the value data in a specific column • Grouping data in specific columns on a frequent basis	Column store table
Operation performed for the table to be defined	• If data is added, updated, or deleted for the table: For the `INSERT`, `UPDATE`, and `DELETE` statements, processing performance is superior when the processing target is a row store table. If the `INSERT`, `UPDATE`, or `DELETE` statement is executed for a column store table, the retrieval performance might be degraded# after the statement is executed. Therefore, if you use SQL statements to add, update, and delete data as part of routine tasks, define a row store table.	Row store table
Size of table being defined	• When the table will store a relatively small amount of data This applies to tables that store a relatively small amount of data, and to tables into which a relatively small amount of data is imported in a single import operation. As a general rule, the size of the input data file to be imported into such a table will be less than 1 GB. Note that the size of the data in a row store table is approximately twice the size of the input data file. However, this varies according to the nature of the data being imported. Normally, the data in a row store table is larger than the same data in a column store table.	Row store table
	• When the table will store a large amount of data This applies to tables that store a large amount of data, and to tables into which a large amount of data is imported in a single import operation. As a general rule, such a table will have an input data file size of 1 GB or more. Note that the size of the data in a column store table is approximately half the input data file size. However, this varies according to the nature of the data being imported. Normally, the data in a column store table is smaller than the same data in a row store table.	Column store table
Use in a data warehouse	• When the table is joined by a primary key with tables that store data to be analyzed This applies to dimension tables and master tables in a data warehouse.	Row store table

Manner of data retrieval from table or salient characteristic of table being defined		Suitable table type
	• When the table stores data to be analyzed This applies to fact tables in a data warehouse, and tables that store log data and sensor data. We particularly recommend that you define a table as a column store table when it will store several gigabytes or more of data to be analyzed (data that is searched by narrowing the search range using a range index). The size of the data to be analyzed can be estimated based on the size of the input data file. If the size of the data to be analyzed does not exceed several hundred megabytes, we recommend that you define the table that stores data to be analyzed as a row store table.	Column store table

#

You can prevent degradation of retrieval performance for column store tables by enabling the updated-row columnizing facility. For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).

Important

A table suited to definition as a column store table is also generally best defined as a multi-chunk table. For this reason, we recommend that you define these tables as multi-chunk column store tables. For details about the points to consider when defining a table as a multi-chunk table, see 5.2.4 Points to consider in defining a multi-chunk table.

To define a table as a row store table, either specify `ROW` in the `STORAGE FORMAT` option in the `CREATE TABLE` statement, or omit the `STORAGE FORMAT` option altogether. To define a table as a column store table, specify `COLUMN` in the `STORAGE FORMAT` option in the `CREATE TABLE` statement.

For details about the `CREATE TABLE` statement, see *Definition SQL* in the manual *HADB SQL Reference*.

(3) B-tree indexes used when retrieving data from column store tables

A B-tree index defined for a column store table can be used to search that column store table only in limited circumstances. The following are typical cases in which a B-tree index is used for a column store table. We recommend that you define a B-tree index for a column store table only when the range of the processing target can be narrowed as shown in these cases.

- Case where an index that will be used for index specification is specified when an SQL statement is executed
- Case where the minimum value is obtained by specifying the set function `MIN` or the maximum value is obtained by specifying the set function `MAX`
- Case where the `UPDATE` or `DELETE` statement is executed by specifying a search condition

In cases such as accessing data for a specific year or month, we recommend that you define range indexes and use them to narrow the search range.

For details about the rules that govern the use of B-tree indexes when retrieving data from column store tables, see *B-tree indexes and text indexes used during execution of SQL statements* in the *HADB Application Development Guide*.

Note

The updated-row columnizing facility is not applied to column store tables for which B-tree indexes are defined. For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).

(4) Column-data compression types for column store tables

When importing data into a column store table, the data in each column is compressed using one of the compression types listed in the following table. This means that the same data will be smaller in a column store table than in a row store table. When importing data, the HADB server automatically selects one of the compression types in the following table based on the data being imported.

Table 5-2: Column-data compression types for column store tables

No.	Compression type	Description
1	Non-compression (`NONE`)	The data is stored uncompressed.
2	Run-length encoding (`RUNLENGTH`)	When the same data value is repeated, the data is stored as the data value and number of consecutive occurrences. This compression type is effective on data where the same value occurs in consecutive elements. Compression example: `AAAABB` → `4A2B`
3	Delta encoding (`DELTA`)	Data is stored as the first data value and the subsequent differences between adjacent data values. This compression type is effective when data is concentrated around a specific value. Compression example: `100,99,101,103` → `100,-1,2,2`
4	Delta run-length encoding (`DELTA_RUNLENGTH`)	This compression type applies run-length encoding to delta-encoded data. This compression type is effective on data that is incremented by a certain value (data that follows an arithmetic progression). This compression type is effective on data such as slip numbers that are incremented by one each time.
5	Dictionary encoding (`DICTIONARY`)	Data that appears often is stored in a dictionary, and the data in the table contains only index references to the dictionary. This compression type is effective on data such as character string data with low cardinality (low uniqueness of data). The maximum dictionary size is 16 kilobytes.

As shown in the preceding table, there are compression methods that eliminate redundancy among data values, and those that make use of the differences between values. This means that compression is less effective when applied to data that has little repetition or little difference between data items, such as character string data and binary data of 128 bytes and longer.

Note

- Upon completing data import, HADB server outputs the compression type it selected in a message.
- The compression type is determined when data is imported. Therefore, if the pattern of data changes while data is being imported into a table, the compression type might change even within the same column.
- HADB server selects the compression type at the level of the individual threads that store the data being imported.
- HADB server calculates the compression rate for the first 16 megabytes of column data to be stored, and selects the compression type with the best compression rate. If a poor compression rate means the

size of the compressed data is the same as or larger than the uncompressed data, the HADB server selects non-compression as the compression type.

Note that you can also specify the compression type for each column in a column store table when performing the following operations. In this case, the column data will always be compressed using the specified compression type.

- Using the CREATE TABLE statement to define a column store table
- Using the ALTER TABLE statement to add a column to a column store table

■Relationship between a column store table for which the UPDATE statement or DELETE statement is executed and the compression type

In the case of column store tables that have an update as a prerequisite, we recommend specifying a compression type other than a delta compression type[#1]. Do not let HADB automatically select the compression type[#2].

Reason:

When an UPDATE statement or DELETE statement is executed on data in column-store format, HADB extends the length of all columns for all the targeted row data. When extending the length for DELTA compression, HADB needs to calculate the data values from the beginning of the data to the data subject to the lengthening. As a result, processing to extend the length of the columns takes longer than other compression methods. Therefore, the time required to execute the UPDATE statement or DELETE statement takes longer when a DELTA compression method is used for a column store table.

#1

In the COMPRESSION TYPE specification in a CREATE TABLE statement, we recommend specifying a compression type other than DELTA and specifying something other than automatic selection (AUTO).

#2

If the compression type specification is omitted, or if AUTO is specified for the compression type specification, HADB automatically selects the compression method. As a result, HADB might apply a DELTA compression method.

(5) Examples of when to use row store tables and column store tables

This subsection explains when you might use a row store table and when you might use a column store table, based on the following operational example:

Operational example

Company A intends to operate a data warehouse for the purpose of analyzing the sales data collected from its sales offices. The analysis axes for sales data are dates, sales offices, products, and customers. Company A decided to implement the following star schema as the schema model for its data warehouse.

Figure 5-2: Data warehouse schema model for analysis of Company A's sales data

In the preceding schema, the SALES table is a fact table. The PRODUCTS table, CUSTOMERS table, OFFICES table, and CALENDAR table are dimension tables. The following explains the data stored in each table:

- SALES table

The SALES table stores the past 10 years of sales data. This consists of such data as product IDs of sold products, the number of products sold, unit prices, customer IDs, IDs of offices whose sales contributed to the totals, and the dates on which sales occurred. Because approximately 100,000 items of data are stored per day, this table will ultimately store approximately 360,000,000 items of sales data.

The data stored as product IDs, customer IDs, office IDs, and sales dates in this table serve as primary keys (foreign keys) for the PRODUCTS, CUSTOMERS, OFFICES, and CALENDAR tables respectively.

- PRODUCTS table

  The PRODUCTS table stores data about the products handled by Company A (such as product IDs, names, and categories).

- CUSTOMERS table

  The CUSTOMERS table stores data about the customers of Company A (such as their customer IDs, names, industries, and addresses).

- OFFICES table

  The OFFICES table stores data about the sales offices of Company A (such as office IDs, names, and addresses).

- CALENDAR table

  The CALENDAR table stores dimension data related to dates and times (date, month, business quarter, year).

When activity such as compiling monthly sales data and analyzing sales at the customer and office level takes place in relation to the preceding schema model, the following forms of data retrieval are likely to occur frequently:

- Retrieving all data in the SALES table
- Retrieving data in the SALES table for a specific period (such as the current month or the previous month)

Therefore, the SALES table is defined as a column store table. A range index is defined for the column that stores dates of sale.

The PRODUCTS table, CUSTOMERS table, OFFICES table, and CALENDAR table are likely to be joined using the foreign keys stored in the SALES table. Because these tables contain a relatively small number of data items, they are defined as row store tables. A B-tree index is defined for columns that correspond to primary keys.

5.2.3 Criteria for selecting single or multi-chunk tables

First, determine whether to define the table as a row store table or a column store table. Then, determine whether to define that table as a single-chunk table or a multi-chunk table (regular multi-chunk table or archivable multi-chunk table). The following table shows the key features of single-chunk tables and multi-chunk tables:

Table 5-3: Key features of single-chunk tables and multi-chunk tables (regular multi-chunk tables and archivable multi-chunk tables)

Item	Table type		
	Single-chunk table	Multi-chunk table	
		Regular multi-chunk table	Archivable multi-chunk table
Can the background-import facility be used?	N	Y	Y
Can a primary key be defined for the table?	Y	N	N
Can a unique index be defined for the table?	Y	N	N

Item	Table type		
	Single-chunk table	Multi-chunk table	
		Regular multi-chunk table	Archivable multi-chunk table
Can the data in a row store table be compressed?	N	N	Y
Can the data in a table be updated or deleted (can an `UPDATE` statement or `DELETE` statement be executed)?	Y	Y	P[#1]
Is data retrieval faster or slower?	Faster	Faster	Slower[#2]

Legend:

Y: Can be executed, or has no impact on performance.

P: Some data cannot be updated or deleted.

N: Cannot be executed.

#1

You cannot update or delete the data in an archived chunk.

#2

Applies when retrieving data from an archived chunk.

Note that the preceding table lists only the key features of single-chunk tables and multi-chunk tables. For details about the advantages, disadvantages, and restrictions of single-chunk tables and multi-chunk table, see the following topics. Determine whether to define a table as a single-chunk table or a multi-chunk table (regular multi-chunk table or archivable multi-chunk table) after reading these topics.

- 5.2.4 Points to consider in defining a multi-chunk table
- 5.2.5 Points to consider in defining an archivable multi-chunk table [Row store table]

Important

You cannot define a column store table as an archivable multi-chunk table.

5.2.4 Points to consider in defining a multi-chunk table

This section describes the points you need to consider when defining a base table as a multi-chunk table.

After reading this section, also read 5.6.2 Points to consider in storing a multi-chunk table in the data DB area.

In addition, if you define a base table as an archivable multi-chunk table, read 5.2.5 Points to consider in defining an archivable multi-chunk table [Row store table] after reading this section.

(1) Advantages of defining a base table as a multi-chunk table

Defining a base table as a multi-chunk table brings the following advantages:

- You can search data and import data simultaneously (you can use the background-import facility).
- You can use the `PURGE CHUNK` statement to delete all data in a chunk.

- When you cannot search data, you can import the data so that you can search it at any time later. For an example of this type of operation, see 11.4.24 Operation taking background import and chunks into consideration (Example 2: Using chunks in wait status).

> **Note**
>
> If you frequently store only small amounts of data (4 megabytes or less), consider using the `INSERT` data manipulation SQL statement instead of the background-import facility. Frequently storing small amounts of data by using background import generates wasted areas, lowering the data storage efficiency. This is because background import reserves and allocates a new segment even if there is free space in already allocated segments.

If you define a base table as a column store table for which the `INSERT`, `UPDATE`, or `DELETE` statement will probably be executed, define the base table as a column store table that is a multi-chunk table. A column store table for which the `INSERT`, `UPDATE`, or `DELETE` statement is executed repeatedly might need to be reorganized. If the table is a multi-chunk table, you can perform reorganization on a chunk basis. This brings the following advantages:

- A shorter time is required for reorganization.
- Less disk space is required for reorganization.

Also, if you define a base table as an archivable multi-chunk table, you can use the chunk archiving function. If you intend to define a base table as an archivable multi-chunk table, read 5.2.5 Points to consider in defining an archivable multi-chunk table [Row store table] after reading this section.

(2) Disadvantages of defining a base table as a multi-chunk table

- Primary keys and unique indexes cannot be defined for multi-chunk tables.
- Management and maintenance of chunks are required. For example, chunks must be merged periodically. For details, see 11.4.9 Merging chunks (to reduce the number of chunks).

(3) Item that must be specified in the `CREATE TABLE` statement (maximum number of chunks)

When you define a base table as a multi-chunk table, specify the maximum number of chunks in the chunk specification in the `CREATE TABLE` statement.

(4) Determining the maximum number of chunks

A new chunk is created each time background import is executed using the `adbimport` command. Multiple created chunks can also be merged into a single chunk by using the `adbmergechunk` command. Note that you cannot create more chunks than the *maximum-number-of-chunks* value specified for *chunk-specification* in the `CREATE TABLE` statement.

Therefore, you need to estimate the value of *maximum-number-of-chunks* based on the following three factors:

- How often background import is executed using the `adbimport` command
- How often chunks are merged using the `adbmergechunk` command
- How long chunks are retained

You can use the PURGE CHUNK statement to delete created chunks. You can then create a number of new chunks equivalent to the number you deleted. The *maximum-number-of-chunks* value specified in the CREATE TABLE statement can be changed by using the ALTER TABLE statement.

Use the following formula to determine the value for the *CHUNK_NUM* variable. Then, specify the value determined by the formula or a larger value for *maximum-number-of-chunks* in the CREATE TABLE statement.

Formula (count)

```
CHUNK_NUM =
    ↑years x ↑(import_num ÷ merge_source_chunk_num)↑ x safe_rate↑
    + (↑merge_source_chunk_num x safe_rate↑)#
```

#: A certain amount of leeway is built into this formula to account for merge-source chunks that might remain as deletion-pending chunks after execution of the adbmergechunk command.

> **Important**
>
> Merge-source chunks that are not deleted when the adbmergechunk command is executed remain as *deletion-pending chunks*. This reduces the number of chunks you can use by a number equivalent to the number of deletion-pending chunks. For this reason, you need to consider the number of deletion-pending chunks when determining the value of the *CHUNK_NUM* variable. For details about deletion-pending chunks, see (3) Using the adbmergechunk command to merge chunks in 11.4.9 Merging chunks (to reduce the number of chunks).

Explanation of variables

years

The number of years over which the chunks (data stored in the base table) are to be retained.

Specify the number of years that must elapse after a chunk has been created by background import using the adbimport command before the chunk can be deleted by the PURGE CHUNK statement.

import_num

How many times a year background import will be executed using the adbimport command.

merge_source_chunk_num

The number of merge-source chunks present when the adbmergechunk command is executed.

For example, if you use the adbimport command to perform background import once a day and use the adbmergechunk command to merge chunks once every five days, use the number 5 for this value.

safe_rate

Safety factor Use 1.2 for this value.

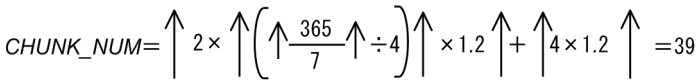
The following shows examples of how the *CHUNK_NUM* variable is determined.

■ Example of determining the *CHUNK_NUM* variable

In this example, the *CHUNK_NUM* variable is determined for the following conditions:

- Chunks (the data stored in the base table) are retained for two years.
- Background import is executed once a week using the adbimport command.
- Four chunks are merged once a month using the adbmergechunk command.
- The safety factor is set to 1.2.

Formula (chunks)



In this example, the value of the *CHUNK_NUM* variable is 39. Accordingly, specify `39` or a larger value for *maximum-number-of-chunks*.

5.2.5 Points to consider in defining an archivable multi-chunk table [Row store table]

This subsection describes the points you need to consider when defining a base table as an archivable multi-chunk table.

Note that the description in this subsection assumes that you have already read the explanation in 5.2.4 Points to consider in defining a multi-chunk table.

(1) Advantages of defining a base table as an archivable multi-chunk table

The database size can be reduced by compressing the data in chunks.

(2) Disadvantages of defining a base table as an archivable multi-chunk table

- Archived data cannot be updated by using the `UPDATE` statement and cannot be deleted by using the `DELETE` statement. To update or delete data in chunks, you must unarchive the data. For details about the conditions in which the `UPDATE` or `DELETE` statement results in an error, see the following sections in the manual *HADB SQL Reference*:
  - *Rules* in *Specification format and rules for the UPDATE statement*
  - *Rules* in *Specification format and rules for the DELETE statement*
- A search in archived data takes more time than a search in non-archived data for the following reasons:
  - Time is required to expand the data.
  - Index-based search cannot be performed.
- When you search an archivable multi-chunk table, narrow down the search range by specifying the information about the archive range column as the search condition in the `SELECT` statement. If you cannot specify the archive range column as a search condition in the `SELECT` statement, a search takes a long time because the search range cannot be narrowed down.

  For the points to consider when searching an archivable multi-chunk table, see *Considerations when searching an archivable multi-chunk table* in the *HADB Application Development Guide*.

Note

- To change an archivable multi-chunk table to a single-chunk table, you must use the `CREATE TABLE` statement to redefine the table. Therefore, carefully consider the advantages and disadvantages, and then decide whether to define an archivable multi-chunk table.

- When changing an archivable multi-chunk table to a regular multi-chunk table, you do not need to redefine the table by using the `CREATE TABLE` statement. You can change the table by using the `ALTER TABLE` statement.

(3) Conditions of data to be stored in an archivable multi-chunk table

In an archivable multi-chunk table, store chronological data that is generated every day. Chronological data here means sensor-originating data, shop sales data, and other data that consists of sequential records that are generated chronologically. Note that each record of chronological data must include datetime information, such as when the record was generated or when a product was sold.

(4) Creating an archive directory

Before you execute the `CREATE TABLE` statement, you must create an archive directory. Also, you must grant the HADB administrator the necessary access privileges for the archive directory. For details, see *chunk-archive-specification* in *chunk-specification* in *Explanation of specification format* in *Specification format and rules for the CREATE TABLE statement* in the manual *HADB SQL Reference*.

We recommend that you create an archive directory for each archivable multi-chunk table. Multiple archivable multi-chunk tables can share one archive directory. However, in such a case, concentration of access on one archive directory might degrade performance.

If you create archive directories on a performance-first basis, the following recommendations apply:

- Create an archive directory for each archivable multi-chunk table.
- Create only one archive directory in a file system dedicated for that archive directory.
- Create only archive directories in a file system dedicated for those archive directories.

(5) Specification of the archive range column

When you define an archivable multi-chunk table, you must specify the archive range column. Note that a column having one of the following data types cannot be specified as the archive range column:

- `CHARACTER` type (if the definition length is 33 bytes or longer)
- `VARCHAR` type
- `BINARY` type
- `VARBINARY` type

The data stored in the archive range column is used to narrow down the search range when an archivable multi-chunk table is searched. Therefore, make sure that the column you want to use as the archive range column contains datetime information, such as when the record was generated or when a product was sold.

Note that NOT NULL constraint is applied to the archive range column.

(6) Range index automatically defined for the archive range column

The range index is automatically defined for the archive range column. For the range index, you do not need to prepare the dedicated data DB area, because the amount of data in the range index is small. Store the range index and other indexes defined for an archivable multi-chunk table in the same data DB area, unless there is a good reason to do otherwise.

In the chunk specification in the `CREATE TABLE` statement, specify the data DB area that will store the range index.

Note that you can specify an index identifier for the range index that is automatically defined for the archive range column. If you do not specify an index identifier for the range index, the HADB server automatically determines the index identifier.

5.2.6 Normalizing a table [Row store table]

When defining a table as a row store table, consider the normalization of the table.

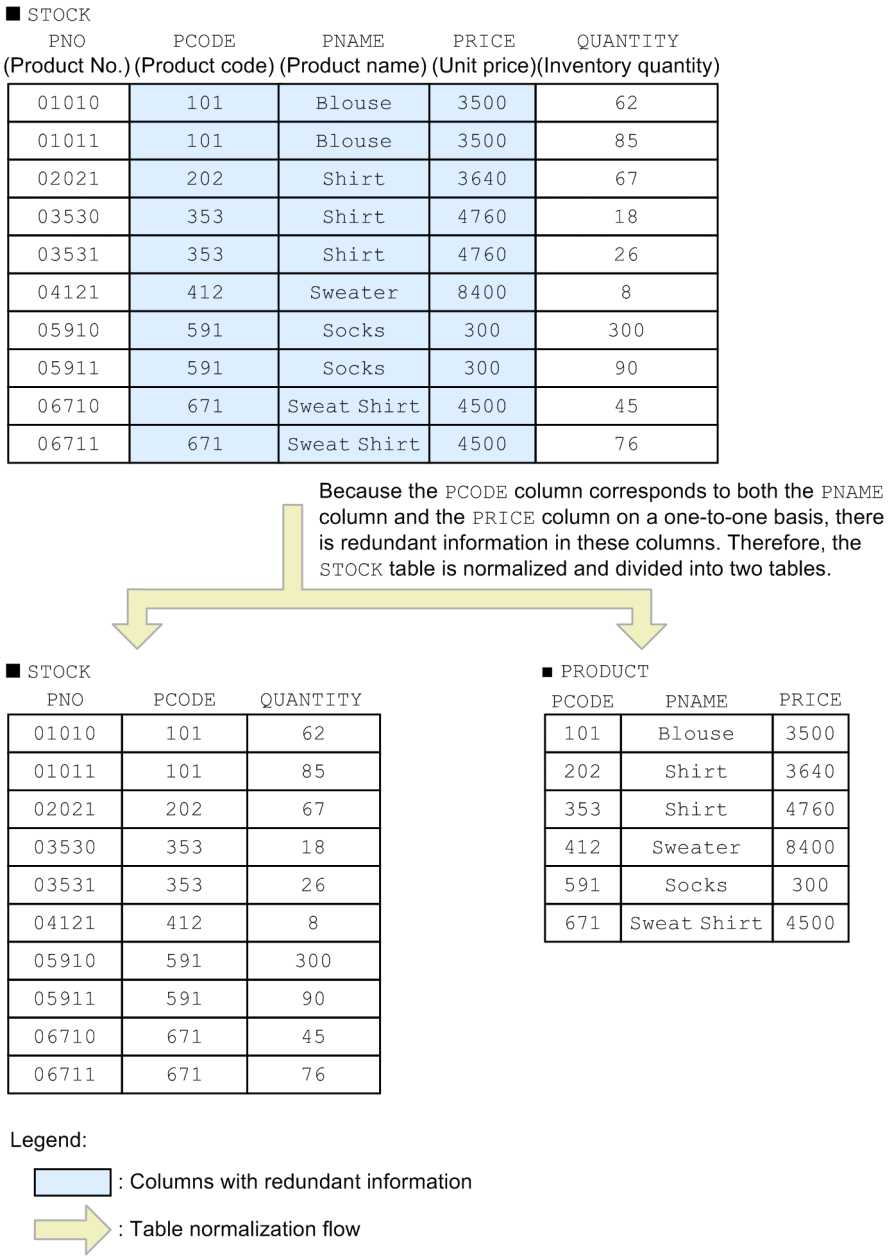
Moving redundant data in a table to another table is called *table normalization*. To improve the storage efficiency of tables, normalize tables. By repeatedly normalizing tables, you can configure a complicated database into an optimum form.

(1) Improving the storage efficiency of tables

When a table contains multiple columns with similar information, normalize the table by dividing it into multiple tables so each table contains only one of the columns with similar information. This operation improves the data storage efficiency in the table.

The following figure shows an example of normalizing a table containing multiple columns with similar information.

Figure 5-3: Example of normalizing a table containing multiple columns with similar information



Explanation

Before the STOCK table is normalized, column PCODE corresponds to both column PNAME and PRICE on a one-to-one basis. As a result, there is redundant information in these columns. In this case, you can break up the STOCK table and create a separate PRODUCT table with columns PCODE, PNAME, and PRICE.

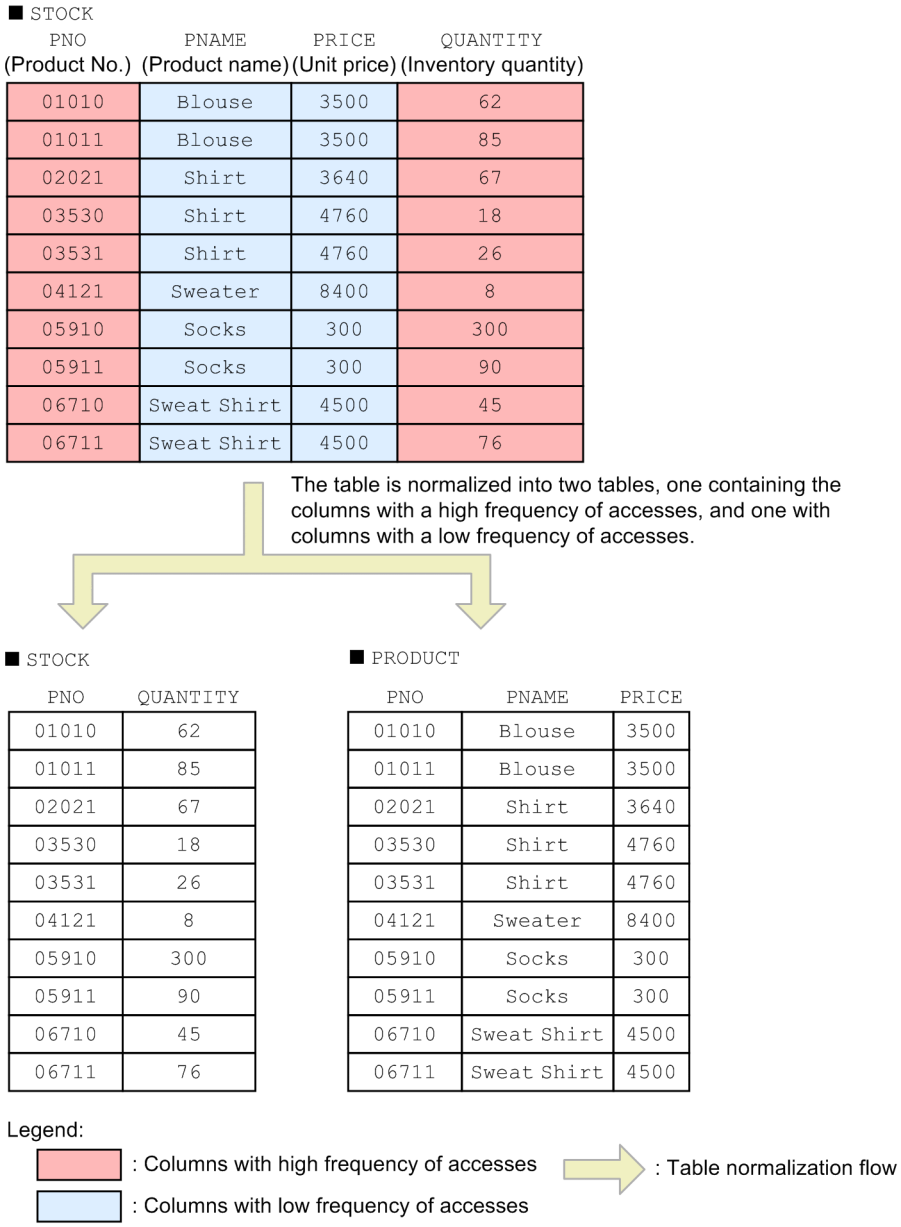
By doing this, the PRODUCT table does not contain redundant information in its columns PCODE, PNAME, and PRICE.

(2) When some columns have high access frequency while others do not

When a table contains some columns that are accessed frequently and others that are accessed less frequently, you normalize the table into a table containing the columns with high access frequency and a table containing the columns with low access frequency.

The following figure shows an example of normalizing a table containing columns with high access frequency and columns with low access frequency.

Figure 5-4: Example of normalizing a table containing columns with high access frequency and columns with low access frequency



Explanation

Before the STOCK table is normalized, it contains columns with high access frequency (PNO and QUANTITY) and columns with low access frequency (PNAME and PRICE). In this case, you can normalize the STOCK table into a table containing only the columns with high access frequency (the STOCK table) and a table containing only the columns with low access frequency (the PRODUCT table).

This normalization improves the overall processing efficiency.

5.2.7 Fixing the row length (FIX specification) [Row store table]

When defining a table as a row store table, consider whether to define the row store table as a FIX table. When defining a table as a column store table, you cannot define that column store table as a FIX table.

A table in which every row contains fixed length data is called a *FIX table*.

Specifying the FIX option for a table containing only fixed-length data, and containing no null values, improves performance. Compared to tables for which the FIX option is not specified, the physical row length becomes 2 bytes shorter per column. Therefore, you can reduce the amount of disk space required when the table contains a large number of columns.

You can define a table as a FIX table by specifying the FIX option in the CREATE TABLE statement. You can specify the FIX option only if both of the following conditions are satisfied:

- There are no columns that contain null values.
- There are no variable-length columns.

5.2.8 Setting a default value for a column (DEFAULT clause)

If you want to have predefined values stored automatically when data is added or updated, consider setting default values for columns. A default value for a column is the value that will be stored in the applicable column in the following cases:

- Row insertion by the INSERT statement

  A default value for a column is stored in the following cases:

  - DEFAULT is specified for the insertion value.
  - DEFAULT VALUES is specified.
  - The name of the column in which data is to be inserted is omitted (except when all column names are omitted).
  - A row is inserted in a viewed table (a default value for the column is stored in the column in the underlying table that does not correspond to a column in the viewed table).
- Column value updating by the UPDATE statement

  If DEFAULT is specified for an update value, the default value for the column is stored.
- Data import by the adbimport command

  When data is imported by the adbimport command and a field's data in the input data file is empty, the default value for the column is stored.

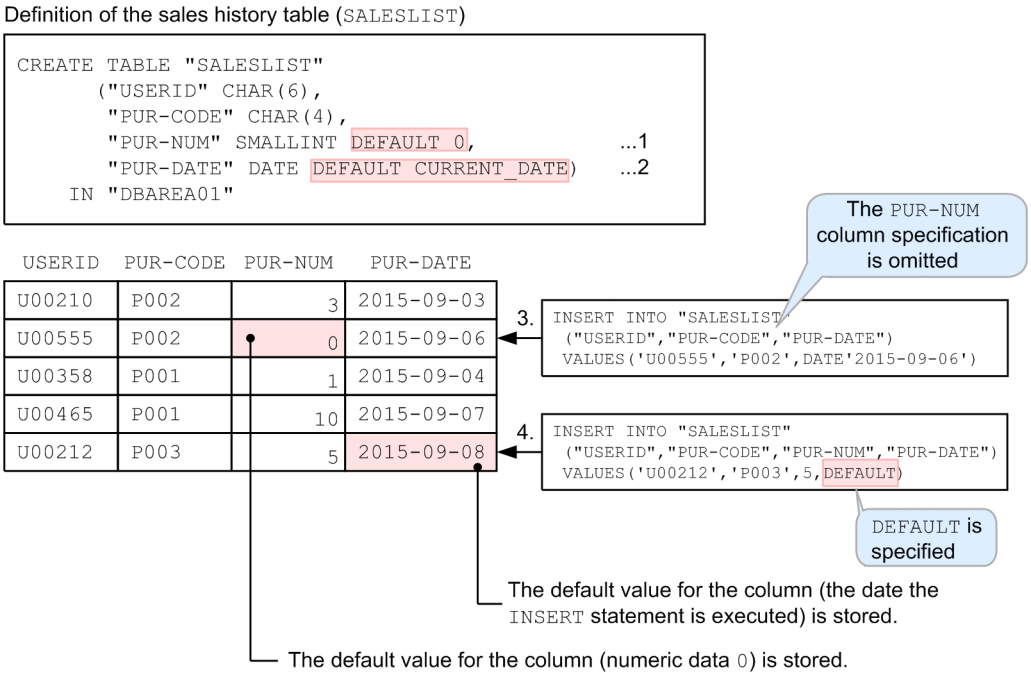
  Note that if NULL is specified in the adb_import_null_string import option, the null value, not the default value for the column, is stored.

To set a default value for a column, specify the DEFAULT clause in the CREATE TABLE statement. For details about the DEFAULT clause, see *DEFAULT clause* in the manual *HADB SQL Reference*.

> **Note**
>
> If the DEFAULT clause is omitted, the default value for the column will be the null value.

The following figure shows an example of data storage when a default value for a column is set by specifying the DEFAULT clause.

Figure 5-5: Example of data storage when a default value for a column is set by specifying the DEFAULT clause



Explanation

1. Specifies the DEFAULT clause for the PUR-NUM column to define the numeric data 0 as the default value for the column.
2. Specifies the DEFAULT clause for the PUR-DATE column to define the date as the default value for the column. CURRENT_DATE means the date the INSERT statement or the UPDATE statement is executed or the date the adbimport command is started.
3. Because the PUR-NUM column is omitted from the columns subject to insertion, the default value for the column (numeric data 0) is stored in the PUR-NUM column.
4. Because DEFAULT is specified as the insertion value, the default value for the column (the date the INSERT statement is executed) is stored in the PUR-DATE column.

For details about the values that can be specified as the default value for a column, see *DEFAULT clause* in the manual *HADB SQL Reference*.

5.2.9 Specifying a primary key (uniqueness constraint definition) (PRIMARY KEY) [Single-chunk table]

When defining a table as a single-chunk table, consider whether to define a primary key. You cannot define a primary key if you define the table as a multi-chunk table.

A key for uniquely identifying a row within a base table is called a *primary key*. By defining a primary key, you can maintain the uniqueness of the base table's data.

To define a primary key, specify a uniqueness constraint definition in the CREATE TABLE statement.

When a primary key is defined, the following two constraints are applied to the columns that comprise the primary key:

- Uniqueness constraint

  This constraint does not permit a column value (or column values in multiple columns) to be duplicated.
- `NOT NULL` constraint

  This constraint (`NOT NULL`) does not permit a column value to be the null value.

A unique index is automatically defined, with the columns holding the primary key set as the indexed columns. The name of the index identifier of the unique index is the same as the constraint name in the uniqueness constraint definition specified in the `CREATE TABLE` statement. For details about unique indexes, see 5.3.5 Points to consider in determining the columns to be defined for a unique index.

When you define a primary key, select a primary key from a column or combination of columns (candidate keys) that can uniquely identify a row within the base table. If there are multiple candidate keys, define the most significant candidate key as the primary key.

Important

Even when a primary key is defined for the base table, the uniqueness constraint cannot be guaranteed if data containing duplicate values is imported using the `adbimport` command. In such a case, search the base table for duplicate keys and delete the rows that have the searched key value. For details about how to search for duplicate keys, see (2) Steps to take when the uniqueness constraint is violated in 15.9.2 Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output).

5.2.10 Specifying a foreign key (FOREIGN KEY)

A column (or a combination of columns) that references the primary key of another base table is called a *foreign key*. Using primary keys and foreign keys, you can define relationships among the tables in a database.

To define a foreign key, specify a referential constraint definition in the `CREATE TABLE` statement.

Important

Even when you specify a referential constraint definition in the `CREATE TABLE` statement, the referential constraint cannot be checked based on the foreign key. Therefore, a foreign key cannot be used to maintain data consistency between tables.

Note

- Define a foreign key when you want to use a business intelligence tool or the like to output an entity-relationship diagram or define a relationship among tables in a database.
- For details about primary keys, see 5.2.9 Specifying a primary key (uniqueness constraint definition) (PRIMARY KEY) [Single-chunk table].

5.2.11 Allocating an unused area inside the data page (PCTFREE) [Row store table]

When defining a table as a row store table, consider the setting to use for the percentage of unused area in the data page. You cannot set the percentage of unused area in the data page when defining a table as a column store table.

You can use the `PCTFREE` operand in the `CREATE TABLE` statement to specify a percentage of unused area in the data page (page that stores base table data).

Executing either of the following commands stores base table data as a percentage of unused area specified for `PCTFREE`:

- `adbimport` command
- `adbunarchivechunk` command

However, if a base table is updated frequently, it might become impossible to store new rows on the same page. As a result, row positioning becomes fragmented, leading to performance degradation. If you expect to update a base table frequently, specify a high percentage of unused area to be reserved in the data page.

Guidelines for allocating an unused area in the data page follow:

If you can estimate the number of rows that will be updated in the base table

Specify the value determined using the following formula in the `PCTFREE` operand.

Formula (%)

percentage of unused area in data page ＝
↑ *upd_row_num*÷（*row_num*＋*upd_row_num*）×100 ↑

Explanation of variables

- *upd_row_num*: Estimated number of rows to be updated (rows)
- *row_num*: Number of rows in the entire base table (rows)

If you cannot estimate the number of rows that will be updated in the base table

Specify `30` (%) (default value of the `PCTFREE` operand) for the percentage of unused area.

If no rows in the base table will be updated

Specify `0` (%) for the percentage of unused area.

5.2.12 Branch specification for column data of variable-length data types (BRANCH) [Row store table]

When defining a table as a row store table, consider whether to specify a *branch specification for column data of variable-length data types*. You cannot specify a *branch specification for column data of variable-length data types* when defining a table as a column store table.

This subsection explains branch specifications for column data of variable-length data types. A variable-length data type means either of the following data types:

- `VARCHAR`
- `VARBINARY`

Normally, an entire row of data is stored on the same page. However, if a column of a variable-length data type is defined in a table, the column data of the variable-length data type might sometimes be branched and stored on separate pages.

Column data of the variable-length data type is stored according to the following rules:

- **For a column of the variable-length data type whose definition length is 255 bytes or less**
  The data of such a column is not branched. All the column data is stored on the same page.
- **For a column of the variable-length data type whose definition length is 256 bytes or greater**
  If the base row fits on one page, the data of such a column is not branched. The entire data of such a column is stored on the same page.
  If the base row does not fit on one page, the data of a column of the variable-length data type is branched and stored on multiple pages.

Using the specification for `BRANCH` in the column definition of the `CREATE TABLE` statement, you can specify whether to make the data of a column of the variable-length data type always subject to branching onto separate pages. Use the following guidelines to determine whether to branch:

- **If the data of a column of the variable-length data type is not referenced frequently**
  Branch the data of such a column for storage on separate pages. This might improve retrieval processing performance for the table.
  If you specify `YES` for `BRANCH`, the specified data of a column of the variable-length data type becomes subject to branching.
- **If the data of a column of the variable-length data type (whose data length is 256 bytes or greater) is referenced frequently**
  Do not branch the data of such a column. This might improve retrieval processing performance for the table.
  If you specify `NO` for `BRANCH`, the specified data of a column of the variable-length data type does not become subject to branching.

If you specify the `BRANCH ALL` table option, all the data in a column of the variable-length data type in the table becomes subject to branching. If such column data is not referenced frequently, specify `BRANCH ALL`.

5.3 Designing a B-tree index

This section explains what must be considered when designing a B-tree index.

Since B-tree indexes greatly affect application performance, you must carefully choose the columns for which B-tree indexes will be defined.

Important

A B-tree index defined for a column store table can be used to search that column store table only in limited circumstances. The following are typical cases in which a B-tree index is used for a column store table. We recommend that you define a B-tree index for a column store table only when the range of the processing target can be narrowed as shown in these cases.

- Case where an index that will be used for index specification is specified when an SQL statement is executed
- Case where the minimum value is obtained by specifying the set function `MIN` or the maximum value is obtained by specifying the set function `MAX`
- Case where the `UPDATE` or `DELETE` statement is executed by specifying a search condition

In cases such as accessing data for a specific year or month, we recommend that you define range indexes and use them to narrow the search range.

For details about the rules that govern the use of B-tree indexes when retrieving data from column store tables, see *B-tree indexes and text indexes used during execution of SQL statements* in the *HADB Application Development Guide*.

5.3.1 Notes on defining B-tree indexes (unfinished status of B-tree indexes)

To define B-tree indexes, you enter the `CREATE INDEX` statement after creating the base table but before storing data.

If a B-tree index is defined for a base table for which row storage segments have been allocated, the defined B-tree index is placed in *unfinished status*. A B-tree index that is in unfinished status is not valid and cannot be used.

To release a B-tree index from unfinished status, you need to rebuild it. Therefore, determine carefully the columns for which B-tree indexes will be defined before storing data in a base table.

An error occurs if any of the following operations is performed on a B-tree index that is in the unfinished status.

■ Operations that will cause an error

- Executing a `SELECT` statement that uses a B-tree index in unfinished status
- Executing the `INSERT`, `UPDATE`, or `DELETE` statement on a table for which a B-tree index in unfinished status is defined
- Executing the `adbimport` command without the `-d` option specification on a base table for which a B-tree index in unfinished status is defined
- Executing the `adbmergechunk` command on a base table for which a B-tree index in unfinished status is defined

- Executing the `adbunarchivechunk` command on an archivable multi-chunk table for which a B-tree index in unfinished status is defined

For details about how to release a B-tree index from unfinished status, see 15.9.1 Steps to take when unfinished status is applied to a B-tree index.

> **Note**
>
> For example, row storage segments have not been allocated at the following times. If the `CREATE INDEX` statement is executed at these times, the B-tree index is created normally.
>
> - Immediately after defining a base table
> - Immediately after executing the `TRUNCATE TABLE` statement
>
> If the `DELETE` statement is used to delete all rows in a table, the row storage segments remain allocated. Therefore, after all table rows are deleted by using the `DELETE` statement, if the `CREATE INDEX` statement is executed, the B-tree index is placed in unfinished status.

5.3.2 Points to consider in determining the columns to be defined for a B-tree index

HADB uses a write-once updating method. Therefore, updating a column value updates the entire row (the existing row becomes invalid and a new row is inserted). When a B-tree index is defined for a table, updating a row also updates the B-tree index key value. If more B-tree indexes than necessary are defined for a table whose rows are frequently updated, the performance of update processing deteriorates accordingly.

When choosing the columns on which B-tree indexes will be defined, consider the points described in the following subsections.

(1) Columns that benefit from being defined for a B-tree index

The following columns benefit if B-tree indexes are defined on them:

Columns to be specified in the conditions used to narrow down data

When you perform a search with a search condition specified in a `WHERE` clause, the number of pages that must be searched can be reduced (narrowing down the search range) if you define a B-tree index on the column specified in the search condition. Since this improves search performance, consider defining a B-tree index on those columns specified in the search condition that can narrow down the data search range.

Columns to be specified in the conditions used to join tables (for searching across multiple tables)

When the search target data is distributed across multiple tables and columns having common information are mapped to each other when searching (by joining the tables), you can reduce the number of times the join condition must be evaluated (the number of times rows must be matched) if you define a B-tree index on the column specified in the join condition. Since this improves the search performance, consider defining a B-tree index on the column to be specified in the join condition.

If all of the following conditions are met, consider defining also a range index on the column to be specified in the join condition:

- The table for which a B-tree index is to be defined is a multi-chunk table.

- The column to be specified in the join condition applies to any of the cases shown in (1) Cases that benefit from a range index in 5.5.1 Points to consider when defining a range index.

(2) Columns that benefit from not being defined for a B-tree index

The following columns benefit if B-tree indexes are not defined on them:

Columns that contain a lot of redundant data

Even if you define a B-tree index on columns that contain a lot of redundant data, you cannot narrow down the data search range. Therefore, this does not reduce the number of pages that must be searched. To avoid search performance degradation, it is best not to define for a B-tree index columns that contain a large amount of redundant data.

5.3.3 Whether to use a single-column index or a multiple-column index

A B-tree index can be either a single-column index or multiple-column index.

- Single-column index

  A single B-tree index is created on a specific column in a table. Specify a single-column index when you want to retrieve data using a single column as the key.

- Multiple-column index

  A single B-tree index is created on multiple columns in a table.

This subsection explains when to use a single-column index, and when to use a multiple-column index.

(1) When it is better to define a single-column index

To retrieve data using a single column as the key, define a single-column index.

For example, for the retrieval described in the following, define a single-column index on column `C1`.

SQL specification example

```
SELECT * FROM "T1"
     WHERE "C1" = 1
```

B-tree index definition example

```
CREATE INDEX "T1IX_C1"
     ON "T1" ("C1" ASC) IN DBAREA01 EMPTY
```

(2) When it is better to define a multiple-column index

If using a single column would not sufficiently narrow down the rows to be retrieved, use multiple columns as keys to define a multiple-column index if doing so allows you to more efficiently narrow down the rows to be retrieved.

For example, for the retrieval described below, if a single-column index is defined on columns `C1` and `C2` separately, only one of the B-tree indexes for either column `C1` or `C2` can be used at any one time. For such cases, define a multiple-column index (`T1IX_C1C2`) consisting of columns `C1` and `C2`.

SQL specification example

```
SELECT * FROM "T1"
      WHERE "C1" = 1 AND "C2" = 'A'
```

B-tree index definition example

```
CREATE INDEX "T1IX_C1C2"
      ON "T1"("C1" ASC,"C2" ASC) IN DBAREA01 EMPTY
```

When a multiple-column index is used to determine whether values satisfy the search condition, they are evaluated according to the order in which the columns were defined in the multiple-column index. Therefore, when defining a multiple-column index, determine the order in which to specify the columns by considering factors such as the search conditions you will be specifying.

For example, to execute the SQL specification example shown above, we recommend that you execute the `CREATE INDEX` statement as described in the B-tree index definition example that is also shown.

(3) Notes on using a multiple-column index

This subsection explains the differences in how the data range is narrowed down based on the search conditions.

When using a multiple-column index to narrow down the rows to be searched, how the search range is narrowed down differs depending on the search conditions specified in the `WHERE` clause of the `SELECT` statement. When specifying the columns to index, specifying columns with frequent = *condition* specifications first will more effectively narrow down the search range.

The figure below shows the relationship between the specification of a search condition with a `WHERE` clause and the search range.

B-tree index definition example

```
CREATE INDEX "T1IX_C1C2C3"
        ON "T1"("C1" ASC,"C2" ASC,"C3" ASC) IN DBAREA01 EMPTY
```

Figure 5-6: Search condition with a WHERE clause and the search range of B-tree indexes

Index configuration example	B-tree index key values											
C1	1	1	1	1	2	2	2	2	3	3	3	3
C2	A	B	B	C	A	B	B	C	A	B	B	C
C3	10	10	20	10	10	10	20	10	10	10	20	10

WHERE clause search condition	Search range
WHERE "C1" = 2 AND "C2" = 'B' AND "C3" = 20	
WHERE "C1" = 2 AND "C2" >= 'B' AND "C3" = 20[#1]	
WHERE "C1" = 2 AND "C2" = 'B'	
WHERE "C1" = 2 AND "C2" >= 'B'	
WHERE "C1" = 2 AND "C3" = 20[#2]	
WHERE "C1" = 2	
WHERE "C1" >= 2 AND "C2" = 'B' AND "C3" = 20[#3]	
WHERE "C1" >= 2	
WHERE "C2" = 'B' AND "C3" = 20[#4]	
WHERE "C2" = 'B'[#4]	
WHERE "C3" = 20[#4]	

Legend: The shaded areas indicate the search range.

#1

Because the condition for column C2 is not = *condition* or IS NULL *condition*, the condition for column C3 is not used for narrowing down the search range.

#2

Because there is no condition for column C2, the condition for column C3 is not used for narrowing down the search range.

#3

Because the condition for column C1 is not = *condition* or IS NULL *condition*, the conditions for columns C2 and C3 are not used for narrowing down the search range.

#4

Because there is no condition for column C1, the conditions for columns C2 and C3 are not used for narrowing down the search range.

5.3.4 Allocating an unused area inside a B-tree index page (PCTFREE)

You use the PCTFREE operand of the CREATE INDEX statement to specify the percentage of the allocated area in a B-tree index page that is to remain unused.

If a B-tree index page that is being used to store index keys contains no unused area, a B-tree index page split will occur whenever a row is added to or updated in the table for which the B-tree index is defined. For details about B-tree index page splits, see (2) B-tree index page splits.

When a B-tree index page split occurs, the number of disk I/O operations increases, which might result in HADB performance degradation. Therefore, when defining a B-tree index for a table to which a row will be added using the `INSERT` data manipulation SQL statement or whose row will be updated using the `UPDATE` statement, allocate some unused area in the index page.

(1) Estimating the size of the unused area to be allocated in a B-tree index page

Use the following guidelines when allocating an unused area in a B-tree index page.

When you can estimate the number of rows that will be added or updated in the table

The number of index keys increases each time a row is added to, or updated in, the table. For the unused area, specify the value determined using the following formula.

Formula (%)

Unused area =
↑ (*number of rows that will be added or updated* / (*number of rows when data page is built* + *number of rows that will be added or updated*)) x 100 ↑

number of rows when data page is built means the number of table data rows that are stored when the `adbimport` command is first executed to populate the table with data.

When you cannot estimate the number of rows that will be added or updated in the table

Specify `30` (%) (default value of the `PCTFREE` operand) for the unused area.

When rows will not be added or updated in the table (only referenced)

Specify `0` (%) for the unused area.

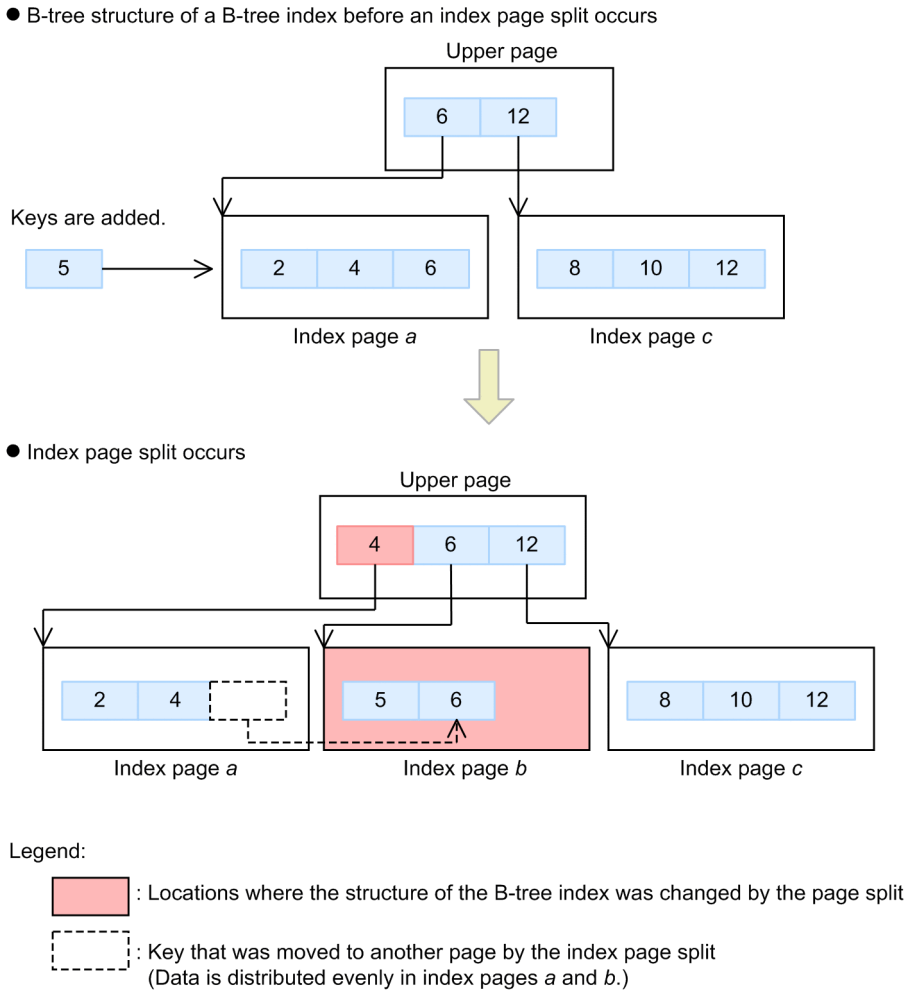
When the following commands are executed, data is stored, leaving free the percentage of unused area specified in the `PCTFREE` operand:

- `adbimport` command
  - When data is imported in creation mode (when the `-d` option is specified)
  - When the background-import facility is used (when the `-b` option is specified)
  - When data is imported in addition mode (when neither the `-d` option nor the `-b` option is specified)
- `adbidxrebuild` command
- `adbmergechunk` command
- `adbunarchivechunk` command

(2) B-tree index page splits

A B-tree index determines the range of keys managed by each page. If you add a key to an index page for storing keys that has no free space, the key group managed by the page is split in two. This is called a *B-tree index page split*. The following figure shows an example of a B-tree index page split.

Figure 5-7: Example of B-tree index page splits



5.3.5 Points to consider in determining the columns to be defined for a unique index

A *unique index* is a B-tree index in which no column on which the B-tree index is defined contains any duplicated key values. This restriction on key value duplication is called the *uniqueness constraint*.

When you need to ensure that a column on which a B-tree index is defined will not have any duplicated key values, consider defining a unique index.

> **Note**
>
> A unique index is also automatically defined when a primary key (uniqueness constraint definition) is defined during base table definition. For details about primary keys, see 5.2.9 Specifying a primary key (uniqueness constraint definition) (PRIMARY KEY) [Single-chunk table].

In the following cases, key values are not considered to be duplicated even if there are duplicate key values (the uniqueness constraint is not violated in these cases):

- The duplicate key value is the null value.

- In a multiple-column index, at least one of the columns has a different value.

You define an index to be a unique index by specifying the UNIQUE operand in the CREATE INDEX statement. When you specify the UNIQUE operand, you cannot perform a data addition or data updating operation that will result in a duplicated key value. Also, if you attempt to add a row that contains a particular key value, you will not be allowed to do so if another executing transaction has already added a row with that same key value.

Important

Even when a unique index is defined, the uniqueness constraint cannot be guaranteed in the following cases (key value duplication might occur):

- Data is imported by the adbimport command.
- A B-tree index is rebuilt by the adbidxrebuild command.

5.3.6 Setting a null-value exclusion specification (EXCLUDE NULL VALUES)

If you set a null-value exclusion specification when you define a B-tree index, a B-tree index is created that does not contain keys comprised of null values only (keys comprised of null values only are not created).

For example, if you set as an indexed column one that is comprised mostly of null values, consider setting a null-value exclusion specification to the index. When you set the null-value exclusion specification, keys comprised of null values only are not created, and therefore you can shorten the time required for creating a B-tree index. You can also reduce the data size of the B-tree index.

You can set the null-value exclusion specification by specifying EXCLUDE NULL VALUES when you use the CREATE INDEX statement to define a B-tree index.

Note that if the table retrieval range includes null values, a B-tree index for which the null-value exclusion specification is specified does not become a candidate index to be used for the retrieval. Examples include cases in which no search condition is specified, and cases in which the IS NULL predicate is specified.

5.4 Designing a text index

This section explains what must be considered when designing a text index.

You use the CREATE INDEX statement to define text indexes.

> **Important**
>
> You cannot define a text index for a column store table.

5.4.1 Points to consider in determining the columns to be defined for a text index

When choosing the columns on which text indexes will be defined, consider the points described in the following subsections.

(1) Cases that benefit from a text index

Consider defining a text index when you will be retrieving data in the following ways:

- When using the LIKE predicate to perform middle match or trailing match retrievals

  If you use a text index, there will be fewer pages to be loaded than if you were using a B-tree index. This means that you can achieve higher-speed retrievals by defining a text index than when you define a B-tree index. Define in the LIKE predicate a text index for a column that is specified as a match value.
- When executing retrieval operations with the LIKE_REGEX predicate specified

  Using a text index requires fewer pages to be loaded. This means that you can achieve faster retrieval by defining a text index.
- When executing retrieval operations with the CONTAINS scalar function specified

  Using a text index requires fewer pages to be loaded. This means that you can achieve faster retrieval by defining a text index.
- When performing a correction search

  Using a text index for correction search requires fewer pages to be loaded. This means that you can achieve faster retrieval by defining a text index for correction search. For details about correction search, see 2.17.1 Correction search.
- When performing word-context search

  Using a text index for a word-context search requires fewer pages to be loaded. This means that you can achieve faster retrieval by defining a text index for a word-context search. For details about word-context search, see 2.17.4 Word-context search.

The following table explains the relationship between the data to be retrieved and text indexes.

Table 5-4: Relationship between the data to be retrieved and text indexes

No.	Retrieved data items	Validity of text index	Notes
1	Document data (such as document data and mail containing at least 100 characters)	V	If a character string is long, the data size for the text index might become large.

No.	Retrieved data items	Validity of text index	Notes
			For document data, the data size for the text index tends to become large compared to other retrieval target data because data for the text index includes such character strings that are not used as search conditions.
2	Document data written in English (such as document data and emails containing at least 100 characters)	V	When searching for English words in documents written in English, specify a text-index-word-context search specification and define a text index. To conduct the word-context search, use the `CONTAINS` scalar function. For details about text-index-word-context search specification, see 5.4.5 Specifying a text index for a word-context search (TEXT WORDCONTEXT). Note that if a character string is long, the data size for the text index might become large. For document data, the data size for the text index tends to be larger than for other retrieval target data because data for the text index includes character strings that unlikely to ever be used as search conditions. The data size for the text index will also be larger when you specify a text-index-word-context search specification than when you do not.
3	Character strings consisting of only alphanumeric characters (such as URLs and access logs)	L	Use of text indexes might not yield any benefit in the following cases: • Data is extremely short. • The variation among characters used is small.
4	Name data (such as product names consisting of about 10 characters)	U	Use of a text index might not yield any benefit in the following case: • Each data item consists of about 10 characters, but there are many identical name data items, such as in a history of purchases.
5	Character strings consisting of only numeric characters (such as IDs and codes)	N	Even if retrieval results are unique, the retrieval processing might require a large amount of time because there are not many character combinations (inasmuch as all character strings consist of only numeric characters). Little benefit can be expected from using text indexes.

Legend:

V: Very likely to be valid

L: Likely to be valid

U: Unlikely to be valid

N: Not valid

(2) Cases that benefit when a text index is not defined

Do not define a text index for columns with the following characteristics:

- There are many duplications in the column data (many partial or complete matches).
- The number of different characters is limited, such as a number column containing only numeric digits (for example, IDs).
- The character strings to be retrieved are short and simple, such as the alphabetic character `a` or the numeric digit `0`.
- The character strings to be retrieved are too long (1,000 characters or more).

If a text index is defined for such a column, the number of pages to be retrieved cannot be reduced because the data retrieval range cannot be narrowed. No benefit can be expected from using a text index.

(3) Columns for which text indexes can be defined

Text indexes can be defined only for columns that satisfy the following conditions:

Columns for which text indexes can be defined

- `CHARACTER` type column
- `VARCHAR` type column

5.4.2 Allocating an unused area inside a text index page (PCTFREE)

You use the `PCTFREE` operand of the `CREATE INDEX` statement to specify the percentage of the allocated area in a text index page that is to remain unused.

If a text index page that is being used to store index keys contains no unused area, a text index page split will occur whenever a row is added to or updated in the table for which the text index is defined. For details about index page splits of text indexes, see (2) Text index page splits.

When a text index page split occurs, the number of disk I/O operations increases, which might result in HADB performance degradation. Therefore, when defining a text index for a table to which a row will be added using the `INSERT` data manipulation SQL statement or whose row will be updated using the `UPDATE` statement, allocate some unused area in the index page.

(1) Estimating the size of the unused area to be allocated in a text index page

Use the following guidelines when allocating an unused area in a text index page.

When you can estimate the number of rows that will be added or updated in the table

The number of index keys increases each time a row is added to, or updated in, the table. For the unused area, specify the value determined using the following formula.

Formula (%)

Unused area =
↑ (*number of rows that will be added or updated* / (*number of rows when data page is built* + *number of rows that will be added or updated*)) x 100 ↑

number of rows when data page is built means the number of table data rows that are stored when the `adbimport` command is first executed to populate the table with data.

When you cannot estimate the number of rows that will be added or updated in the table

Specify `30` (%) (default value of the `PCTFREE` operand) for the unused area.

When rows will not be added or updated in the table (only referenced)

Specify `0` (%) for the unused area.

When the following commands are executed, data is stored, leaving free the percentage of unused area specified in the `PCTFREE` operand:

- `adbimport` command

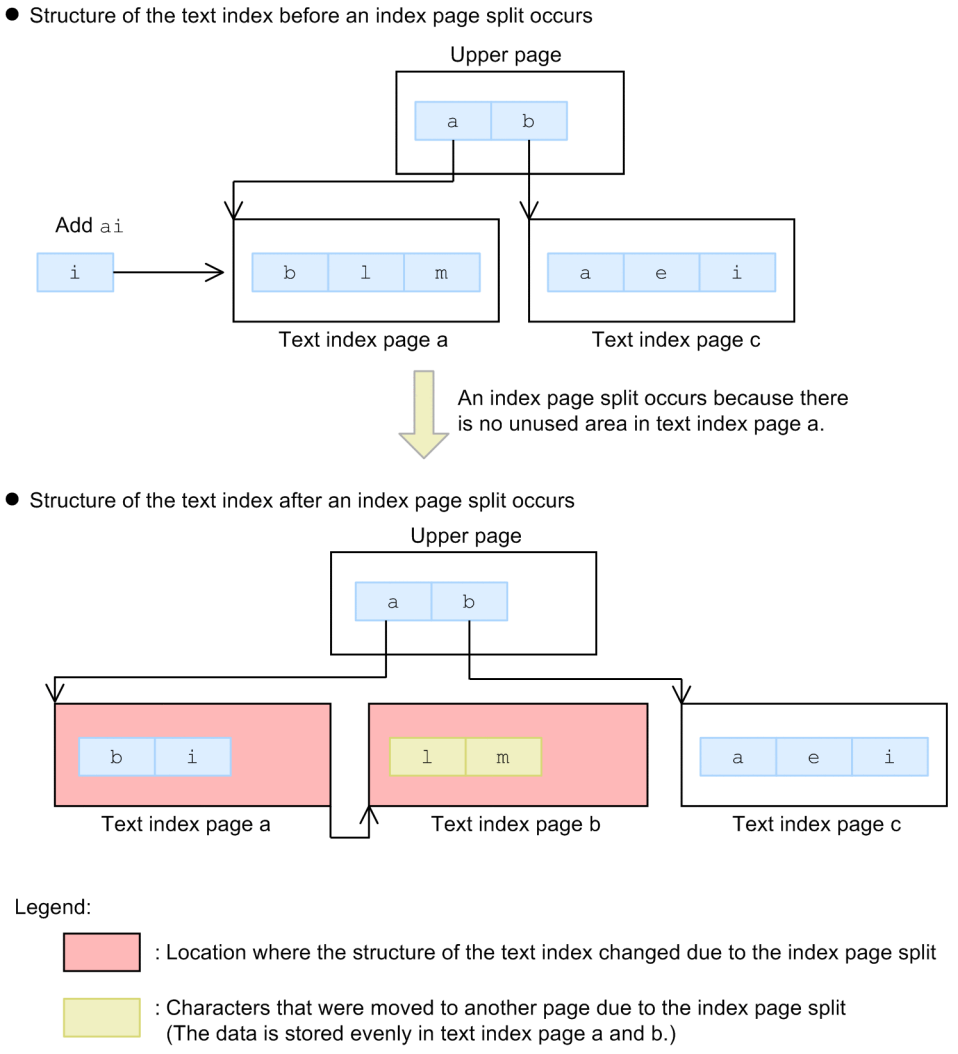
- When data is imported in creation mode (when the `-d` option is specified)
- When the background-import facility is used (when the `-b` option is specified)
- When data is imported in addition mode (when neither the `-d` option nor the `-b` option is specified)

- `adbidxrebuild` command
- `adbmergechunk` command
- `adbunarchivechunk` command

(2) Text index page splits

When data is added to or updated in a base table for which a text index has been defined, data is added to an index page of the text index. If the index page has no room for the data, a new index page is allocated for the text index. This is called a *text index page split*. The following figure shows an example of a text index page split.

Figure 5-8: Example of a text index page split



5.4.3 Notes on defining text indexes (unfinished status of text indexes)

To define text indexes, you enter the `CREATE INDEX` statement after creating the base table but before storing data.

If a text index is defined for a base table for which row storage segments have been allocated, the defined text index is placed in unfinished status. A text index in unfinished status cannot be used because it is invalid.

To release a text index from unfinished status, you need to rebuild it. Therefore, determine carefully the columns for which text indexes will be defined before storing data in a base table.

An error occurs if any of the following operations is performed on a text index that is in the unfinished status.

■ Operations that will cause an error

- Executing a `SELECT` statement that uses a text index in unfinished status
- Executing the `INSERT`, `UPDATE`, or `DELETE` statement on a base table for which a text index in unfinished status is defined
- Executing the `adbimport` command without the `-d` option specification on a base table for which a text index in unfinished status is defined
- Executing the `adbmergechunk` command on a base table for which a text index in unfinished status is defined
- Executing the `adbunarchivechunk` command on an archivable multi-chunk table for which a text index in unfinished status is defined

For details about how to release a text index from unfinished status, see 15.10.1 Steps to take when unfinished status is applied to a text index.

> **Note**
>
> For example, row storage segments have not been allocated at the following times. If the `CREATE INDEX` statement is executed at these times, the text index is created normally.
>
> - Immediately after defining a base table
> - Immediately after executing the `TRUNCATE TABLE` statement
>
> If the `DELETE` statement is used to delete all rows in a table, the row storage segments remain allocated. Therefore, after all table rows are deleted by using the `DELETE` statement, if the `CREATE INDEX` statement is executed, the text index is placed in unfinished status.

5.4.4 Notation-correction-search text-index specification (CORRECTIONRULE)

To use correction search, define a text index that supports correction search. For details about correction search, see 2.17.1 Correction search.

If you define a text index that supports correction search, you can reduce the number of pages to be loaded when you perform the following search operations. This improves table retrieval performance.

- Perform correction search by specifying the scalar function `CONTAINS`.
- Use a regular expression to perform case-insensitive search by specifying `IGNORECASE` (or `I`) for `FLAG` in the `LIKE_REGEX` predicate.

To define a text index that supports correction search, you specify `CORRECTIONRULE` (notation-correction-search text-index specification) in a `CREATE INDEX` statement.

Note that the index data in a text index that supports correction search is larger than the index data in an ordinary text index. Therefore, before you define a text index, obtain the size of index data, and confirm that you can define a text index without problems. For details, see 5.8.5 Determining the number of storage pages for each text index segment.

Important

If the character encoding used on the HADB server is Shift-JIS (that is, the value specified for the environment variable ADBLANG is SJIS), you cannot define a text index that supports correction search.

A text index that supports correction search can also be used as an ordinary text index.

5.4.5 Specifying a text index for a word-context search (TEXT WORDCONTEXT)

To use word-context search, define a text index that supports word-context searches. For details about word-context search, see 2.17.4 Word-context search.

By defining a text index for a word-context search, you can reduce the number of pages to be loaded when you perform a word-context search by specifying the CONTAINS scalar function. This improves table retrieval performance.

To define a text index for a word-context search, specify TEXT WORDCONTEXT for INDEXTYPE in the CREATE INDEX statement. Specifying TEXT WORDCONTEXT is called text-index-word-context search specification.

Note

When defining a text index for a word-context search, you need to specify a value for DELIMITER (text-index delimiter specification). For details about DELIMITER, see 5.4.6 Selecting the delimiting characters for word-context searches (DELIMITER).

Important

Do not define a text index for a word-context search if you will be performing leading-match word-context searches that specify short character strings of one or two characters. Using a text index for a word-context search in this scenario will actually lower performance because the data retrieval range cannot be narrowed. If you intend to perform leading-match word-context searches that specify short character strings of one or two characters, use a search method other than a word-context search.

Note that the index data in a text index for a word-context search is larger than the index data in an ordinary text index. Therefore, before you define the text index, determine the size of the index data and confirm that the additional size will not cause any problems. For details, see 5.8.5 Determining the number of storage pages for each text index segment.

A text index for a word-context search can also be used as an ordinary text index.

5.4.6 Selecting the delimiting characters for word-context searches (DELIMITER)

When defining a text index for a word-context search, you need to specify the types of delimiting characters that delimit one word from another. For details about text indexes that support word-context search, see 5.4.5 Specifying a text index for a word-context search (TEXT WORDCONTEXT).

To specify the delimiting characters, you use the DELIMITER (text-index delimiter specification) operand of the CREATE INDEX statement. You must specify DELIMITER if you specify TEXT WORDCONTEXT for INDEXTYPE.

As the delimiting character type, you can specify DEFAULT or ALL. You must specify one or the other when specifying DELIMITER. The following shows which characters are handled as delimiting characters when DEFAULT is specified and when ALL is specified.

DEFAULT

When performing a word-context search, the following characters are handled as delimiting characters:

- Single-byte spaces (0x20)
- Tabs (0x09)
- Line feeds (0x0A)
- Carriage returns (0x0D)
- Periods (0x2E)
- Question marks (0x3F)
- Exclamation marks (0x21)

ALL

When performing a word-context search, the following characters are handled as delimiting characters:

- Single-byte spaces (0x20)
- Tabs (0x09)
- Line feeds (0x0A)
- Carriage returns (0x0D)
- Half-width symbols including periods, question marks, and exclamation marks (0x21 to 0x2F, 0x3A to 0x40, 0x5B to 0x60, and 0x7B to 0x7E)

The result of a word-context search of English text that contains symbols differs according to the delimiting characters. Keep this in mind when selecting the type of delimiting characters. The following explains cases that benefit from specifying DEFAULT, and cases that benefit from specifying ALL.

■ Cases that benefit from specifying DEFAULT

When using a word-context search to search for data that includes symbols, we recommend that you specify DEFAULT. Examples are cases in which you are searching for URLs and email addresses. If you specify DEFAULT and search with taro@hitachi.com as the search term, taro@hitachi is handled as one word. This allows you to retrieve only data that contains taro@hitachi.

If you specify ALL, taro and hitachi are handled as separate words. This means that the word-context search will also retrieve data such as xxxxx@taro.hitachi.com and taro.hitachi@com.

■ Cases that benefit from specifying ALL

When using a word-context search to search for data that does not include symbols, we recommend that you specify ALL. Examples are cases in which you are searching for phrases like high△speed that consist of multiple words. If you specify ALL and search with high△speed as the search term, the word-contest search will retrieve not only high△speed but also similar phrases such as high-speed and high_speed. Here, △ represents a half-width space.

■ Notes

The DELIMITER specification might be ignored if the table targeted by the word-context search is an internal derived table.

- If the derived table is expanded

  The DELIMITER specification is valid for the text index for the word-context search defined for columns in the result of expanding the internal derived table.

- If the derived table is not expanded

  DELIMITER is invalid if ALL is specified. The word-context search will be executed subject to the same delimiting characters as if DEFAULT were specified for DELIMITER.

Note that this rule does not apply to internal derived tables obtained by equivalent exchange of an archivable multi-chunk table.

For details about the rules for derived table expansion, see *Internal derived tables* in *Constituent Elements* in the manual *HADB SQL Reference*.

5.5 Designing a range index

This section explains what must be considered when designing a range index.

The `CREATE INDEX` statement is used to define range indexes.

5.5.1 Points to consider when defining a range index

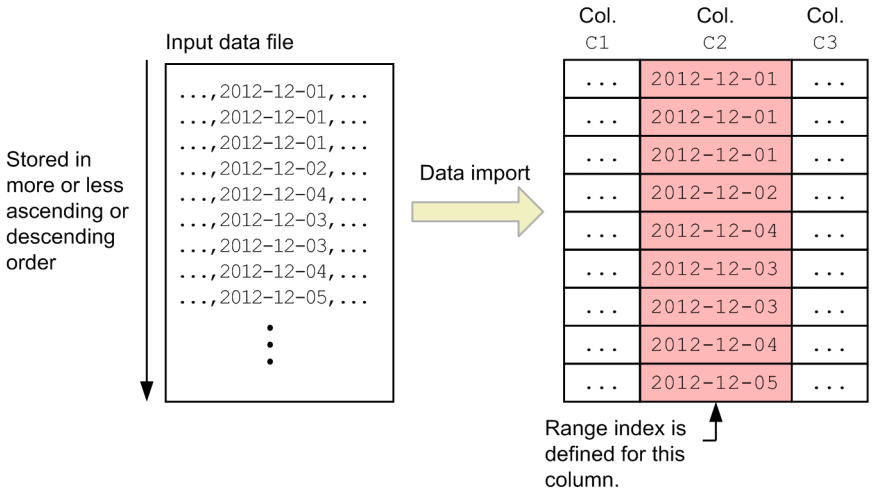
(1) Cases that benefit from a range index

In a search that uses a range index, the data to be searched is narrowed based on ranges of values of the data in the column on which the range index has been defined. Therefore, define a range index in the following cases:

- **When there is a column whose data is stored in the input data file in ascending or descending order (or more or less in ascending or descending order)**

  For example, if a table contains a column that stores the data acquisition date, and typically these data acquisition dates are stored in the input data files in ascending or descending date order, defining a range index on that column will be beneficial. The following figure shows an example.

  Figure 5-9: Case that benefits when a range index is defined (1 of 3)

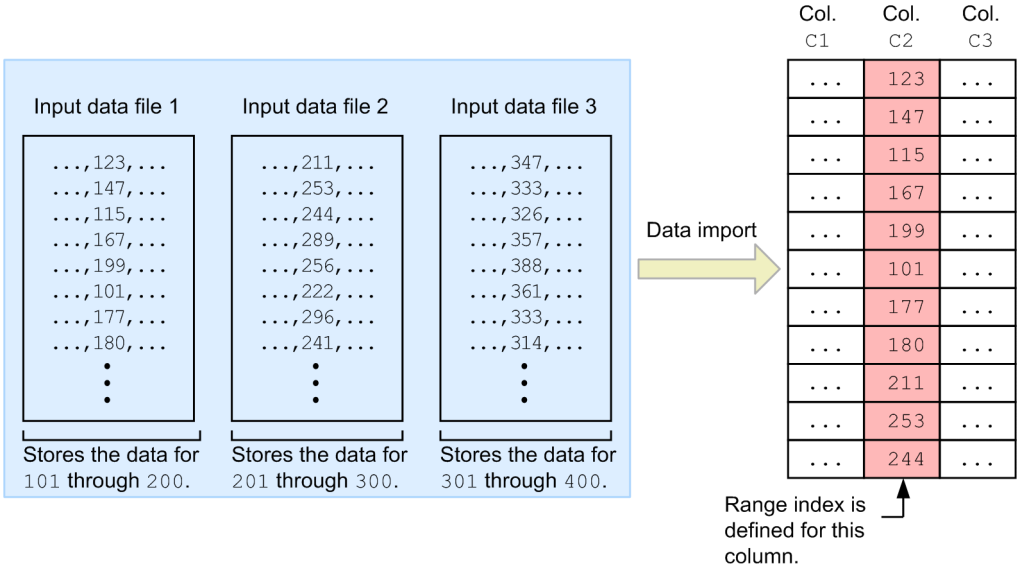


  When searching a table, specify as a search condition a column for which a range index is defined (column `C2` in the preceding figure). However, depending on the specified search condition, the range index might not be used. For details about the conditions under which range indexes are used, see *Range indexes used during execution of SQL statements* in the manual *HADB Application Development Guide*.

- **When there is a column in all the input data files that always contains a predetermined range of data values**

  For example, if a table contains a column that stores product numbers and the ranges of product number values is more or less predetermined in all the input data files, defining a range index on that column will be beneficial. The following figure shows an example.

Figure 5-10: Case that benefits when a range index is defined (2 of 3)

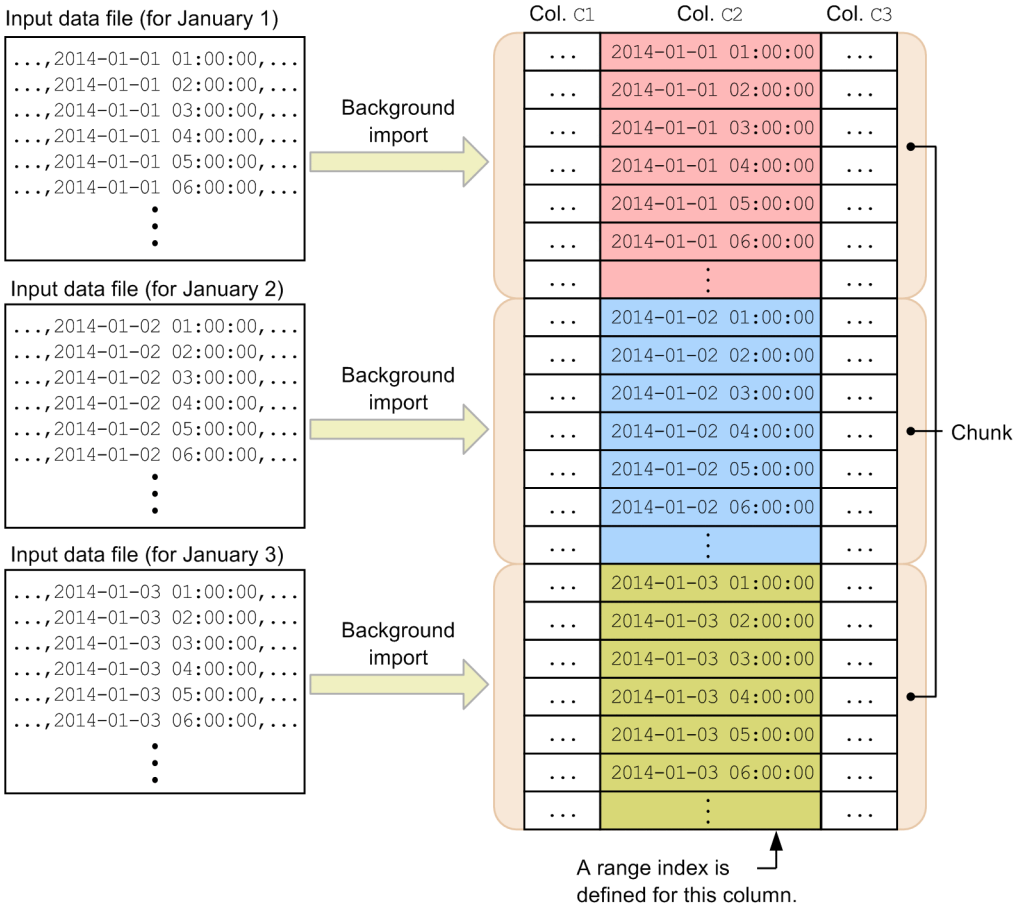


When searching a table, specify as a search condition a column for which a range index is defined (column `C2` in the preceding figure). However, depending on the specified search condition, the range index might not be used. For details about the conditions under which range indexes are used, see *Range indexes used during execution of SQL statements* in the manual *HADB Application Development Guide*.

- **When data to be stored using background import is not duplicated among chunks**

  Consider a case in which data collected by a sensor (a day's worth of data) is stored daily in a table, using background import. If the table targeted for background import contains a column that stores the data generation date and time, defining a range index for that column is effective. The following figure shows an example.

Figure 5-11: Case that benefits when a range index is defined (3 of 3)



Explanation

Each input data file stores a day's worth of data. When background import is executed, the data representing the data generation date and time stored in column C2 is not duplicated among the individual chunks.

In this case, because the range is managed for each chunk, retrieval processing of chunks that store ranges of data that do not satisfy the search condition can be skipped. This allows retrieval to take place efficiently.

When searching a table, specify as a search condition a column for which a range index is defined (column C2 in the preceding figure). However, depending on the specified search condition, the range index might not be used. For details about the conditions under which range indexes are used, see *Range indexes used during execution of SQL statements* in the manual *HADB Application Development Guide*.

(2) Cases that benefit when a range index is not defined

In the following cases, the range of each segment expands, with the result that defining a range index does not provide the benefit of segment skipping.

- When there are no columns in which the data in the input data files is arranged in more or less ascending or descending order
- When rows will be subject to frequent addition, update, and deletion processing

When rows are deleted, although the ranges of the values in the segments do not expand, the range of the range index itself becomes greater compared with the ranges of the data stored in the segments. In such a case, the benefits of using a range index will not be realized.

In the following case, the ranges of individual chunks overlap with each other, with the result that defining a range index does not provide the benefit of chunk skipping.

- When data to be stored using background data import is duplicated among chunks

Do not define a range index for any column that you do not expect to benefit from segment skipping or chunk skipping. Doing so reduces search performance.

(3) Columns for which a range index cannot be defined

You cannot define a range index based on the following types of columns:

- `CHAR` column whose definition length is 33 bytes or greater
- `VARCHAR` column
- `BINARY` column
- `VARBINARY` column

5.5.2 Estimating the number of pages for the global buffer used exclusively for range indexes

You use the `adbbuff` operand in the server definition to specify the global buffer to be allocated to the DB area for storing range indexes. When doing so, you need to be prepared to estimate accurately the number of pages to be allocated for the global buffer that will be used exclusively for range indexes. You will specify this value in the `-a` option of the `adbbuff` operand. For details about how to estimate the number of pages, see the explanation of the -a option of the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

5.5.3 Pre-reading of range indexes

You can load all range indexes into the global buffer when you start the HADB server. This is called *pre-reading of range indexes*. To pre-read range indexes, specify the `adb_sql_rngidx_preread` operand in the server definition.

(1) Benefit of pre-reading of range indexes

Normally, at the time of the first table search after the HADB server has started, the global buffer will not yet contain any range index data. Consequently, an I/O operation will be required to load data into the global buffer, which will degrade search performance. You can eliminate this I/O operation by pre-reading range indexes.

(2) Preventing pre-read range index data from being flushed out

Pre-read range index data is also a target of flushing from the global buffer. You can prevent such flushing from occurring by taking the following steps:

- Store pre-read range index data that you do not want to be flushed in a dedicated data DB area.
- Estimate accurately the number of pages in the global buffer to be allocated to the data DB area that stores the pre-read range index data that you do not want to be flushed out (that is, estimate accurately the value to be specified in the `-a` option of the `adbbuff` operand in the server definition).

For details about the formula for estimating the value to be specified in the -a option, see the explanation of the -a option of the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

5.5.4 Notes on defining range indexes (unfinished status of range indexes)

You define a range index by entering the CREATE INDEX statement after creating the base table but before storing data, in the same manner as when you define a B-tree index.

If a range index is defined for a base table for which row storage segments have been allocated, the defined range index is placed in *unfinished status*. A range index that is in unfinished status is not valid and cannot be used.

To release a range index from unfinished status, you need to rebuild it. Therefore, determine carefully the columns for which range indexes will be defined before you store any data in the base table.

An error occurs if any of the following operations is performed on a range index that is in unfinished status.

■ Operations that will cause an error

- Executing a SELECT statement that uses a range index that is in unfinished status
- Executing the INSERT or UPDATE statement on a table for which a range index in unfinished status is defined
- Executing the adbimport command without the -d option specification on a base table for which a range index in unfinished status is defined
- Executing the adbmergechunk command on a base table for which a range index in unfinished status is defined
- Executing the adbunarchivechunk command on an archivable multi-chunk table for which a range index in unfinished status is defined

For details about how to release a range index from unfinished status, see 15.11.1 Steps to take when unfinished status is applied to a range index.

Note

For example, row storage segments have not been allocated at the following times. If the CREATE INDEX statement is executed at these times, the range index is created normally.

- Immediately after defining a base table
- Immediately after executing the TRUNCATE TABLE statement

If the DELETE statement is used to delete all rows in a table, the row storage segments remain allocated. Therefore, after all table rows are deleted by using the DELETE statement, if the CREATE INDEX statement is executed, the range index is placed in unfinished status.

5.6 Designing a data DB area

A data DB area consists of segments and pages. To improve the efficiency of data storage, it is important to design a page size based on appropriate factors, because the segment size is set automatically on the basis of the page size.

An inappropriate page size will result in poor data access efficiency and degraded performance.

Consider the following points when designing a data DB area:

- The number of tables (indexes) that will be stored and the data DB areas in which they will be stored
- The number of data DB area files that will make up one data DB area
- Whether to store multi-chunk tables in the data DB area
- Whether to store both column store tables and row store tables in the same data DB area
- The size of each page

Details regarding these points are explained in the following sections:

> **Note**
>
> Once you have designed the page size, estimate the size of the data DB area. For details about how to estimate the size of the data DB area, see 5.8 Estimating the size of the data DB area.

5.6.1 Points to consider when designing a data DB area

This subsection describes points to consider when designing a data DB area.

If you will be storing a multi-chunk table in the data DB area, also read the explanation in 5.6.2 Points to consider in storing a multi-chunk table in the data DB area.

(1) Points to consider about the data DB area for storing tables and indexes

Consider the following points when storing tables and indexes in a data DB area:

- If a table to be stored in a data DB area has a large amount of data, store only the table in the data DB area.
- If an index to be stored in a data DB area has a large amount of data, store only the index in the data DB area.
- If tables or indexes to be stored in data DB areas have a small amount of data (for example, several megabytes), storing multiple tables or indexes in a single data DB area causes no problem.
- When defining a column store table, prepare a dedicated data DB area for storing only that column store table.

We do not recommend that you store multiple tables or indexes that have a large amount of data in a single data DB area. If you do so and perform a search such as shown below, I/O operations might concentrate on the same DB area, thus degrading search performance:

- Multiple tables are stored in a single data DB area and a search using an SQL statement that joins tables is executed
- Multiple indexes are stored in a single data DB area and a search that uses indexes is executed

Issues like the following might also arise:

- While an operation (SQL statement or command) is being executed on a particular table, a lock placed on a data DB area prevents operations from being executed on other tables stored in the same data DB area.
- When sharing a global buffer, execution of an SQL statement for a particular table impacts the data retrieval performance for other tables stored in the same data DB area.

  The performance of data retrieval in row store tables or by using indexes might be degraded, especially in the following case: the same global buffer is allocated to a data DB area that stores column store tables and to a data DB area that stores row store tables and indexes.

(2) Points to consider about the data DB area for storing range indexes

Consider the following points, which are unique to range indexes:

We recommend that you not store any other tables or indexes in a data DB area that stores tables for which range indexes are defined. Storing other tables or indexes might increase the range index size.

(3) Points to consider about the data DB area files that comprise a data DB area

If one data DB area consists of multiple data DB area files, the workload can be distributed because the data DB area files are accessed in parallel. For this reason, consider using multiple data DB area files for a data DB area that stores tables and indexes that are accessed frequently. A guideline for the number of data DB area files to make up one data DB area is 20 to 40 percent of the CPU cores in the machine on which the HADB server is installed.

The following are benefits of storing multiple data DB areas files in one data DB area:

- When executing the `adbimport` command to import data

  One processing real thread is allocated to each data DB area file, so data storage processes and index creation processes are performed in parallel.
- When allocated segments are released

  One processing real thread is allocated to each data DB area file, so segment release processes are performed in parallel.

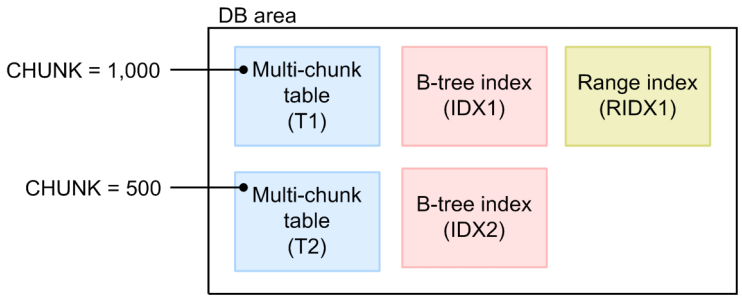
5.6.2 Points to consider in storing a multi-chunk table in the data DB area

When you store multi-chunk tables and the indexes defined for those tables in one data DB area, consider the following points:

- The maximum number of chunks that can be managed in one data DB area is 30,000.
- Take into account the number of chunks in indexes, in addition to the number of chunks in tables.

Example

Assume that you store multi-chunk tables and indexes in one data DB area as follows:



- The B-tree index `IDX1` and the range index `RIDX1` are defined for multi-chunk table `T1`. The maximum number of chunks specified for the multi-chunk table `T1` is 1,000.
- The B-tree index `IDX2` is defined for multi-chunk table `T2`. The maximum number of chunks specified for multi-chunk table `T2` is 500.

In this case, the maximum number of chunks to be created in `T1`, `IDX1`, and `RIDX1` is 3,000. The maximum number of chunks to be created in `T2` and `IDX2` is 1,000.

1,000 (`T1`) + 1,000 (`IDX1`) + 1,000 (`RIDX1`) = 3,000 (chunks)

500 (`T2`) + 500 (`IDX2`) = 1,000 (chunks)

Therefore, the maximum number of chunks to be created in the data DB area is 4,000.

The total number of chunks cannot exceed 30,000.

A chunk is created for a stored table or index when background import is executed, enabling you to store multiple tables and indexes in a single data DB area. When storing multiple tables or indexes, do not exceed the maximum number of chunks that a single data DB area can manage.

Note

For details about how to determine the maximum number of chunks, see 5.2.4 Points to consider in defining a multi-chunk table.

▪ For archivable multi-chunk tables

For archivable multi-chunk tables, also consider the following points:

- **Number of chunks in the range index that is automatically defined for the archive range column**

  Chunks are also created for the range index that is automatically defined for the archive range column. Therefore, you also need to calculate the number of chunks in this range index.

- **Number of chunks in a location table**

  An archivable multi-chunk table and its location table are stored in the same data DB area. Chunks are also created for this location table. Therefore, you also need to calculate the number of chunks in the location table. Use the following formula to determine the number of chunks in a location table:

  Formula

  Number of chunks in a location table =
  Number of archivable multi-chunk tables stored in a data DB area x 10

5.6.3 Points to consider when determining the page size in data DB areas

This subsection explains the points to be considered when determining the page size in data DB areas.

You can use one of the following methods to specify the page size of a data DB area:

- Specifying the page size in kilobytes for the `-p` option of the `adbinitdbarea` initialization option in the `adbinit` command
- Specifying the page size in kilobytes for the `-p` option of the `adbaddarea` DB area addition and modification option in the `adbmodarea` command

(1) Data DB area for storing tables

Consider the following points when determining the page size for a DB area in which tables are to be stored:

- The page size must be large enough to store base rows. When storing multiple tables in a DB area, specify a page size that can store the longest base row in any of the tables. If the page size is not long enough to store base rows, an error occurs during table definition. For details about how to determine the length of base rows, see 5.8.2 Determining the number of pages for storing each type of row.
- The maximum number of rows that can be stored in a page is 255. Therefore, if the size of the page is larger than the total length of the rows, excess space that cannot be used is allocated, reducing the data storage efficiency. For example, if the row length is 100 bytes, the maximum amount of data that can be stored in a single page is 25,500 bytes (that is, 100 bytes × 255 rows). In this case, if the page size is set to 32,768 bytes (32 kilobytes), an area of approximately 7,000 bytes (32,768 bytes - 25,500 bytes - *management area byte count*) cannot be used and is wasted.

(2) Data DB area for storing B-tree indexes

For the page size, specify 8 or 16 kilobytes.

(3) Data DB area for storing text indexes

For the page size, specify 8 kilobytes.

(4) Data DB area for storing range indexes

For the page size, specify the maximum value, which is 32 kilobytes.

5.7 Designing a work table DB area

A work table DB area is used for storing work tables. A work table is created when you execute an SQL statement that creates a work table.

You specify the page size (in kilobytes) of the work table DB area in the `adb_init_wrk_page_size` operand, which is an initialization option in the `adbinit` command. For the page size of the work table DB area, specify 256 kilobytes in most cases.

If the page size of the work table DB area is too small, it will be unable to store work tables with long row lengths. An error might occur depending on the SQL statement that is executed. An example in which an error would occur is when executing a SQL statement that results in a work table with a long row length because the columns that constitute the work table include columns with long definition lengths. Therefore, you must consider the row length of the resulting work table when specifying the page size of the work table DB area. For details about how to determine the row length of work tables, see 5.9.2 Determining the number of pages for base rows that are needed for storing work tables.

To change the page size of a work table DB area, specify the new page size in kilobytes for the `adb_dbarea_wrk_page_size` operand in the server definition. Then, restart the HADB server.

For details about the SQL statements that create work tables, see *Work tables created when SQL statements are executed* under *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

5.8 Estimating the size of the data DB area

Use the following formula to determine the data DB area size.

Formula (kilobytes)

```
data DB area size = DATA_PAGE_NUM × page_size ÷ 1,024
```

Explanation of variables

DATA_PAGE_NUM: Total number of pages in the data DB area

page_size: Size of the pages in the data DB area (bytes)

For details about *DATA_PAGE_NUM*, see 5.8.1 Determining the total number of pages in the data DB area.

5.8.1 Determining the total number of pages in the data DB area

Use the following formula to determine the total number of pages in the data DB area (*DATA_PAGE_NUM*).

(1) Formula

The following shows the formula for determining the total number of pages in the data DB area (*DATA_PAGE_NUM*).

Formula

$$DATA\_PAGE\_NUM = 1 + dbarea\_file\_num \times \left(21 + \left\uparrow \frac{32{,}000}{\left\downarrow \frac{page\_size}{160} \right\downarrow} \right\uparrow \right) + \left\uparrow \frac{SGDATA}{SEGBF} \right\uparrow + \left\uparrow \frac{SGDATA}{\left\downarrow \frac{page\_size \times 125}{16} \right\downarrow \times SEGBF} \right\uparrow + SGDATA \times SEGSIZE$$

(2) Explanation of variables

The following explains the variables that are used to determine the total number of pages in the data DB area (*DATA_PAGE_NUM*).

Explanation of variables

dbarea_file_num

Number of files in the data DB area (number of files)

page_size

Page size (bytes)

SEGBF

Segment blocking factor

This value differs for each data DB area page size. Determine it from the following table.

Table 5-5: Segment blocking factor for each data DB area page size

No.	Page size (kilobytes)	Segment blocking factor
1	4	12
2	8	42
3	16	127
4	32	341

SEGSIZE

Segment size (pages)

Use the following formula to determine this value.

```
SEGSIZE = 4,194,304 ÷ page_size
```

SGDATA

Number of segments in the data DB area

Use the following formula to determine this value.

```
SGDATA= SGROWTBL+SGCOLUMNTBL+SGIDX+SGRIX+SGTIX
```

Note that the methods for determining the variables *SGROWTBL*, *SGCOLUMNTBL*, *SGIDX*, *SGRIX*, and *SGTIX* differ for single-chunk tables and multi-chunk tables. If you store both single-chunk tables and multi-chunk tables in the same data DB area, you must determine the variables *SGROWTBL*, *SGCOLUMNTBL*, *SGIDX*, *SGRIX*, and *SGTIX* for both types of tables and add them up. The following shows the methods of determining the values of the variables *SGROWTBL*, *SGCOLUMNTBL*, *SGIDX*, *SGRIX*, and *SGTIX*:

■ For a single-chunk table

SGROWTBL **(for single-chunk tables)**

Number of segments for managing the data pages for row store tables in the data DB area

For details, see (a) Determining the variable SGROWTBL (for a single-chunk table).

SGCOLUMNTBL **(for single-chunk tables)**

Number of segments for managing the data pages for column store tables in the data DB area

For details, see (b) Determining the variable SGCOLUMNTBL (for a single-chunk table).

SGIDX **(for single-chunk tables)**

Number of segments for managing the B-tree index pages in the data DB area

For details, see (c) Determining the variable SGIDX (for a single-chunk table).

SGRIX **(for single-chunk tables)**

Number of segments for managing the range index pages in the data DB area

For details, see (d) Determining the variable SGRIX (for a single-chunk table).

SGTIX **(for single-chunk tables)**

Number of segments for managing the text index pages in the data DB area

For details, see (e) Determining the variable SGTIX (for a single-chunk table).

■ For a multi-chunk table

SGROWTBL **(for multi-chunk tables)**

Number of segments for managing the data pages for row store tables in the data DB area

For details, see (f) Determining the variable SGROWTBL (for a multi-chunk table).

SGCOLUMNTBL **(for multi-chunk tables)**

Number of segments for managing the data pages for column store tables in the data DB area

For details, see (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table).

SGIDX **(for multi-chunk tables)**

Number of segments for managing the B-tree index pages in the data DB area

For details, see (h) Determining the variable SGIDX (for a multi-chunk table).

SGRIX **(for multi-chunk tables)**

Number of segments for managing the range index pages in the data DB area

For details, see (i) Determining the variable SGRIX (for a multi-chunk table).

SGTIX **(for multi-chunk tables)**

Number of segments for managing the text index pages in the data DB area

For details, see (j) Determining the variable SGTIX (for a multi-chunk table).

(a) Determining the variable SGROWTBL (for a single-chunk table)

Use the following formula to determine this value.

$$SGROWTBL = \sum_{i=1}^{row\_tbl\_num\_in\_dbarea} \left(\left\uparrow \frac{BP(i)}{SEGSIZE} \right\uparrow + \left\uparrow \frac{VP(i)}{SEGSIZE} \right\uparrow \right)$$

row_tbl_num_in_dbarea

Total number of tables in the data DB area (tables)

BP(i)

Number of pages for base rows required for storing tables

See (1) Determining the number of pages for base rows (variable BP(i)) in 5.8.2 Determining the number of pages for storing each type of row.

VP(i)

Number of pages for branch rows required for storing tables

See (2) Determining the number of pages for branch rows (variable VP(i)) in 5.8.2 Determining the number of pages for storing each type of row.

(b) Determining the variable SGCOLUMNTBL (for a single-chunk table)

Because data is compressed when it is stored in a column store table, the number of segments cannot be calculated from the record length in the same way as for row store tables. Instead, the process involves storing a small amount of data and estimating the number of segments based on the resulting compression rate.

Use the following formula to determine this value.

$$SGCOLUMNTBL = \sum_{i=1}^{column\_tbl\_num\_in\_dbarea} \left(\left\uparrow \frac{COLUMNTBLSIZE_{(i)}}{4} \right\uparrow + \left\uparrow \frac{ADDTBLPGNUM_{(i)} + DELTBLPGNUM_{(i)}}{SEGSIZE - 1} \right\uparrow \right)$$

column_tbl_num_in_dbarea

Total number of column store tables in the data DB area (tables)

SEGSIZE

Segment size (pages)

Use the following formula to determine this value.

```
SEGSIZE = 4,194,304 ÷ page_size
```

COLUMNTBLSIZE(i)

Data size of the i-th column store table (megabytes)

Use the following formula to determine this value.

```
COLUMNTBLSIZE(i) = IMPORTDATASIZE(i) × COMPRESSION_RATE + COLUMNIZESIZE(i)
```

IMPORTDATASIZE(i)

Size of the source data (CSV file) stored in the i-th column store table (megabytes)

COMPRESSION_RATE

Compression rate of source data

Use the `adbimport` command to store part of the source data to be stored in the i-th column store table. Then, use the following formula to determine the compression rate:

$$COMPRESSION\_RATE = \frac{testdbsize}{testcsvsize}$$

testdbsize

The amount of space the partial source data imported by the `adbimport` command occupies in the database (megabytes)

As the partial source data imported to complete the preceding formula, we recommend that you prepare approximately 100 to 500 megabytes of data whose composition is representative of the actual data. When executing the `adbimport` command, specify `2` for the `adb_import_rthd_num` import option.

After executing the `adbimport` command, use the `adbdbstatus` command to output the `Used_segments` (number of segments used by the table) value in the *table summary information* in megabytes. Use this information to check the number of segments used in the database.

For details about the `adbimport` command and the `adbdbstatus` command, see the manual *HADB Command Reference*.

testcsvsize

The size of the CSV file containing the partial source data imported by the `adbimport` command (megabytes)

COLUMNIZESIZE(i)

Size of data that is stored in the i-th column store table in column store format by the maintenance processing of the updated-row columnizing facility (megabytes)

Add the *COLUMNIZESIZE(i)* variable only if the updated-row columnizing facility is enabled. If a B-tree index is defined for the i-th column store table, assume that this value is 0 during estimation.

The formula is as follows:

$$COLUMNIZESIZE(i) = \uparrow IMPORTDATASIZE(i) \times COMPRESSION\_RATE \times \frac{insert\_row\_num(i)}{import\_row\_num(i)} \uparrow$$

IMPORTDATASIZE(i)

Size of the source data (CSV file) stored in the i-th column store table (megabytes)

COMPRESSION_RATE

Compression rate of source data

Determine the value as in the explanation of the *COMPRESSION_RATE* variable shown earlier.

insert_row_num(i)

Number of rows that are added or updated by the INSERT or UPDATE statement in the i-th column store table

import_row_num(i)

Number of rows in the source data (CSV file) stored in the i-th column store table

ADDTBLPGNUM(i)

The number of pages of data to be added to the i-th column store table by using the INSERT or UPDATE statement

Use the following formula to determine this value.

```
ADDTBLPGNUM(i) = BP(i) + VP(i)
```

BP(i)

The number of basic row pages of data to be added to the i-th column store table by using the INSERT or UPDATE statement

Determine this value by referring to (1) Determining the number of pages for base rows (variable BP(i)) in 5.8.2 Determining the number of pages for storing each type of row based on the number of rows to be added to the i-th column store table by using the INSERT or UPDATE statement.

VP(i)

The number of branch row pages of data to be added to the i-th column store table by using the INSERT or UPDATE statement

Determine this value by referring to (2) Determining the number of pages for branch rows (variable VP(i)) in 5.8.2 Determining the number of pages for storing each type of row based on the number of rows to be added to the i-th column store table by using the INSERT or UPDATE statement.

DELTBLPGNUMI(i)

The number of pages of data to be added to the invalid row information pages for the i-th column store table by using the UPDATE or DELETE statement

Use the following formula to determine this value.

```
DELTBLPGNUM(i) = DELLEAFPGNUM(i) + DELUPPERPGNUM(i)
```

DELLEAFPGNUM(i)

Use the following formula to determine this value.

$$DELLEAFPGNUM_{(i)} = \left\uparrow \frac{12 \times (update\_row\_num + delete\_row\_num)}{0.5 \times (page\_size - 80)} \right\uparrow$$

update_row_num

The number of rows to be updated in the i-th column store table by using the UPDATE statement

delete_row_num

The number of rows to be deleted from the i-th column store table by using the DELETE statement

page_size

The page size of the DB area in which the i-th column store table is defined (bytes)

DELUPPERPGNUM(i)

Use the following formula to determine this value.

$$DELUPPERPGNUM_{(i)} = \left\uparrow \frac{12 \times DELLEAFPGNUM_{(i)}}{0.5 \times (page\_size - 80)} \right\uparrow$$

The value of the variable *SGCOLUMNTBL* can also be estimated by assuming that *non-compression (NONE)* is specified as the compression type for all columns and determining the size accordingly. However, depending on the selected compression type and the compression rate that is achieved, the estimated value might be larger than the actual size of the stored data.

Use the following formula to estimate the value based on the assumption that *non-compression (NONE)* is specified as the compression type for all columns:

$$SGCOLUMNTBL = \sum_{i=1}^{column\_tbl\_num\_in\_dbarea} (COLUMNDATASEGNUM(i) + ROWDATASEGNUM(i) + COLUMNIZESEGNUM(i))$$

COLUMNDATASEGNUM(i)

Number of column-data segments for i-th column store table

Use the following formula to determine this value.

$$COLUMNDATASEGNUM_{(i)} = \frac{row\_num}{\mathrm{MIN}\left\{262{,}144 \ , \left(\left\downarrow \frac{page\_size - 80}{COL\_MAX\_SIZE \times \uparrow column\_num \div COL\_PAGE\_NUM \uparrow} \right\downarrow\right) \times \mathrm{MAX}\left(1, \left\downarrow \frac{COL\_PAGE\_NUM}{column\_num} \right\downarrow\right)\right\}}$$

row_num

Number of rows stored in i-th column store table

column_num

Number of columns in i-th column store table

page_size

Page size of data DB area (bytes)

COL_PAGE_NUM

The value to substitute for the *COL_PAGE_NUM* variable differs depending on the page size of the data DB area. Substitute the value in the following table that corresponds to the page size of the data DB area:

Table 5-6: Value to substitute for COL_PAGE_NUM variable

No.	Page size of data DB area (kilobytes)	Value of *COL_PAGE_NUM*
1	4	1,021
2	8	510
3	16	255
4	32	127

COL_MAX_SIZE

Maximum data length of column data in i-th column store table

Use the following formula to determine this value.

$$COL\_MAX\_SIZE = \max_{k=1,\ldots,column\_num} COLUMNDATASIZE(k)$$

COLUMNDATASIZE(k)

Data length of column data in k-th column (bytes)

The value to substitute for the *COLUMNDATASIZE* variable differs depending on the data type of the column data. See the following table, and substitute the applicable value.

Table 5-7: Data length of each type of column data

No.	Classification	Data type		Data length (bytes)
1	Numeric data	`INTEGER`		10
2		`SMALLINT`		6
3		`DECIMAL`(m, n)#	$m \leq 4$	4
4			$5 \leq m \leq 8$	6
5			$9 \leq m \leq 16$	10
6			$17 \leq m$	18
7		`DOUBLE PRECISION`		10
8	Character string data	`CHARACTER`(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
9			$128 \leq n$	10
10		`VARCHAR`(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
11			$128 \leq n \leq 255$	128
12			$256 \leq n$	130
13	Datetime data	`DATE`		6
14		`TIME`(p)		$\left\uparrow \frac{3 + \uparrow p \div 2 \uparrow}{2} \right\uparrow \times 2$
15		`TIMESTAMP`(p)		$\left\uparrow \frac{7 + \uparrow p \div 2 \uparrow}{2} \right\uparrow \times 2$
16	Binary data	`BINARY`(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
17			$128 \leq n$	10
18		`VARBINARY`(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
19			$128 \leq n \leq 255$	128
20			$256 \leq n$	130

Legend:

m, n: Positive integers

p: 0, 3, 6, 9 or 12

#

Indicates a fixed-point number that has a total of m digits, with n digits following the decimal point. If m is omitted, `38` is assumed.

ROWDATASEGNUM(i)

Number of row-data segments for i-th column store table

Use the following formula to determine this value.

$$ROWDATASEGNUM_{(i)} = \left\{ MAX\left(\left\lceil \frac{row\_num \times var\_num}{255} \right\rceil , \left\lceil \frac{row\_num \times (14 + var\_size) \times var\_num}{page\_size - 80} \right\rceil \right) + ADDTBLPGNUM_{(i)} + DELTBLPGNUM_{(i)} \right\} \div SEGSIZE$$

row_num

Number of rows stored in i-th column store table

page_size

Page size of data DB area (bytes)

var_num

Number of columns managed as branch rows (columns)

When the data type is `CHAR` or `BINARY`, determine the number of columns whose definition length is 128 bytes or longer.

When the data type is `VARCHAR` or `VARBINARY`, determine the number of columns that include data whose actual length is 128 bytes or longer.

var_size

Data length of columns in the branch rows (bytes)

The value to substitute for the *var_size* variable differs depending on the data type. See the following table, and substitute the applicable value.

Table 5-8: Value to be substituted for the variable var_size

No.	Classification	Data type	Data length (bytes)
1	Character string data	`CHAR`	Definition length
2		`VARCHAR`	*d*
3	Binary data	`BINARY`	Definition length
4		`VARBINARY`	*d*

Legend:

d: Actual data length

COLUMNIZESEGNUM(i)

Number of column-data segments that are stored in the i-th column store table by the maintenance processing of the updated-row columnizing facility

Add the *COLUMNIZESEGNUM(i)* variable only if the updated-row columnizing facility is enabled. If a B-tree index is defined for the i-th column store table, assume that this value is 0 during estimation.

The formula is as follows:

```
COLUMNIZESEGNUM(i) = COLUMNDATASEGNUM(i)
```

For details about how to determine the value of the *COLUMNDATASEGNUM(i)* variable, see the explanation of the *COLUMNDATASEGNUM(i)* variable shown earlier. When doing so, for the *row_num* variable, substitute the number of rows added or updated by the `INSERT` or `UPDATE` statement in the i-th column store table.

(c) Determining the variable SGIDX (for a single-chunk table)

Use the following formula to determine this value.

$$SGIDX = \sum_{i=1}^{idx\_num\_in\_dbarea} \left(\left\uparrow \frac{IP\_LOWER_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{IP\_UPPER_{(i)}}{SEGSIZE} \right\uparrow \right)$$

idx_num_in_dbarea

Total number of B-tree indexes in the data DB area (indexes)

IP_LOWER(i)

Number of pages used in the lower page segment of each B-tree index

See (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) in 5.8.3 Determining the number of storage pages for each B-tree index segment.

IP_UPPER(i)

Number of pages used in the upper page segment of each B-tree index

See (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) in 5.8.3 Determining the number of storage pages for each B-tree index segment.

(d) Determining the variable SGRIX (for a single-chunk table)

Use the following formula to determine this value.

$$SGRIX = \sum_{i=1}^{rng\_num\_in\_dbarea} RS_{(i)}$$

rng_num_in_dbarea

Number of range indexes in the data DB area (indexes)

RS(i)

Number of segments required for storing each range index

For details, see 5.8.6 Determining the number of segments for storing each range index.

(e) Determining the variable SGTIX (for a single-chunk table)

Use the following formula to determine its value.

$$SGTIX = \sum_{i=1}^{idx\_num\_in\_dbarea} \left(\left\uparrow \frac{TIP\_STRSEG_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{TIP\_APPSEG_{(i)}}{SEGSIZE} \right\uparrow \right)$$

idx_num_in_dbarea

Total number of text indexes in the data DB area (indexes)

TIP_STRSEG(i)

Number of pages used in the string control segment of each text index

See (1) Determining the number of storage pages used in the string control segment (variable TIP_STRSEG(i)) in 5.8.5 Determining the number of storage pages for each text index segment.

TIP_APPSEG(i)

Number of pages used in the position control segment of each text index

See (2) Determining the number of storage pages used in the position control segment (variable TIP_APPSEG(i)) in 5.8.5 Determining the number of storage pages for each text index segment.

(f) Determining the variable SGROWTBL (for a multi-chunk table)

Use the following formula to determine this value.

$$SGROWTBL = \sum_{i=1}^{row\_tbl\_num\_in\_dbarea} \left\{ \sum_{k=1}^{chunk\_num} \left(\left\uparrow \frac{CHBP_{(i,k)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{CHVP_{(i,k)}}{SEGSIZE} \right\uparrow \right) + SGLTBL_{(i)} + SGLIDX_{(i)} \right\}$$

row_tbl_num_in_dbarea

Total number of row store tables in the data DB area (tables)

chunk_num

Number of non-archived chunks to be created (chunks)

CHBP(i,k)

Number of pages for base rows required for storing tables of each chunk

Substitute the value estimated from the amount of data to be stored by the `adbimport` command with the `-b` option specified for the target chunk, the `INSERT` statement, and the `UPDATE` statement.

CHVP(i,k)

Number of pages for branch rows required for storing tables of each chunk

Substitute the value estimated from the amount of data to be stored by the `adbimport` command with the `-b` option specified for the target chunk, the `INSERT` statement, and the `UPDATE` statement.

SGLTBL(i)

Number of segments required to store the location table for the i-th archivable multi-chunk table

You must determine this variable for only archivable multi-chunk tables.

Use the following formula to determine its value.

Formula

$$SGLTBL_{(i)} = \left\uparrow \frac{LBP_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{LVP_{(i)}}{SEGSIZE} \right\uparrow$$

SEGSIZE

See *SEGSIZE* in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area.

LBP(i)

Use the following formula to determine its value.

Formula

$$LBP_{(i)} = \left\lceil \frac{LROWNUM}{\mathrm{MIN}\left(255 , \left\lfloor \frac{\left\lfloor \frac{page\_size \times 7}{10} \right\rfloor - 56}{row\_size} \right\rfloor \right)} \right\rceil$$

page_size

Page size of the data DB area in which archivable multi-chunk tables are defined (bytes)

row_size

Row size (bytes)

The value to be substituted for the variable *row_size* differs depending on the data type of the archive range column. The following table shows the relationship between the variable *row_size* and the data type of the archive range column.

Table 5-9: Relationship between the variable row_size and the data type of the archive range column

No.	Data type of the archive range column		Value to be substituted for the variable row_size
1	INTEGER		102
2	SMALLINT		94
3	DECIMAL(*m*,*n*)	$1 \leq m \leq 4$	90
4		$5 \leq m \leq 8$	94
5		$9 \leq m \leq 16$	102
6		$17 \leq m \leq 38$	118
7	DOUBLE PRECISION		102
8	CHARCTER(*p*)		88 + 2 x *p*
9	DATE		94
10	TIME		104
11	TIMESTAMP		112

Legend:

m: Precision

n: Scaling

p: String length (bytes)

LROWNUM

Number of rows to be stored in the location table (rows)

Use the following formula to determine this value.

Formula

$$LROWNUM = \sum_{i=1}^{archive\_chunk\_num} FILENUM_{(i)} + \sum_{j=1}^{merge\_chunk\_num} FILENUM_{(j)}$$

archive_chunk_num

Number of executed `adbarchivechunk` commands

merge_chunk_num

Number of executed `adbmergechunk` commands

FILENUM(*i*)

Total number of archive files created by the i-th `adbarchivechunk` command

Use the following formula to determine this value.

Formula

$$FILENUM_{(i)} = \sum_{i=1}^{chunk\_num} (SGMTGRPNUM \times thd\_num)$$

FILENUM(j)

Total number of archive files corresponding to the archived chunks to be merged by the j-th `adbmergechunk` command

Determine this value by using the same formula that is used for determining the variable *FILENUM(i)*. During determination, replace *i* by *j*.

chunk_num

Number of chunks to be processed by the command (chunks)

thd_num

Substitute the following value.

- To determine the number of archive files created by the `adbarchivechunk` command

  Use the following formula to determine this value.

  ```
  ↓(value-specified-for-archive-chunk-option-adb_arcv_rthd_num-when-adbarchivechunk-command-is-executed - 1) ÷ 2↓
  ```

- To determine the number of archive files for the archived chunks to be merged by the `adbmergechunk` command

  Specify the following value:

  ```
  value-specified-for-archive-chunk-option-adb_arcv_rthd_num-when-adbarchivechunk-command-is-executed-for-chunks-processed-by-adbmergechunk-command
  ```

SGMTGRPNUM

The value of the variable *SGMTGRPNUM* changes according to the value of the variable *sgmtnum_in_chunk*. The following table shows the relationship between the variables *sgmtnum_in_chunk* and *SGMTGRPNUM*.

Table 5-10: Relationship between the variables sgmtnum_in_chunk and SGMTGRPNUM

No.	Value of the variable *sgmtnum_in_chunk*	Value of the variable *SGMTGRPNUM*
1	*sgmtnum_in_chunk* < 16	For *SGMTGRPNUM*, substitute 1.
2	16 ≤ *sgmtnum_in_chunk* and *sgmtnum_in_chunk* < 32	For *SGMTGRPNUM*, substitute 4.
3	32 ≤ *sgmtnum_in_chunk* and *sgmtnum_in_chunk* < 64	For *SGMTGRPNUM*, substitute 8.
4	64 ≤ *sgmtnum_in_chunk* and *sgmtnum_in_chunk* < 256	For *SGMTGRPNUM*, substitute 16.

No.	Value of the variable *sgmtnum_in_chunk*	Value of the variable *SGMTGRPNUM*
5	256 ≤ *sgmtnum_in_chunk*	Use the following formula to determine the value of *SGMTGRPNUM*: *SGMTGRPNUM*= MAX(↓ *sgmtnum_in_chunk*÷256 ↓ , 32)

sgmtnum_in_chunk

Specify the number of segments in the largest table among the chunks to be archived.

Use the following formula to determine this value.

Formula

$$sgmtnum\_in\_chunk = \left\uparrow \frac{CHBP(j)}{SEGSIZE} \right\uparrow + \left\uparrow \frac{CHVP(j)}{SEGSIZE} \right\uparrow$$

CHBP(j)

For details, see *CHBP(i,k)*.

CHVP(j)

For details, see *CHVP(i,k)*.

SEGSIZE

See *SEGSIZE* in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area.

page_size

See *page_size* in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area.

LVP(i)

Use the following formula to determine this value.

Formula

$$LVP_{(i)} = \mathrm{MAX}(A\ ,\ B)$$

Explanation of variables :

A : Use the following formula to determine its value.

$$\left\uparrow \frac{LROWNUM}{255} \right\uparrow$$

B : Use the following formula to determine its value.

$$\left\uparrow \frac{LROWNUM \times 524}{\mathrm{MAX}\left(\left\downarrow \frac{page\_size \times 7}{10} \right\downarrow - 56\ ,\ \mathrm{MIN}(524\ ,\ page\_size - 56) \right)} \right\uparrow$$

SGLIDX(i)

Number of segments required to store the indexes defined for the location table corresponding to the i-th archivable multi-chunk table

You must determine this variable for only archivable multi-chunk tables.

Use the following formula to determine this value.

Formula

$$SGLIDX_{(i)} = \left\uparrow \frac{LICHKIDP\_LOWER_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{LICHKIDP\_UPPER_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{LIRNG01P\_LOWER_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{LIRNG01P\_UPPER_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{LIRNG02P\_LOWER_{(i)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{LIRNG02P\_UPPER_{(i)}}{SEGSIZE} \right\uparrow$$

LICHKIDP_LOWER(i)

Number of pages that are used in the lower page segments of the indexes defined for the chunk ID of the chunk for the archive file (pages)

LICHKIDP_UPPER(i)

Number of pages that are used in the upper page segments of the indexes defined for the chunk ID of the chunk for the archive file (pages)

LIRNG01P_LOWER(i)

Number of pages that are used in the lower page segment of the indexes defined for the value range of the archive range column (upper and lower limit values) for the data stored in the archive file (pages)

LIRNG01P_UPPER(i)

Number of pages that are used in the upper page segment of the indexes defined for the value range of the archive range column (upper and lower limit values) for the data stored in the archive file (pages)

LIRNG02P_LOWER(i)

Number of pages that are used in the lower page segment of the indexes defined for the lower limit value of the archive range column for the data stored in the archive file (pages)

LIRNG02P_UPPER(i)

Number of pages that are used in the upper page segment of the indexes defined for the lower limit value of the archive range column for the data stored in the archive file (pages)

For the variables *LICHKIDP_LOWER(i)*, *LIRNG01P_LOWER(i)*, and *LIRNG02P_LOWER(i)*, determine the values by using the formula for determining the variable *IP_LOWER(i)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment. For the variables *LICHKIDP_UPPER(i)*, *LIRNG01P_UPPER(i)*, and *LIRNG02P_UPPER(i)*, determine the values by using the formula for determining the variable *IP_UPPER(i)* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment. At this time, substitute values for variables as follows:

R

Substitute 1.5.

key_num

Substitute 0.

dup_key_num

Substitute the value determined from the following formula.

Formula

$$dup\_key\_num = \uparrow FILENUM_{(i)} \div 256 \uparrow$$

page_size

Substitute in bytes the page size of the data DB area that stores the archivable multi-chunk table.

pctfree

Substitute 30.

key_dup

Substitute 0.

KEYSZDB

See the following table, and substitute the applicable value.

Table 5-11: Value to be substituted for the variable KEYSZDB

No.	Variable to be estimated	Data type of the archive range column		Value to be substituted for the variable KEYSZDB
1	The variables to be estimated are as follows: • *LICHKIDP_LOWER(i)* • *LICHKIDP_UPPER(i)*	--		8
2	The variables to be estimated are as follows: • *LIRNG01P_LOWER(i)* • *LIRNG01P_UPPER(i)*	INTEGER		16
3		SMALLINT		8
4		DECIMAL(*m*,*n*)	$1 \leq m \leq 4$	4
5			$5 \leq m \leq 8$	8
6			$9 \leq m \leq 16$	16
7			$17 \leq m \leq 38$	32
8		DOUBLE PRECISION		16
9		CHARACTER(*p*)		2 x *p*
10		DATE		8
11		TIME		18
12		TIMESTAMP		26
13	The variables to be estimated are as follows: • *LIRNG02P_LOWER(i)* • *LIRNG02P_UPPER(i)*	INTEGER		8
14		SMALLINT		4
15		DECIMAL(*m*,*n*)	$1 \leq m \leq 4$	2
16			$5 \leq m \leq 8$	4
17			$9 \leq m \leq 16$	8
18			$17 \leq m \leq 38$	16
19		DOUBLE PRECISION		8
20		CHARACTER(*p*)		*p*
21		DATE		4
22		TIME		9
23		TIMESTAMP		13

Legend:

--: Not applicable.

m: Precision

n: Scaling

p: String length (bytes)

(g) Determining the variable SGCOLUMNTBL (for a multi-chunk table)

Because data is compressed when it is stored in a column store table, the number of segments cannot be calculated from the record length in the same way as for row store tables. Instead, the process involves storing a small amount of data and estimating the number of segments based on the resulting compression rate.

Use the following formula to determine this value.

$$SGCOLUMNTBL = \sum_{i=1}^{column\_tbl\_num\_in\_dbarea} \left(\left\uparrow \frac{COLUMNTBLSIZE_{(i)}}{4} \right\uparrow + UPDATESEGNUM_{(i)} \right)$$

column_tbl_num_in_dbarea

Total number of column store tables in the data DB area (tables)

COLUMNTBLSIZE(i)

Data size of the i-th column store table (megabytes)

Use the following formula to determine this value.

$$COLUMNTBLSIZE(i) = \sum_{k=1}^{chunk\_num} (IMPORTDATASIZE(i,k) \times COMPRESSION\_RATE + COLUMNIZESIZE(i,k))$$

chunk_num

Number of chunks to be created (chunks)

IMPORTDATASIZE(i,k)

Size of the source data (CSV file) stored in the k-th chunk of the i-th column store table (megabytes)

COMPRESSION_RATE

Compression rate of source data

Use the `adbimport` command to store part of the source data to be stored in the i-th column store table. Then, use the following formula to determine the compression rate:

$$COMPRESSION\_RATE = \frac{testdbsize}{testcsvsize}$$

testdbsize

The amount of space the partial source data imported by the `adbimport` command occupies in the database (megabytes)

As the partial source data imported to complete the preceding formula, we recommend that you prepare approximately 100 to 500 megabytes of data whose composition is representative of the actual data. When executing the `adbimport` command, specify `2` for the `adb_import_rthd_num` import option.

After executing the `adbimport` command, use the `adbdbstatus` command to output the `Used_segments` (number of segments used by the table) value in the *table summary information* in megabytes. Use this information to check the number of segments used in the database.

For details about the `adbimport` command and the `adbdbstatus` command, see the manual *HADB Command Reference*.

testcsvsize

The size of the CSV file containing the partial source data imported by the `adbimport` command (megabytes)

COLUMNIZESIZE(i,k)

Size of data that is stored in the k-th chunk of the i-th column store table in column store format by the maintenance processing of the updated-row columnizing facility (megabytes)

Add *COLUMNIZESIZE(i,k)* only if the updated-row columnizing facility is enabled. If a B-tree index is defined for the i-th column store table, assume that this value is 0 during estimation.

The formula is as follows:

$$COLUMNIZESIZE(i,k) = \left\uparrow IMPORTDATASIZE(i,k) \times COMPRESSION\_RATE \times \frac{insert\_row\_num(i,k)}{import\_row\_num(i,k)} \right\uparrow$$

IMPORTDATASIZE(i,k)

Size of the source data (CSV file) stored in the k-th chunk of the i-th column store table (megabytes)

COMPRESSION_RATE

Compression rate of source data

Determine the value as in the explanation of the *COMPRESSION_RATE* variable shown earlier.

insert_row_num(i,k)

Number of rows that are added or updated by the `INSERT` or `UPDATE` statement in the k-th chunk of the i-th column store table

import_row_num(i,k)

Number of rows in the source data (CSV file) stored in the k-th chunk of the i-th column store table

UPDATESEGNUM(i)

The number of segments that will be added when an update SQL statement is executed for the i-th column store table

Use the following formula to determine this value.

$$UPDATESEGNUM_{(i)} = \sum_{k=1}^{chunk\_num} \left\uparrow \frac{ADDTBLPGNUM_{(i,k)} + DELTBLPGNUM_{(i,k)}}{SEGSIZE - 1} \right\uparrow$$

SEGSIZE

Segment size (pages)

Use the following formula to determine this value.

```
SEGSIZE = 4,194,304 ÷ page_size
```

page_size

The page size of the DB area in which the i-th column store table is defined (bytes)

ADDTBLPGNUM(i,k)

The number of pages of data to be added to the k-th chunk in the i-th column store table by using the `INSERT` or `UPDATE` statement

Use the following formula to determine this value.

```
ADDTBLPGNUM(i,k) = BP(i,k) + VP(i,k)
```

BP(i,k)

The number of basic row pages of data to be added to the k-th chunk in the i-th column store table by using the `INSERT` or `UPDATE` statement

Determine this value by referring to (1) Determining the number of pages for base rows (variable BP(i)) in 5.8.2 Determining the number of pages for storing each type of row based on the number of rows to be added to the k-th chunk in the i-th column store table by using the `INSERT` or `UPDATE` statement.

VP(i,k)

The number of branch row pages of data to be added to the k-th chunk in the i-th column store table by using the `INSERT` or `UPDATE` statement

Determine this value by referring to (2) Determining the number of pages for branch rows (variable VP(i)) in 5.8.2 Determining the number of pages for storing each type of row based on the number of rows to be added to the k-th chunk in the i-th column store table by using the `INSERT` or `UPDATE` statement.

DELTBLPGNUM(i,k)

The number of pages of data to be added to the invalid row information pages for the k-th chunk in the i-th column store table by using the `UPDATE` or `DELETE` statement

Use the following formula to determine this value.

```
DELTBLPGNUM(i,k) = DELLEAFPGNUM(i,k) + DELUPPERPGNUM(i,k)
```

DELLEAFPGNUM(i,k)

Use the following formula to determine this value.

$$DELLEAFPGNUM_{(i,k)} = \left\uparrow \frac{12 \times (update\_row\_num + delete\_row\_num)}{0.5 \times (page\_size - 80)} \right\uparrow$$

update_row_num

The number of rows to be updated for the k-th chunk in the i-th column store table by using the `UPDATE` statement

delete_row_num

The number of rows to be deleted from the k-th chunk in the i-th column store table by using the `DELETE` statement

page_size

The page size of the DB area in which the i-th column store table is defined (bytes)

DELUPPERPGNUM(i,k)

Use the following formula to determine this value.

$$DELUPPERPGNUM_{(i,k)} = \left\uparrow \frac{12 \times DELLEAFPGNUM_{(i,k)}}{0.5 \times (page\_size - 80)} \right\uparrow$$

The value of the variable *SGCOLUMNTBL* can also be estimated by assuming that *non-compression (`NONE`)* is specified as the compression type for all columns and determining the size accordingly. However, depending on the selected compression type and the compression rate that is achieved, the estimated value might be larger than the actual size of the stored data.

Use the following formula to estimate the value based on the assumption that *non-compression (`NONE`)* is specified as the compression type for all columns:

$$SGCOLUMNTBL = \sum_{i=1}^{column\_tbl\_num\_in\_dbarea} COLUMNTBLSEGNUM_{(i)}$$

COLUMNTBLSEGNUM(i)

Number of segments for i-th column store table

Use the following formula to determine this value.

$$COLUMNTBLSEGNUM(i) = \sum_{k=1}^{chunk\_num} (COLUMNDATASEGNUM(i,k) + ROWDATASEGNUM(i,k) + COLUMNIZESEGNUM(i,k))$$

COLUMNDATASEGNUM(i,k)

Number of column-data segments for k-th chunk of i-th column store table

Use the following formula to determine this value.

$$COLUMNDATASEGNUM(i,k) = \frac{row\_num}{\mathrm{MIN}\left\{262{,}144\ ,\left(\left\downarrow \frac{page\_size - 80}{COL\_MAX\_SIZE \times \uparrow column\_num \div COL\_PAGE\_NUM \uparrow} \right\downarrow\right) \times \mathrm{MAX}\left(1, \left\downarrow \frac{COL\_PAGE\_NUM}{column\_num} \right\downarrow\right)\right\}}$$

row_num

Number of rows stored in k-th chunk of i-th column store table

column_num

Number of columns in i-th column store table

page_size

Page size of data DB area (bytes)

COL_PAGE_NUM

The value to substitute for the *COL_PAGE_NUM* variable differs depending on the page size of the data DB area. Substitute the value in the following table that corresponds to the page size of the data DB area:

Table 5-12: Value to substitute for COL_PAGE_NUM variable

No.	Page size of data DB area (kilobytes)	Value of *COL_PAGE_NUM*
1	4	1,021
2	8	510
3	16	255
4	32	127

COL_MAX_SIZE

Maximum data length of column data in i-th column store table

Use the following formula to determine this value.

$$COL\_MAX\_SIZE = \max_{s=1,\ldots,column\_num} COLUMNDATASIZE(s)$$

COLUMNDATASIZE(s)

Data length of column data in s-th column

The value to substitute for the *COLUMNDATASIZE* variable differs depending on the data type of the column data. See the following table, and substitute the applicable value.

Table 5-13: Data length of each type of column data

No.	Classification	Data type		Data length (bytes)
1	Numeric data	INTEGER		10
2		SMALLINT		6
3		DECIMAL(m, n)#	$m \leq 4$	4
4			$5 \leq m \leq 8$	6
5			$9 \leq m \leq 16$	10
6			$17 \leq m$	18
7		DOUBLE PRECISION		10
8	Character string data	CHARACTER(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
9			$128 \leq n$	10
10		VARCHAR(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
11			$128 \leq n \leq 255$	128
12			$256 \leq n$	130
13	Datetime data	DATE		6
14		TIME(p)		$\left\uparrow \frac{3 + \uparrow p \div 2 \uparrow}{2} \right\uparrow \times 2$
15		TIMESTAMP(p)		$\left\uparrow \frac{7 + \uparrow p \div 2 \uparrow}{2} \right\uparrow \times 2$
16	Binary data	BINARY(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
17			$128 \leq n$	10
18		VARBINARY(n)	$n \leq 127$	$\left\uparrow \frac{n+1}{2} \right\uparrow \times 2$
19			$128 \leq n \leq 255$	128
20			$256 \leq n$	130

Legend:

m, n: Positive integers

p: 0, 3, 6, 9 or 12

#

Indicates a fixed-point number that has a total of m digits, with n digits following the decimal point. If m is omitted, 38 is assumed.

ROWDATASEGNUM(i,k)

Number of row-data segments for k-th chunk of i-th column store table

Use the following formula to determine this value.

$$ROWDATASEGNUM_{(i,k)} = \left\{ MAX \left(\left\lceil \frac{row\_num \times var\_num}{255} \right\rceil , \left\lceil \frac{row\_num \times (14 + var\_size) \times var\_num}{page\_size - 80} \right\rceil \right) + ADDTBLPGNUM_{(i,k)} + DELTBLPGNUM_{(i,k)} \right\} \div SEGSIZE$$

row_num

Number of rows stored in k-th chunk of i-th column store table

page_size

Page size of data DB area (bytes)

var_num

Number of columns managed as branch rows (columns)

When the data type is `CHAR` or `BINARY`, determine the number of columns whose definition length is 128 bytes or longer.

When the data type is `VARCHAR` or `VARBINARY`, determine the number of columns that include data whose actual length is 128 bytes or longer.

var_size

Data length of columns in the branch rows (bytes)

The value to substitute for the *var_size* variable differs depending on the data type. See the following table, and substitute the applicable value.

Table 5-14: Value to be substituted for the variable var_size

No.	Classification	Data type	Data length (bytes)
1	Character string data	`CHAR`	Definition length
2		`VARCHAR`	*d*
3	Binary data	`BINARY`	Definition length
4		`VARBINARY`	*d*

Legend:

d: Actual data length

COLUMNIZESEGNUM(i,k)

Number of column-data segments that are stored in the k-th chunk of the i-th column store table by the maintenance processing of the updated-row columnizing facility

Add *COLUMNIZESEGNUM(i,k)* only if the updated-row columnizing facility is enabled. If a B-tree index is defined for the i-th column store table, assume that this value is 0 during estimation.

The formula is as follows:

```
COLUMNIZESEGNUM(i,k) = COLUMNDATASEGNUM(i,k)
```

For details about how to determine the value of *COLUMNDATASEGNUM(i,k)*, see the explanation of the *COLUMNDATASEGNUM(i,k)* variable shown earlier. When doing so, for the *row_num* variable, assign the number of rows added or updated by the `INSERT` or `UPDATE` statement in the k-th chunk of the i-th column store table.

(h) Determining the variable SGIDX (for a multi-chunk table)

Use the following formula to determine this value.

$$SGIDX = \sum_{i=1}^{idx\_num\_in\_dbarea} \left\{ \sum_{k=1}^{chunk\_num} \left(\left\uparrow \frac{CHIP\_LOWER_{(i,k)}}{SEGSIZE} \right\uparrow + \left\uparrow \frac{CHIP\_UPPER_{(i,k)}}{SEGSIZE} \right\uparrow \right) \right\}$$

idx_num_in_dbarea

Total number of B-tree indexes in the data DB area (indexes)

chunk_num

Number of non-archived chunks to be created (chunks)

CHIP_LOWER(i,k)

Number of pages used in the lower page segment of the B-tree index for each chunk

See (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) in 5.8.3 Determining the number of storage pages for each B-tree index segment.

At this time, substitute the value estimated from the amount of data to be stored by the `adbimport` command with the `-b` option specified for the target chunk, the `INSERT` statement, and the `UPDATE` statement.

CHIP_UPPER(i,k)

Number of pages used in the upper page segment of the B-tree index for each chunk

See (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) in 5.8.3 Determining the number of storage pages for each B-tree index segment.

At this time, substitute the value estimated from the amount of data to be stored by the `adbimport` command with the `-b` option specified for the target chunk, the `INSERT` statement, and the `UPDATE` statement.

(i) Determining the variable SGRIX (for a multi-chunk table)

Use the following formula to determine this value.

$$SGRIX = \sum_{i=1}^{rng\_num\_in\_dbarea} \left\{ \sum_{k=1}^{chunk\_num} CHRS_{(i,k)} \right\}$$

rng_num_in_dbarea

Number of range indexes in the data DB area (indexes)

chunk_num

Number of non-archived chunks to be created (chunks)

CHRS(i,k)

Number of segments required for storing the range indexes of each chunk

Substitute the value estimated from the amount of data to be stored by the `adbimport` command with the `-b` option specified for the target chunk, the `INSERT` statement, and the `UPDATE` statement.

(j) Determining the variable SGTIX (for a multi-chunk table)

Use the following formula to determine this value.

$$SGTIX = \sum_{i=1}^{idx\_num\_in\_dbarea} \left\{ \sum_{k=1}^{chunk\_num} \left(\left\lceil \frac{CHTIP\_STRSEG_{(i,k)}}{SEGSIZE} \right\rceil + \left\lceil \frac{CHTIP\_APPSEG_{(i,k)}}{SEGSIZE} \right\rceil \right) \right\}$$

idx_num_in_dbarea

Total number of text indexes in the data DB area (indexes)

chunk_num

Number of non-archived chunks to be created (chunks)

CHTIP_STRSEG(i,k)

Number of pages used in the string control segment of the text index for each chunk

See (1) Determining the number of storage pages used in the string control segment (variable TIP_STRSEG(i)) in 5.8.5 Determining the number of storage pages for each text index segment.

At this time, substitute the value estimated from the amount of data to be stored by the `adbimport` command with the `-b` option specified for the target chunk, the `INSERT` statement, and the `UPDATE` statement.

CHTIP_APPSEG(i,k)

Number of pages used in the position control segment of the text index for each chunk

See (2) Determining the number of storage pages used in the position control segment (variable TIP_APPSEG(i)) in 5.8.5 Determining the number of storage pages for each text index segment.

At this time, substitute the value estimated from the amount of data to be stored by the `adbimport` command with the `-b` option specified for the target chunk, the `INSERT` statement, and the `UPDATE` statement.

5.8.2 Determining the number of pages for storing each type of row

This subsection explains how to determine the number of pages for tables that will store base rows (variable *BP(i)*) and the number of pages for tables that will store branch rows (variable *VP(i)*).

(1) Determining the number of pages for base rows (variable BP(i))

Use the formula shown below to determine the number of pages for base rows (variable *BP(i)*). The part inside the parentheses in the denominator of variable *BP(i)* indicates the number of rows stored per page, and is a value in the range from 1 to 255.

Formula

$$BP_{(i)} = \left\lceil \frac{row\_num}{\text{MIN}\left(255, \left\lfloor \frac{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor - CTRLSIZE}{ROWSZ} \right\rfloor \right)} \right\rceil$$

Explanation of variables

- *row_num*
  Number of rows to be stored in the table (rows)
- *page_size*
  Page size in the data DB area (bytes)

- *pctfree*

  Percentage of unused area specified in the `PCTFREE` operand of the `CREATE TABLE` statement (%)

  If the percentage of unused area is not specified, assume 30% for the calculation.

  To obtain the number of basic row pages of data to be added to a column store table by using the `INSERT` or `UPDATE` statement, assume 0% for the calculation.

  Determine the value of *pctfree* such that it satisfies the following formula.

  Formula (pctfree value)

$$\left\downarrow \frac{page\_size \times (100 - pctfree)}{100} \right\downarrow - 56 \geq ROWSZ$$

- *CTRLSIZE*

  If the target table is a row store table, specify `56`.

  To obtain the number of basic row pages of data to be added to a column store table by using the `INSERT` or `UPDATE` statement, specify `80`.

Use the formulas shown below to determine the row length (*ROWSZ*). The formula depends on whether the table is a non-FIX table for which the `BRANCH ALL` table option is not specified, is a non-FIX tables for which the `BRANCH ALL` table option is specified, or is a FIX table.

To obtain the number of basic row pages of data to be added to a column store table by using the `INSERT` or `UPDATE` statement, use the formula for a non-FIX table for which the BRANCH ALL table option is not specified.

Formula for a non-FIX table for which the BRANCH ALL table option is not specified

$$ROWSZ = \left\uparrow \frac{28 + 2 \times col\_num + \sum_{i=1}^{col\_num} col\_size_{(i)}}{2} \right\uparrow \times 2$$

Formula for a non-FIX table for which the BRANCH ALL table option is specified

$$ROWSZ = \left\uparrow \frac{28 + \left\downarrow \frac{col\_num + 7}{8} \right\downarrow + \sum_{i=1}^{col\_num} col\_size_{(i)}}{2} \right\uparrow \times 2$$

Formula for a FIX table

$$ROWSZ = \left\uparrow \frac{26 + \sum_{i=1}^{col\_num} col\_size_{(i)}}{2} \right\uparrow \times 2$$

Explanation of variables

- *col_num*

  Total number of columns in the table (columns)

- *col_size(i)*

  Data length in each column (bytes)

  Determine the data length for each column based on the following table. Then, calculate the total for all columns.

Table 5-15: Data length of each data type

No.	Classification	Data type		Data length (bytes)
1	Numeric data	INTEGER		8
2		SMALLINT		4
3		DECIMAL(*m*, *n*)#	$1 \leq m \leq 4$	2
4			$5 \leq m \leq 8$	4
5			$9 \leq m \leq 16$	8
6			$17 \leq m \leq 38$	16
7		DOUBLE PRECISION		8
8	Character string data	CHARACTER(*n*)		*n*
9		VARCHAR(*n*)		*var_col_size*
10	Datetime data	DATE		4
11		TIME(*p*)		$3 + \uparrow p \div 2 \uparrow$
12		TIMESTAMP(*p*)		$7 + \uparrow p \div 2 \uparrow$
13	Binary data	BINARY(*n*)		*n*
14		VARBINARY(*n*)		*var_col_size*

Legend:

m, *n*: Positive integers

p: 0, 3, 6, 9 or 12

#

Indicates a fixed-point number that has a total of *m* digits, with *n* digits following the decimal point. If *m* is omitted, 38 is assumed.

var_col_size: Data length of VARCHAR and VARBINARY columns

Determine the data length based on the following table.

Table 5-16: Data length of VARCHAR and VARBINARY columns

No.	Table option Whether BRANCH ALL is specified	Column definition BRANCH specification	Definition length n (in bytes)	Data length (in bytes)
1	Yes	--	--	9
2	No	YES	--	11
3		NO	$1 \leq n \leq 255$	$d + 2$
4			$256 \leq n \leq 32{,}000$	$d + 3$
5		AUTO or not specified	$1 \leq n \leq 255$	$d + 2$
6			$256 \leq n \leq 32{,}000$	11

Legend:

n: Positive integer

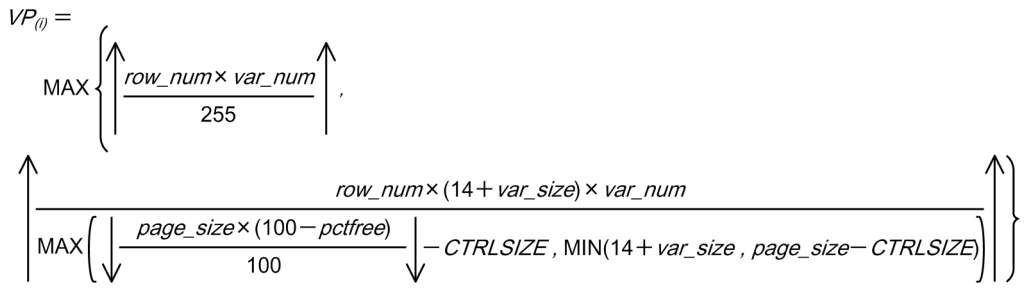
d: Actual data length

--: Not applicable

(2) Determining the number of pages for branch rows (variable VP(i))

Use the following formula to determine the number of pages for branch rows (variable *VP(i)*).

Formula



Explanation of variables

- *row_num*
  Number of rows to be stored in the table (rows)
- *var_num*
  Number of columns managed as branch rows (columns)
  Determine the number of the following types of columns:
  If the `BRANCH ALL` table option is specified
  • Number of `VARCHAR` and `VARBINARY` columns
  If the `BRANCH ALL` table option is not specified
  Total number of `VARCHAR` and `VARBINARY` columns that satisfy any of the following conditions:
  • Column for which `YES` is specified for the `BRANCH` column definition
  • Column whose defined length is 256 bytes or greater and for which `AUTO` is specified for the `BRANCH` column definition
  • Column whose defined length is 256 bytes or greater and for which the `BRANCH` column definition is omitted
- *page_size*
  Page size in the data DB area (bytes)
- *pctfree*
  Percentage of unused area specified in the `PCTFREE` operand of the `CREATE TABLE` statement (%)
  If the percentage of unused area is not specified, assume 30% for the calculation.
  To obtain the number of branch row pages of data to be added to a column store table by using the `INSERT` or `UPDATE` statement, assume 0% for the calculation.
  For *pctfree*, use the value that satisfies the formula in *pctfree* in (1) Determining the number of pages for base rows (variable BP(i)).
- *var_size*
  Average data length of columns in the branch rows (bytes)
  Determine the length based on the following table.

  Table 5-17: Data length of columns that become branch rows

Classification	Data type	Data length (bytes)
Character string data	`VARCHAR`	*d*

Classification	Data type	Data length (bytes)
Binary data	VARBINARY	*d*

Legend:

d: Actual data length

- *CTRLSIZE*

  If the target table is a row store table, specify 56.

  To obtain the number of branch row pages of data to be added to a column store table by using the INSERT or UPDATE statement, specify 80.

5.8.3 Determining the number of storage pages for each B-tree index segment

This subsection describes how to determine the number of storage pages for each B-tree index segment.

The B-tree index segments are as follows:

- Lower page segment
- Upper page segment

(1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i))

The number of storage pages used in the lower page segment (variable *IP_LOWER(i)*) can be determined by using the formula shown later.

If the null-value exclusion specification is specified for a B-tree index, exclude keys that are comprised of null values only from the formulas for determining the number of key types and the average number of duplicated key values.

Formula

```
IP_LOWER(i) = (PIDX_LEAF + PDUP) × R
```

Determine *PIDX_LEAF* from Formula 1.

Determine *PDUP* from Formula 2. *PDUPLIST* is the average number of row ID list pages for each key value, and *PDUPDIR* is the average number of row ID directory pages for each key value.

R is the number of duplicate keys in each B-tree index page, or a value between 1.2 and 1.5 if the key length varies. If the value converges around the average value, use 1.

Formula 1 (for determining PIDX_LEAF)

$$PIDX\_LEAF = \mathrm{MAX}\left(1, \uparrow \frac{key\_num + dup\_key\_num}{KNPP} \uparrow\right)$$

Formula 2 (for determining PDUP)

$$PDUP = (PDUPLIST + PDUPDIR) \times dup\_key\_num$$

$$PDUPLIST = \left\lceil \frac{dup\_key\_dup}{MAX\left(1, \left\lfloor \frac{\left\lfloor \frac{page\_size \times 50}{100} \right\rfloor - 94}{6} \right\rfloor\right)} \right\rceil$$

$$PDUPDIR = \left\lceil \frac{PDUPLIST}{MAX\left(1, \left\lfloor \frac{\left\lfloor \frac{page\_size \times 50}{100} \right\rfloor - 84}{16} \right\rfloor\right)} \right\rceil$$

Explanation of variables

- *key_num*
  Number of key types in which the number of duplicate key values is 255 or less (key types)
- *dup_key_num*
  Number of key types in which the number of duplicate key values is 256 or greater (key types)
- *dup_key_dup*
  Average number of duplicate keys in which the number of duplicate key values is 256 or greater (keys)
- *page_size*
  Page size of data DB area (bytes)
- *KNPP*
  Average number of keys stored per B-tree index page
  Use the following formula to determine this value.
  Formula (for determining KNPP)

$$KNPP = MAX\left(1, \left\lfloor \frac{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor - 80}{8 + KEYSZDB + 6 \times MIN(dbarea\_file\_num, key\_dup) + 6 \times key\_dup} \right\rfloor\right)$$

- *pctfree*
  Percentage of unused area specified in the `CREATE INDEX` statement (%)
  If the percentage of unused area is not specified, assume 30% for the calculation.
- *key_dup*
  Average number of duplicate keys in which the number of duplicate key values is 255 or less (keys)
- *dbarea_file_num*
  Number of files in the data DB area (number of files)
- *KEYSZDB*
  Length of keys stored in the database (bytes)
  To determine the key length, use the *KEYSZ* value determined in 5.8.4 Determining the key length (KEYSZ) of a B-tree index in the following formula:
  Formula (for determining KEYSZDB)

$$KEYSZDB = \left\lceil \frac{KEYSZ}{2} \right\rceil \times 2$$

(2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i))

The number of storage pages used in the upper page segment (variable *IP_UPPER(i)*) can be determined by using the following formula:

Formula

$$IP\_UPPER_{(i)} = \sum_{k=2}^{n} PIDX_{(k)} \times R$$

Determine *PIDX(k)* using the recursion formula shown in Formula 1. Always determine the value of *PIDX(2)*, and repeat the formula for *PIDX(k + 1)* until *PIDX(n)* = 1 is reached.

R is the number of duplicate keys in each B-tree index page, or a value between 1.2 and 1.5 if the key length varies. If the value converges around the average value, use 1.

Formula 1 (for determining PIDX(k))

$$PIDX_{(1)} = PIDX\_LEAF$$

$$PIDX_{(k+1)} = \left\lceil \frac{PIDX_{(k)}}{\mathrm{MAX}\left(2, \left\lfloor \frac{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor - 80}{12 + KEYSZDB} \right\rfloor\right)} \right\rceil$$

Explanation of variables

- *PIDX_LEAF*

  Determine this value by using *Formula 1 (for determining PIDX_LEAF)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)).
- *page_size*

  Page size of data DB area (bytes)
- *pctfree*

  Percentage of unused area specified in the `CREATE INDEX` statement (%)

  If the percentage of unused area is not specified, assume 30% for the calculation.
- *KEYSZDB*

  Length of keys stored in the database (bytes)

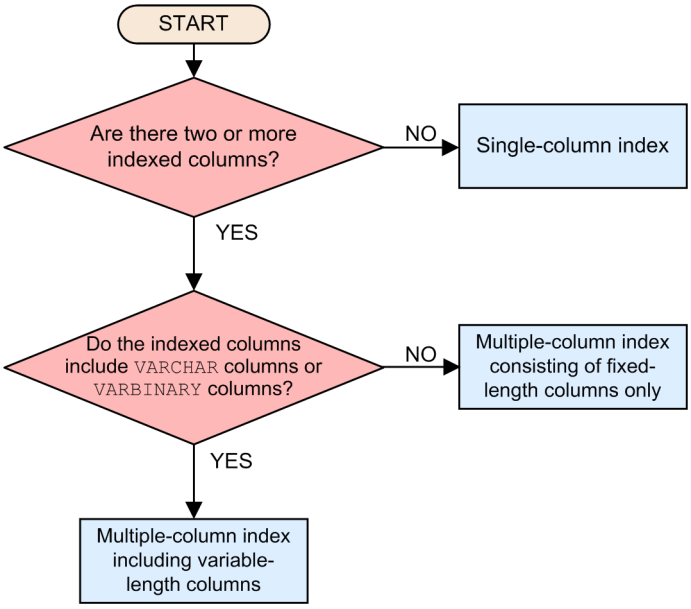
  Determine the value in the same way as when determining the variable *KEYSZDB* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)).

5.8.4 Determining the key length (KEYSZ) of a B-tree index

This subsection explains how to compute the key length (*KEYSZ*) of a B-tree index.

First, identify the B-tree index type based on the following figure.

Figure 5-12: Identifying the B-tree index type



Next, determine *KEYSZ* by using the formula for the identified B-tree index type. For the key length of each indexed column, see the following table.

Table 5-18: B-tree index key length

No.	Classification	Data type		Key length (bytes)
1	Numeric data	INTEGER		8
2		SMALLINT		4
3		DECIMAL(*m*, *n*)	$1 \leq m \leq 4$	2
4			$5 \leq m \leq 8$	4
5			$9 \leq m \leq 16$	8
6			$17 \leq m \leq 38$	16
7		DOUBLE PRECISION		8
8	Character string data	CHARACTER(*n*)		*n*
9		VARCHAR(*n*)		*d*
10	Datetime data	DATE		4
11		TIME(*p*)		$3 + \uparrow p \div 2 \uparrow$
12		TIMESTAMP(*p*)		$7 + \uparrow p \div 2 \uparrow$
13	Binary data	BINARY(*n*)		*n*
14		VARBINARY(*n*)		*d*

Legend:

d: Actual data length

m, *n*: Positive integers

p: 0, 3, 6, 9 or 12

(1) Single-column index

Formula (single-column index) (bytes)

```
KEYSZ = flag_size + key_size
```

Explanation of variables

- *flag_size*
  One of the following values (in bytes):
  - If the indexed column is not a `VARCHAR` or `VARBINARY` column: 0
  - If the indexed column is a `VARCHAR` or `VARBINARY` column with a defined length of no more than 255 bytes: 1
  - If the indexed column is a `VARCHAR` or `VARBINARY` column with a defined length of 256 or more bytes: 2
- *key_size*
  Key length of the indexed column (bytes)

(2) Multiple-column index comprised of fixed-length columns only

Formula (multiple-column index comprised of fixed-length columns only) (bytes)

$$KEYSZ = idx\_col\_num \times flag\_null + \sum_{i=1}^{N} key\_size_{(i)}$$

Explanation of variables

- *idx_col_num*
  Number of indexed columns
- *flag_null*
  Either of the following values (bytes):
  - For a key that does not include a null value in the indexed columns: 0
  - For a key that includes a null value in the indexed columns: 1
- $key\_size_{(i)}$
  Key length of i-th indexed column (bytes)

(3) Multiple-column index containing variable-length columns

Formula (multiple-column index containing variable-length columns) (bytes)

$$KEYSZ = (idx\_col\_num + 1) \times flag\_type + \sum_{i=1}^{N} key\_size_{(i)}$$

Explanation of variables

- *idx_col_num*

Number of indexed columns

- *flag_type*
  Either of the following values (bytes):
  • If the size of the columns comprising the B-tree index is 255 bytes or less: 1
  • If the size of the columns comprising the B-tree index is 256 bytes or greater: 2
  For details about the length of columns in a multiple-column index, see *CREATE INDEX (define an index)* in *Definition SQL* in the manual *HADB SQL Reference*.
- $key\_size_{(i)}$
  Key length of i-th indexed column (bytes)

5.8.5 Determining the number of storage pages for each text index segment

This subsection describes how to determine the number of storage pages for each text index segment.

The text index segments are as follows:

- String control segment
- Position control segment

(1) Determining the number of storage pages used in the string control segment (variable TIP_STRSEG(i))

The number of storage pages used in the string control segment (variable *TIP_STRSEG(i)*) can be determined by using the following formula:

Formula

$$TIP\_STRSEG_{(i)} = (TIXSTR + TIXCOR + TIXWORD) \times 1.25$$

Explanation of variables

TIXSTR

Total number of pages for the text index's string control

Use the following formula to determine its value.

When a text index is added for a meaningful character string (such as a sentence), the final value will approach the value obtained from the formula for variable *B*. When a text index is added for a character string that consists of unrelated characters (such as a random character string), the final value might not approach the value obtained from the formula for variable *B*.

Formula

TIXSTR= MAX（*A*， *B*）

Explanation of variables :

A : Use the following formula to determine its value.

$$\left\lceil \frac{row\_num \times string\_num \times 52}{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor} \right\rceil + 3$$

B : Use the following formula to determine its value.

$$\left\lceil \frac{18 \times 1{,}024^2}{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor} \right\rceil + 3$$

row_num

Number of rows that are stored in the base table for which the text index has been defined

string_num

Average number of characters stored in the columns in the text index

page_size

Page size of data DB area (bytes)

pctfree

Percentage of unused area specified in the `CREATE INDEX` statement (%)

If the percentage of unused area is not specified, assume 30% for the calculation.

TIXCOR

Total number of pages required for correction search using a text index

You must determine this variable if you add the notation-correction-search text-index specification when defining a text index.

Use the following formula to determine its value.

When a text index is added for a meaningful character string (such as a sentence), the final value will approach the value obtained from the formula for variable *B*. When a text index is added for a character string that consists of unrelated characters (such as a random character string), the final value might not approach the value obtained from the formula for variable *B*.

Formula

TIXCOR= MAX（*A*， *B*）

Explanation of variables :

A : Use the following formula to determine its value.

$$\left\lceil \frac{row\_num \times string\_num \times 116}{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor} \right\rceil + 4$$

B : Use the following formula to determine its value.

$$\left\lceil \frac{29 \times 1{,}024^2}{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor} \right\rceil + 4$$

row_num

Number of rows that are stored in the base table for which the text index has been defined

string_num

Average number of characters stored in the columns in the text index

page_size

Page size of data DB area (bytes)

pctfree

Percentage of unused area specified in the `CREATE INDEX` statement (%)

If the percentage of unused area is not specified, assume 30% for the calculation.

TIXWORD

Total number of pages required for word-context search using a text index

You must determine this value if you add the text-index-word-context search specification when defining a text index.

Use the following formula to determine this value.

When a text index is added for a meaningful character string (such as a sentence), the final value will approach the value obtained from the formula for variable *B*. When a text index is added for a character string that consists of unrelated characters (such as a random character string), the final value might not approach the value obtained from the formula for variable *B*.

Formula

$$TIXWORD = \mathrm{MAX}(A, B)$$

Explanation of variables

A: Determine using the following formula:

$$\left\lceil \frac{row\_num \times string\_num \times (80 + 54 \times subarea\_num)}{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor} \right\rceil$$

B: Determine using the following formula:

$$\left\lceil \frac{(260 + 205 \times subarea\_num) \times 1{,}024^2 + 5{,}040 \times datapage\_num}{\left\lfloor \frac{page\_size \times (100 - pctfree)}{100} \right\rfloor} \right\rceil$$

row_num

Number of rows that are stored in the base table for which the text index has been defined

string_num

Average number of characters stored in the columns in the text index

subarea_num

Total number of data DB area files that make up the data DB area in which the base table for which the text index has been defined is stored

page_size

Page size of data DB area (bytes)

datapage_num

Number of data pages for the base table for which the text index has been defined (sum of the number of base row pages and the number of branch row pages)

pctfree

Percentage of unused area specified in the `CREATE INDEX` statement (%)

If the percentage of unused area is not specified, assume 30% for the calculation.

(2) Determining the number of storage pages used in the position control segment (variable TIP_APPSEG(i))

The number of storage pages used in the position control segment (variable *TIP_APPSEG(i)*) can be determined by using the following formula:

Formula

```
TIP_APPSEG(i) = TIXAPP × 1.25
```

Explanation of variables

TIXAPP

Total number of pages for the text index's position control

Use the following formula to determine this value.

When a text index is added for a meaningful character string (such as a sentence), the final value will approach the value obtained from the formula for variable *B*. When a text index is added for a character string that consists of unrelated characters (such as a random character string), the final value might not approach the value obtained from the formula for variable *B*.

Formula

TIXAPP= MAX（*A*， *B*）

Explanation of variables :

A：Use the following formula to determine its value.

$$\left\uparrow \frac{row\_num \times string\_num \times 54 \times subarea\_num}{\left\downarrow \frac{page\_size \times (100 - pctfree)}{100} \right\downarrow} \right\uparrow + 6$$

B：Use the following formula to determine its value.

$$\left\uparrow \frac{198 \times 1{,}024^2 \times subarea\_num + 5{,}040 \times datapage\_num}{\left\downarrow \frac{page\_size \times (100 - pctfree)}{100} \right\downarrow} \right\uparrow + 6$$

row_num

Number of rows that are stored in the base table for which the text index has been defined

string_num

Average number of characters stored in the columns in the text index

page_size

Page size of data DB area (bytes)

subarea_num

Total number of data DB area files that make up the data DB area in which the base table for which the text index has been defined is stored

datapage_num

Number of data pages for the base table for which the text index has been defined (sum of the number of base row pages and the number of branch row pages)

pctfree

Percentage of unused area specified in the `CREATE INDEX` statement (%)

If the percentage of unused area is not specified, assume 30% for the calculation.

5.8.6 Determining the number of segments for storing each range index

Use the following formula to determine the number of segments needed to store each range index (*RS(i)* variable).

Formula (segments)

```
RS(i) = SGRI + SGSAM
```

Explanation of variables

SGRI: See (1) Determining the SGRI variable.

SGSAM: See (2) Determining the SGSAM variable.

(1) Determining the SGRI variable

Use the following formula to determine the *SGRI* variable.

Formula (segments)

$$SGRI = \left\uparrow \frac{(PGGRPRI + PGGRPSGM) \times PGGRPSIZE}{SEGSIZE} \right\uparrow$$

Explanation of variables

SEGSIZE

Segment size in the DB area in which a range index is defined (pages)

Use the following formula to determine this value:

Formula

$$SEGSIZE = 4{,}194{,}304 \div page\_size$$

page_size

Page size of the DB area that stores range indexes (bytes)

PGGRPRI

Use the following formula to determine this value.

Formula (page groups)

$$PGGRPRI = \left\uparrow \frac{DBASGM - SUBAREANUM + 1}{\left\downarrow \frac{(page\_size - 64) \times 8}{col\_size \times 16 + 1} \right\downarrow \times PGGRPSIZE} \right\uparrow + SUBAREANUM - 1$$

DBASGM

Use the following formula to determine this value.

Formula (segments)

$$DBASGM = SGTBL + SGIDX + SGTIX + \sum_{i=1}^{rng\_num\_in\_table\_dbarea} ASGRI_{(i)}$$

SUBAREANUM

Use the following formula to determine this value.

Formula (segments)

$$SUBAREANUM = \sum_{i=1}^{cmd\_num} \left(tbl\_dbareafile\_num_{(i)} \times \left\uparrow \frac{tbl\_dbarea\_initsize_{(i)}}{16,384} \right\uparrow \right)$$

cmd_num

The total number of times the following commands were used to initialize or add a DB area that stores the tables for which range indexes are defined

- `adbinit` command
- `adbmodarea` command

tbl_dbareafile_num(i)

The number of DB area files created when the i-th use of the following commands were used to initialize or add a DB area that stores the tables for which range indexes are defined (files)

- `adbinit` command
- `adbmodarea` command

tbl_dbarea_initsize(i)

The size specified for the initial allocation size option of the i-th use of the following commands to initialize or add the DB area that stores the tables for which range indexes are defined (gigabytes)

- `adbinit` command
- `adbmodarea` command

SGTBL

The number of segments that will store the table for which a range index has been defined and the tables stored in the same data DB area as for the former (segments)

See the following subsections in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the table.

- Single-chunk table that is a row store table
  (a) Determining the variable SGROWTBL (for a single-chunk table)
- Single-chunk table that is a column store table
  (b) Determining the variable SGCOLUMNTBL (for a single-chunk table)
- Multi-chunk table that is a row store table
  (f) Determining the variable SGROWTBL (for a multi-chunk table)
- Multi-chunk table that is a column store table
  (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table)

SGIDX

Number of segments in the B-tree index stored in the same data DB area that stores the tables for which range indexes are defined (segments)

See the following subsections in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the table.

- Single-chunk table
  (c) Determining the variable SGIDX (for a single-chunk table)

- Multi-chunk table

  (h) Determining the variable SGIDX (for a multi-chunk table)

SGTIX

Number of segments in the text index stored in the same data DB area that stores the tables for which range indexes are defined (segments)

See the following subsections in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the table.

- Single-chunk table

  (e) Determining the variable SGTIX (for a single-chunk table)

- Multi-chunk table

  (j) Determining the variable SGTIX (for a multi-chunk table)

rng_num_in_table_dbarea

Number of range indexes in the data DB area that stores the tables for which range indexes are defined (indexes)

ASGRI(i)

Number of segments in the range index stored in the same data DB area that stores the tables for which range indexes are defined (segments)

Formula (segments)

$$ASGRI_{(i)} = SGSAM + \left\uparrow \frac{SUBAREANUM \times PGGRPSIZE \times 2}{SEGSIZE} \right\uparrow + \left\uparrow \frac{SGTBL}{65{,}536} \right\uparrow$$

SGSAM

For details, see (2) Determining the SGSAM variable.

SGTBL

The number of segments in the table for which the i-th range index is defined, and which is stored in the same data DB area as the tables for which range indexes are defined (files)

See the following subsections in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the table.

- Single-chunk table that is a row store table

  (a) Determining the variable SGROWTBL (for a single-chunk table)

- Single-chunk table that is a column store table

  (b) Determining the variable SGCOLUMNTBL (for a single-chunk table)

- Multi-chunk table that is a row store table

  (f) Determining the variable SGROWTBL (for a multi-chunk table)

- Multi-chunk table that is a column store table

  (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table)

col_size

Data length of the columns for which range indexes are defined (bytes)

To determine the data length of each column, see Table 5-15: Data length of each data type in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

PGGRPSGM

Use the following formula to determine this value.

Formula (page groups)

$$PGGRPSGM = \left\uparrow \frac{PGGRPRI}{\left\downarrow \frac{(page\_size - 64) \times 8}{65} \right\downarrow \times PGGRPSIZE} \right\uparrow + SUBAREANUM - 1$$

PGGRPSIZE

The variable *PGGRPSIZE* is the smallest power of 2 that is equal to or greater than the variable *TMPPGGRPSIZE*, which is determined using the formula below.

However, if the targeted range index was defined before version 02-02 and the `adbidxrebuild` command has never been executed on it, substitute the value of the variable *SEGSIZE* for the variable *PGGRPSIZE*.

Formula

$$TMPPGGRPSIZE = \left\uparrow \frac{8{,}192 \times \sqrt{col\_size}}{page\_size} \right\uparrow$$

(2) Determining the SGSAM variable

Use the following formula to determine the *SGSAM* variable.

Formula (segments)

$$SGSAM = \left\uparrow \frac{SUBAREAPTRNUM + \left\uparrow \frac{col\_size}{4} \right\uparrow}{\left\downarrow \frac{(page\_size - 64) \times 8}{65} \right\downarrow \times SEGSIZE} \right\uparrow$$

Explanation of variables

SUBAREAPTRNUM

Use the following formula to determine this value.

Formula (segments)

$$SUBAREAPTRNUM = \sum_{i=1}^{cmd\_num} \left(tbl\_dbareafile\_num_{(i)} \right) + \left\downarrow \frac{tbl\_dbarea\_max\_initsize}{16{,}384} \right\downarrow \times 8{,}192$$

cmd_num

The total number of times the following commands were used to initialize or add a DB area that stores the tables for which range indexes are defined

- `adbinit` command
- `adbmodarea` command

tbl_dbareafile_num(i)

The number of DB area files created when the i-th iteration of the following commands were used to initialize or add a DB area that stores the tables for which range indexes are defined (files)

- `adbinit` command
- `adbmodarea` command

tbl_dbarea_max_initsize

The largest size specified for the initial allocation size option among all uses of the following commands to initialize or add the DB area that stores the tables for which range indexes are defined (gigabytes)

- `adbinit` command
- `adbmodarea` command

col_size

Data length of a column for which a range index is defined (bytes)

To determine the data length of each column, see Table 5-15: Data length of each data type in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

If the targeted range index is a range index that cannot skip chunks, substitute `0` for the variable *col_size*. To check whether the targeted range index can skip chunks, see 11.3.6 Checking a range index (whether it can skip chunks).

page_size

Page size of the DB area that stores range indexes (bytes)

SEGSIZE

Segment size (pages)

Use the following formula to determine this value:

Formula

SEGSIZE = 4,194,304 ÷ *page_size*

5.9 Estimating the size of the work table DB area

Use the following formula to determine the size of the work table DB area.

Formula (kilobytes)

```
work table DB area size = WRK_PAGE_NUM × page_size ÷ 1,024
```

Explanation of variables

WRK_PAGE_NUM: Total number of pages in the work table DB area

Determine the value as explained in 5.9.1 Determining the total number of pages in the work table DB area.

page_size: Page size in the work table DB area

Substitute the value specified for the initialization option `adb_init_wrk_page_size` determined in 5.7 Designing a work table DB area.

Note that the `adb_init_wrk_page_size` option is specified in kilobytes, while the *page_size* variable is specified in bytes.

For details about the initialization option `adb_init_wrk_page_size`, see *adbinit (Initialize the Database)* in the manual *HADB Command Reference*.

5.9.1 Determining the total number of pages in the work table DB area

Use the following formula to determine the total number of pages in the work table DB area (variable *WRK_PAGE_NUM*).

Formula

$$WRK\_PAGE\_NUM = 1 + dbarea\_file\_num + \sum_{i=1}^{tbl\_num} WP_{(i)}$$

Explanation of variables

dbarea_file_num

Number of work table DB area files (files)

Assign `1` to the `dbarea_file_num` variable because there is only one work table DB area file.

tbl_num

Total number of work tables to be created at the same time (tables)

WP(i)

Number of pages required for work tables that will store base rows

For details about the variable *WP(i)*, see 5.9.2 Determining the number of pages for base rows that are needed for storing work tables.

5.9.2 Determining the number of pages for base rows that are needed for storing work tables

Use the following formula to determine the number of pages for base rows that are needed for storing the work tables (*WP(i)*).

Formula

$$WP_{(i)} = \left\uparrow \frac{row\_num}{\left\downarrow \frac{page\_size - 64}{ROWSZ} \right\downarrow} \right\uparrow$$

Explanation of variables

row_num

Number of rows to be stored in the table (rows)

page_size

Page size in the work table DB area (bytes)

Substitute the value specified for the initialization option `adb_init_wrk_page_size` determined in 5.7 Designing a work table DB area.

The unit for the value of `adb_init_wrk_page_size` is kilobytes and the unit for the value of the `page_size` variable is bytes. Note this difference in the units.

For details about the initialization option `adb_init_wrk_page_size`, see *adbinit (Initialize the Database)* in the manual *HADB Command Reference*.

Use the following formula to determine the variable *ROWSZ*.

If the value determined for *ROWSZ* by this formula is greater than 32,704, add the value of 2 x (*col_num* + 1) to the result of the formula.

Formula

$$ROWSZ = \left\uparrow \frac{28 + 2 \times col\_num + \sum_{i=1}^{col\_num} col\_size_{(i)}}{2} \right\uparrow \times 2$$

Explanation of variables

col_num: Total number of columns in the work table (columns)

col_size(i): Data length of each column (bytes)

For the variable *col_size(i)*, determine the data length of each column from the table shown below. Then, calculate the sum total for all columns.

Table 5-19: Data length of each data type

No.	Classification	Data type		Data length (bytes)
1	Numeric data	`INTEGER`		8
2		`SMALLINT`		4
3		`DECIMAL`(m, n)#	$1 \leq m \leq 4$	2

No.	Classification	Data type		Data length (bytes)
4			$5 \leq m \leq 8$	4
5			$9 \leq m \leq 16$	8
6			$17 \leq m \leq 38$	16
7		DOUBLE PRECISION		8
8	Character string data	CHARACTER(*n*)		*n*
9		VARCHAR(*n*)		*wrk_var_col_size*
10	Datetime data	DATE		4
11		TIME(*p*)		$3 + \uparrow p \div 2 \uparrow$
12		TIMESTAMP(*p*)		$7 + \uparrow p \div 2 \uparrow$
13	Binary data	BINARY(*n*)		*n*
14		VARBINARY(*n*)		$d + 2$

Legend:

m, *n*: Positive integers

d: Actual data length

p: 0, 3, 6, 9 or 12

#

Indicates a fixed-point number that has a total of *m* digits, with *n* digits following the decimal point. If *m* is omitted, 38 is assumed.

wrk_var_col_size: Data length of VARCHAR columns for work tables

Determine the data length based on the following table:

Table 5-20: Data lengths of VARCHAR columns for work tables

No.	Definition length n (bytes)	Data length (bytes)
1	$1 \leq n \leq 32{,}000$	$d + 2$
2	$32{,}001 \leq n \leq 64{,}000$	$d + 8$

Legend:

n: Positive integer

d: Actual data length

5.10 Estimating the size of the master directory DB area

Estimate the size of the master directory DB area to be 8.1 megabyte.

A single master directory DB area file (file name: `ADBMST`), which constitutes the master directory DB area, is created using a regular file under the DB directory. It can also be created using a block special file that is not under the DB directory.

5.11 Estimating the size of the dictionary DB area

Use the following formula to determine the dictionary DB area size.

Formula (kilobytes)

dictionary DB area size ＝
　　ADBDICSIZE_INIT＋*ADBDICSIZE_OPERATION*

Explanation of variables

ADBDICSIZE_INIT

Size of the dictionary DB area determined by the database size

Use the following formula to determine this value.

Formula (kilobytes)

ADBDICSIZE_INIT＝
　　↑8,200＋*USERSCM*×1.1＋*USERTBL*×9,000＋*USERIDX*×31＋*USERVIEW*×1,100
　　＋*DBAREANUM*＋*USERNUM*×71＋*USERFKEYNUM*×42＋*USERARCHIVE*×82↑

USERSCM

Number of user-defined schemas

USERTBL

Number of user-defined base tables

The value of the variable *USERTBL* includes the number of archivable multi-chunk tables.

USERIDX

Number of user-defined indexes

USERVIEW

Number of user-defined viewed tables

DBAREANUM

Number of DB areas

USERNUM

Number of HADB users created using the CREATE USER statement + 1

1 indicates the HADB user that was created during execution of the `adbinit` command.

USERFKEYNUM

Number of foreign keys defined by the HADB user

USERARCHIVE

Number of archivable multi-chunk tables

ADBDICSIZE_OPERATION

Size of the dictionary DB area determined according to database operation

Use the following formula to determine this value.

Formula (kilobytes)

ADBDICSIZE_OPERATION＝
　↑*CREATE_USER*＋*CREATE_SCHEMA*＋*CREATE_TABLE*＋*CREATE_VIEW*＋*CREATE_INDEX*
　　＋*CREATE_AUDIT*＋*GRANT*＋*ALTER_USER*＋*ALTER_TABLE*＋*ALTER_VIEW*＋*DROP_USER*
　　＋*DROP_SCHEMA*＋*DROP_TABLE*＋*DROP_VIEW*＋*DROP_INDEX*＋*REVOKE*↑

CREATE_USER

Use the following formula to determine this value.

Formula (kilobytes)

CREATE_USER = 4 x *CREATE_USER_EXEC_NUM*

CREATE_USER_EXEC_NUM

Substitute the number of times the `CREATE USER` statement is executed.

CREATE_SCHEMA

Use the following formula to determine this value.

Formula (kilobytes)

CREATE_SCHEMA = 4 x *CREATE_SCHEMA_EXEC_NUM*

CREATE_SCHEMA_EXEC_NUM

Substitute the number of times the `CREATE SCHEMA` statement is executed.

CREATE_TABLE

Use the following formula to determine this value.

Formula (kilobytes)

CREATE_TABLE = 596 x *CREATE_TABLE_EXEC_NUM*

CREATE_TABLE_EXEC_NUM

Substitute the number of times the `CREATE TABLE` statement is executed.

CREATE_VIEW

Use the following formula to determine this value.

Formula (kilobytes)

CREATE_VIEW = 742 x *CREATE_VIEW_EXEC_NUM*

CREATE_VIEW_EXEC_NUM

Substitute the number of times the `CREATE VIEW` statement is executed.

CREATE_INDEX

Use the following formula to determine this value.

Formula (kilobytes)

CREATE_INDEX = 24 x *CREATE_INDEX_EXEC_NUM*

CREATE_INDEX_EXEC_NUM

Substitute the number of times the `CREATE INDEX` statement is executed.

CREATE_AUDIT

Use the following formula to determine the value:

Formula (kilobytes)

CREATE_AUDIT = 472 × *CREATE_AUDIT_EXEC_NUM*

CREATE_AUDIT_EXEC_NUM

Substitute the number of times the `CREATE AUDIT` statement is executed.

GRANT

Use the following formula to determine this value.

Formula (kilobytes)

GRANT = 656 x *GRANT_EXEC_NUM*

GRANT_EXEC_NUM

Substitute the number of times the `GRANT` statement is executed.

ALTER_USER

Use the following formula to determine this value.

Formula (kilobytes)

ALTER_USER = 4 x *ALTER_USER_EXEC_NUM*

ALTER_USER_EXEC_NUM

Substitute the number of times the `ALTER USER` statement is executed.

ALTER_TABLE

Use the following formula to determine this value.

Formula (kilobytes)

ALTER_TABLE =
ADD_COLUMN + *RENAME_COLUMN* + *CHANGE_CHUNK_MAX*
+ *CHANGE_CHUNK_ARCHIVE* + *CHANGE_CHUNK_UNARCHIVE*

ADD_COLUMN

Use the following formula to determine this value.

Formula (kilobytes)

ADD_COLUMN = 36 x *ADD_COLUMN_EXEC_NUM*

ADD_COLUMN_EXEC_NUM

Substitute the number of times the `ALTER TABLE` statement is used to add a column to a table.

RENAME_COLUMN

Use the following formula to determine this value.

Formula (kilobytes)

RENAME_COLUMN = 76 x *RENAME_COLUMN_EXEC_NUM*

RENAME_COLUMN_EXEC_NUM

Substitute the number of times the `ALTER TABLE` statement is used to rename a table column.

CHANGE_CHUNK_MAX

Use the following formula to determine this value.

Formula (kilobytes)

CHANGE_CHUNK_MAX = 4 x *CHANGE_CHUNK_MAX_EXEC_NUM*

CHANGE_CHUNK_MAX_EXEC_NUM

Substitute the number of times the `ALTER TABLE` statement is used to change the maximum number of chunks in a multi-chunk table.

CHANGE_CHUNK_ARCHIVE

Use the following formula to determine this value.

Formula (kilobytes)

CHANGE_CHUNK_ARCHIVE = 84 x *CHANGE_CHUNK_ARCHIVE_EXEC_NUM*

CHANGE_CHUNK_ARCHIVE_EXEC_NUM

Substitute the number of times the `ALTER TABLE` statement is used to change a regular multi-chunk table to an archivable multi-chunk table.

CHANGE_CHUNK_UNARCHIVE

Use the following formula to determine this value.

Formula (kilobytes)

CHANGE_CHUNK_UNARCHIVE = 48 x *CHANGE_CHUNK_UNARCHIVE_EXEC_NUM*

CHANGE_CHUNK_UNARCHIVE_EXEC_NUM

Substitute the number of times the `ALTER TABLE` statement is used to change an archivable multi-chunk table to a regular multi-chunk table.

ALTER_VIEW

Use the following formula to determine this value.

Formula (kilobytes)

ALTER_VIEW = 678 x *ALTER_VIEW_EXEC_NUM*

ALTER_VIEW_EXEC_NUM

Substitute the number of times the `ALTER VIEW` statement is executed.

DROP_USER

Use the following formula to determine this value.

Formula (kilobytes)

DROP_USER = 1,992 x *DROP_USER_EXEC_NUM*

DROP_USER_EXEC_NUM

Substitute the number of times the `DROP USER` statement is executed.

DROP_SCHEMA

Use the following formula to determine this value.

Formula (kilobytes)

DROP_SCHEMA = 1,992 x *DROP_SCHEMA_EXEC_NUM*

DROP_SCHEMA_EXEC_NUM

Substitute the number of times the `DROP SCHEMA` statement is executed.

DROP_TABLE

Use the following formula to determine this value.

Formula (kilobytes)

DROP_TABLE = 204 x *DROP_TABLE_EXEC_NUM*

DROP_TABLE_EXEC_NUM

Substitute the number of times the `DROP TABLE` statement is executed.

DROP_VIEW

Use the following formula to determine this value.

Formula (kilobytes)

DROP_VIEW = 36 x *DROP_VIEW_EXEC_NUM*

DROP_VIEW_EXEC_NUM

Substitute the number of times the `DROP VIEW` statement is executed.

DROP_INDEX

Use the following formula to determine this value.

Formula (kilobytes)

DROP_INDEX = 8 x *DROP_INDEX_EXEC_NUM*

DROP_INDEX_EXEC_NUM

Substitute the number of times the `DROP INDEX` statement is executed.

REVOKE

Use the following formula to determine this value.

Formula (kilobytes)

REVOKE = 1,992 x *REVOKE_EXEC_NUM*

REVOKE_EXEC_NUM

Substitute the number of times the `REVOKE` statement is executed.

A single dictionary DB area file (file name: `ADBDIC`), which constitutes the dictionary DB area, is created using a regular file under the DB directory. It can also be created using a block special file that is not under the DB directory.

5.12 Estimating the size of the system-table DB area

Use the following formula to determine the system-table DB area size.

Formula (kilobytes)

```
system-table DB area size ＝
    10,000＋STBLSIZE_SUM＋REORGSDAREASIZE
```

Explanation of variables

STBLSIZE_SUM

Total size of each system table

Use the following formula to determine this value.

Formula (kilobytes)

```
STBLSIZE_SUM＝
  STBLTABLESSIZE＋STBLCOLUMNSSIZE＋STBLINDEXESSIZE
   ＋STBLCHUNKSSIZE＋STBLSYNONYMDICSIZE
```

STBLTABLESSIZE

Size of the table `STATUS_TABLES`

STBLCOLUMNSSIZE

Size of the table `STATUS_COLUMNS`

STBLINDEXESSIZE

Size of the table `STATUS_INDEXES`

STBLCHUNKSSIZE

Size of the table `STATUS_CHUNKS`

STBLSYNONYMDICSIZE

Size of the table `STATUS_SYNONYM_DICTIONARIES`

Determine the size of each system table as explained in 5.12.1 Estimating the sizes of system tables.

REORGSDAREASIZE

Size of the area for executing the `adbreorgsystemdata` command

Add this variable when executing the `adbreorgsystemdata` command.

Use the following formula to determine this value.

Formula (kilobytes)

```
REORGSDAREASIZE＝
  MAX（STBLTABLESSIZE，STBLCOLUMNSSIZE，STBLINDEXESSIZE
      ，STBLCHUNKSSIZE，STBLSYNONYMDICSIZE）
```

The largest of the following system table sizes:

STBLTABLESSIZE

Size of the table `STATUS_TABLES`

STBLCOLUMNSSIZE

Size of the table `STATUS_COLUMNS`

STBLINDEXESSIZE

Size of the table STATUS_INDEXES

STBLCHUNKSSIZE

Size of the table STATUS_CHUNKS

STBLSYNONYMDICSIZE

Size of the table STATUS_SYNONYM_DICTIONARIES

Determine the size of each system table as explained in 5.12.1 Estimating the sizes of system tables.

A single system-table DB area file (file name: ADBSTBL), which constitutes the system-table DB area, is created using a regular file under the DB directory. It can also be created using a block special file that is not under the DB directory.

5.12.1 Estimating the sizes of system tables

The following describes how to estimate the sizes of the following system tables:

- STATUS_TABLES table
  Determine the value as explained in (1) Estimating the size of the STATUS_TABLES table.
- STATUS_COLUMNS table
  Determine the value as explained in (2) Estimating the size of the STATUS_COLUMNS table.
- STATUS_INDEXES table
  Determine the value as explained in (3) Estimating the size of the STATUS_INDEXES table.
- STATUS_CHUNKS table
  Determine the value as explained in (4) Estimating the size of the STATUS_CHUNKS table.
- STATUS_SYNONYM_DICTIONARIES table
  Determine the value as explained in (5) Estimating the size of the STATUS_SYNONYM_DICTIONARIES table.

(1) Estimating the size of the STATUS_TABLES table

The following shows the formula to calculate the size of the STATUS_TABLES table (the value of *STBLTABLESSIZE*).

If you do not execute the adbgetcst command, substitute 0.

Formula (kilobytes)

$$STBLTABLESSIZE = 0.4 \times COSTTBL\_NUM \times (1 + GETCOST\_NUM)$$

Explanation of the variables

COSTTBL_NUM

Specify the number of tables in which to collect cost information.

GETCOST_NUM

Specify the number of times the adbgetcst command is executed per table.

Estimate how many times the adbgetcst command will be executed per execution of the adbreorgsystemdata command.

For details about the interval for executing the adbreorgsystemdata command, see 11.17.2 Timing for reorganizing a system table.

The following shows an estimation example, which assumes the following operational conditions.

▪ Operational conditions

- Cost information is collected once a day.
- The system tables (base tables) are reorganized once every six months.

▪ How to obtain GETCOST_NUM based on the conditions

GETCOST_NUM = *A* x *B*

■ Explanation of variables
A: Frequency (number of times) of adbgetcst command executed per day.
Substitute 1.
(Assume that the command is executed once a day.)
B: Interval (number of days) for executing the adbreorgsystemdata command is executed.
Substitute 186.
(Obtained by 6 x 31, assuming that the command is executed once every six months and a month has 31 days.)

To perform operation under the preceding conditions, substitute 186 for the variable *GETCOST_NUM*.

(2) Estimating the size of the STATUS_COLUMNS table

The following shows the formula to calculate the size of the STATUS_COLUMNS table (the value of *STBLCOLUMNSSIZE*).

If you do not execute the adbgetcst command, substitute 0.

Formula (kilobytes)

STBLCOLUMNSSIZE =
6.4 × (*COSTCOL_NUM* × (1 + *GETCOST_NUM*) + *ALTERCOL_NUM*)

Explanation of the variables

COSTCOL_NUM

Specify the total number of table columns in which to collect cost information.

For example, if you collect cost information in 10 columns of table A and in 20 columns of table B, the value of this variable is 30.

GETCOST_NUM

For details, see the description of the variable *GETCOST_NUM* in (1) Estimating the size of the STATUS_TABLES table.

ALTERCOL_NUM

Specify the number of times a column is renamed by using the ALTER TABLE statement.

(3) Estimating the size of the STATUS_INDEXES table

The following shows the formula to calculate the size of the STATUS_INDEXES table (the value of *STBLINDEXESSIZE*).

If you do not execute the adbgetcst command, substitute 0.

Formula (kilobytes)

```
STBLINDEXESSIZE=
  0.4 × COSTIDX_NUM × (1 + GETCOST_NUM)
```

Explanation of the variables

COSTIDX_NUM

Specify the total number of indexes that are defined for the tables in which to collect cost information.

For example, if you collect cost information in table `A` with two indexes defined and in table `B` with one index defined, the value of this variable is 3.

GETCOST_NUM

For details, see the description of the variable *GETCOST_NUM* in (1) Estimating the size of the STATUS_TABLES table.

(4) Estimating the size of the STATUS_CHUNKS table

The following shows the formula to calculate the size of the `STATUS_CHUNKS` table (the value of *STBLCHUNKSSIZE*).

If you do not define multi-chunk tables, substitute 0.

Formula (kilobytes)

```
STBLCHUNKSSIZE=
  (0.4 + CHUNK_COMMENT_SIZE) × STBLCHUNKS_ROWNUM
```

Explanation of the variables

CHUNK_COMMENT_SIZE

Specify the average data length of each chunk comment (kilobytes).

STBLCHUNKS_ROWNUM

Specify the number of rows in the `STATUS_CHUNKS` table.

Use the following formula to determine the value.

Formula

```
STBLCHUNKS_ROWNUM=TOTAL_CHUNKNUM+STBLCHUNK_INVALID_ROWNUM
```

TOTAL_CHUNKNUM

Specify the total number of chunks in the multi-chunk tables that are used for operation.

For example, if you are using multi-chunk table `A` that has 700 chunks and multi-chunk table `B` that has 600 chunks, the value of this variable is 1,300.

STBLCHUNK_INVALID_ROWNUM

Specify the number of invalid-data rows that arise in the `STATUS_CHUNKS` table.

Estimate how many invalid-data rows will arise in the `STATUS_CHUNKS` table in the time between the execution of the `adbreorgsystemdata` command and the re-execution of the command. The following shows the formula:

Formula

```
STBLCHUNK_INVALID_ROWNUM= reorg_period × backgroundimport_table_num × INVALID_ROW_NUM
```

reorg_period

Specify the interval for executing the adbreorgsystemdata command (months).

For example, if the adbreorgsystemdata command is executed once every six months, substitute 6 for this variable.

For details about the interval for executing the adbreorgsystemdata command, see 11.17.2 Timing for reorganizing a system table.

backgroundimport_table_num

Specify the number of multi-chunk tables subject to background import.

For example, if background import is performed for five multi-chunk tables every day, substitute 5.

INVALID_ROW_NUM

Specify the number of invalid-data rows that will arise per month for one multi-chunk table.

First, estimate the total number of SQL statements and commands that will be executed for a multi-chunk table. Then, obtain the number of invalid-data rows that will arise from the following table.

Table 5-21: Number of invalid-data rows that arise in the STATUS_CHUNKS table when an SQL statement or command is executed

No.	SQL statement or command executed for a multi-chunk table	Number of invalid-data rows that arise in the STATUS_CHUNKS table[#1]
1	Execution of a definition SQL statement (when executed to delete multi-chunk tables)	Total number of chunks in the deletion-target tables
2	Execution of a PURGE CHUNK statement	Number of deleted chunks
3	Execution of a TRUNCATE TABLE statement	Number of chunks in the processing-target tables
4	Execution of a adbimport command (when executed in creation mode)	Number of chunks in the processing-target tables
5	Execution of a adbimport command (when executed for background import)	1
6	Execution of a adbidxrebuild command (when background import started by the adbimport command is interrupted)	1
7	Execution of a adbmergechunk command	Number of chunks to be merged × 2[#2]
8	Execution of a adbchgchunkcomment command	1
9	Execution of a adbchgchunkstatus command	Number of chunks whose status is changed
10	Execution of a adbarchivechunk command	Number of archived chunks
11	Execution of a adbunarchivechunk command	Number of unarchived chunks

#1

The value in this column indicates the number of invalid-data rows that will arise when the SQL statement or command executed for a multi-chunk table in the left column is executed once. If the statement or command is executed multiple times, multiply the value by the number of times the statement or command is executed.

#2

Invalid row data does not arise if merge-source chunks cannot be deleted for the following reasons:

• The adbmergechunk command is interrupted after the merge-target chunk is ready but before the merge-source chunks are deleted

• A merge-chunk target table is already being referenced when the adbmergechunk command is executed with NOWAIT specified in the --purge-chunk option

The following shows the operational conditions of an example operation, and describes how to obtain the value of the variable *STBLCHUNK_INVALID_ROWNUM* when the operation is executed.

▪ Operational conditions

- The system tables (base tables) are reorganized once every six months.
- Background import is performed for 10 multi-chunk tables.
- Background import is performed once every five minutes (288 times per day).
- The 288 chunks created by background import are merged as a daily chunk.
- 31 daily chunks are merged as a monthly chunk at each month end (assuming 31 days per month).
- Each monthly chunk is deleted at the end of a month after two years since creation.

▪ To obtain STBLCHUNK_INVALID_ROWNUM under the indicated conditions

For the variable *reorg_period*, which indicates the interval for executing the adbreorgsystemdata command, substitute 6.

For the variable *backgroundimport_table_num*, which indicates the number of multi-chunk tables subject to background import, substitute 10.

For the variable *INVALID_ROW_NUM*, which indicates the number of invalid-data rows that arise per month for one multi-chunk table, substitute the value obtained from the following formula:

Formula to obtain INVALID_ROW_NUM

INVALID_ROW_NUM = *backgroundimport_exec_num* + *mergechunk_chunk_num* x 2 + *purgechunk_chunk_num*

■ Explanation of variables

backgroundimport_exec_num
Number of times background import is performed in one month.
Substitute 8,928.
(Obtained by 288 x 31, assuming that background import is performed 288 times per day and that a month has 31 days.)

mergechunk_chunk_num
Number of chunks merged in one month.
Substitute 8,959.
(Obtained by 288 x 31 + 31, assuming that 288 chunks are merged per day, a month has 31 days, and 31 daily chunks are merged at the month end.)

purgechunk_chunk_num
Number of chunks deleted in one month.
Substitute 1.
(Assume that a monthly chunk that was created two years ago is deleted every month.)

For the variable *INVALID_ROW_NUM*, substitute 26,847, which is obtained from the preceding formula.

Obtain the value of *STBLCHUNK_INVALID_ROWNUM* by substituting the obtained values for the variables on the right side. In the preceding example, the number of invalid-data rows that will arise in the STATUS_CHUNKS table over a six-month period is estimated to be 1,610,820.

(5) Estimating the size of the STATUS_SYNONYM_DICTIONARIES table

The following shows the formula to calculate the size of the STATUS_SYNONYM_DICTIONARIES table (the value of *STBLSYNONYMDICSIZE*).

If you do not define any synonym dictionaries, substitute 0.

Formula (kilobytes)

STBLSYNONYMDICSIZE=
（0.8＋MAX（*BINARY_PATH_SIZE*，*SYNONYMDIC_COMMENT_SIZE*）×2）
×（*SYNONYMDIC_NUM*＋*SYNONYMDIC_INVALID_ROWNUM*）

Explanation of the variables

BINARY_PATH_SIZE

Specify the data length of the path name of the directory that stores synonym dictionary files (kilobytes).

SYNONYMDIC_COMMENT_SIZE

Specify the average data length of each synonym dictionary comment (kilobytes).

SYNONYMDIC_NUM

Specify the number of synonym dictionaries that will be created.

SYNONYMDIC_INVALID_ROWNUM

Specify the number of invalid-data rows that arise in the `STATUS_SYNONYM_DICTIONARIES` table.

Estimate how many invalid-data rows will arise in the `STATUS_SYNONYM_DICTIONARIES` table per execution of the `adbreorgsystemdata` command. For details about the interval for executing the `adbreorgsystemdata` command, see 11.17.2 Timing for reorganizing a system table.

Obtain the number of invalid-data rows that will arise in the `STATUS_SYNONYM_DICTIONARIES` table from the following table. At that time, also estimate how many times the command is executed.

Table 5-22: Number of invalid-data rows that arise in the STATUS_SYNONYM_DICTIONARIES table when a command is executed

No.	Command to be executed	Specify the number of invalid-data rows that arise in the STATUS_SYNONYM_DICTIONARIES table#
1	`adbsyndict` command (when executed to update synonym dictionaries)	Number of updated synonym dictionaries
2	`adbsyndict` command (when executed to delete synonym dictionaries)	Number of deleted synonym dictionaries

#

The value in this column indicates the number of invalid-data rows that will arise when the command in the left column is executed once. If the command is executed multiple times, multiply the value by the number of times the command is executed.

6 Preparing Resources

This chapter explains how to prepare disks and estimate various resources.

6.1 Preparing disks

This section explains the points to consider when preparing disks.

6.1.1 Points to consider when preparing disks

To improve HADB's processing performance, it is important to prepare disks so that I/O operations are distributed. For example, if all data is stored on one disk, I/O operations will be concentrated on that disk. If the number of I/O operations exceeds the disk's processing capability at any time, processing performance will be degraded because of the I/O operations that must wait. Therefore, distribute the data across multiple disks in order to reduce the number of I/O operations for each disk.

Especially when a large volume of data must be handled, storing the files listed in the table below across multiple disks can be an effective way to distribute I/O operations.

Table 6-1: Files that might result in improved performance by storing them on multiple disks

No.	File or directory type	Detailed listing of files
1	DB area files	• Master directory DB area file • Dictionary DB area file • System-table DB area file • Work table DB area file • Data DB area file
2	Directories and files used by the HADB server	• Server directory and the files under the server directory • DB directory and the files under the DB directory
3	Temporary work files	Temporary work files created when the following commands are executed: • `adbimport` command • `adbidxrebuild` command • `adbmergechunk` command • `adbunarchivechunk` command • `adbreorgsystemdata` command
4	Input data files	• Input data files used during execution of the `adbimport` command • Input data files (CSV files) used during execution of the `ADB_CSVREAD` function
5	Output data files	Output data files used during execution of the `adbexport` command
6	Archive directory	archive file
7	Synonym dictionary file directory	synonym dictionary file
8	Unload file	Unload file created when the `adbreorgsystemdata` command is executed

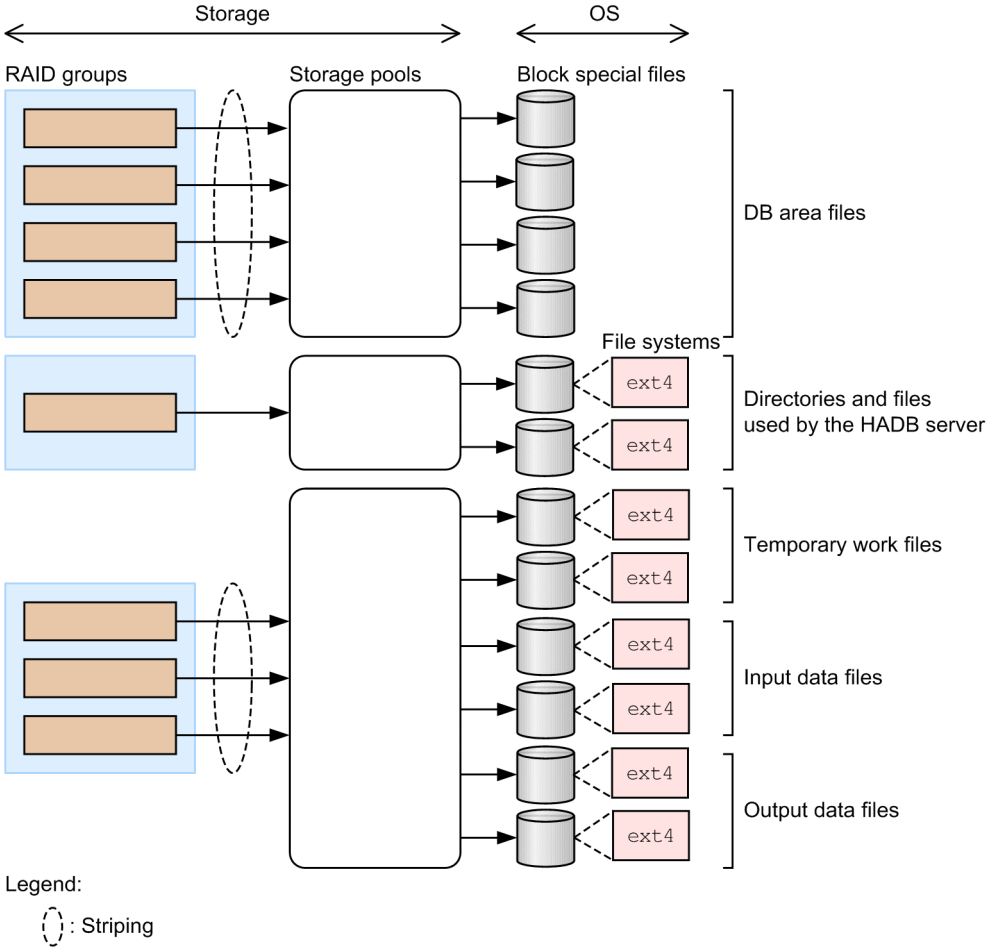
When determining the storage locations for the files shown in the table, it is best to use disks in a RAID array. If there are multiple RAID groups (groups of disks configured in RAID arrays), you can distribute disk I/O operations more efficiently by employing one of the following techniques:

- Use of storage functionality to perform distributed placement of data on virtualized disks
- Striping configuration that uses a logical volume manager (LVM) supported by the OS

■ **Example 1**

The following figure shows an example of using storage functionality to perform distributed placement of data on virtualized disks.

Figure 6-1: Example of using storage functionality to perform distributed placement of data on virtualized disks



This example groups four RAID groups for DB area files together to form one storage pool (virtual disk) and uses a striping method that distributes data to the four RAID groups. Several logical units are created by using virtual disks and are allocated as block special files that make up the DB area files.

When data is accessed in the DB area files, the I/O operations are distributed almost evenly among the four RAID groups. Such a configuration enables you to issue more I/O requests than when there is only one RAID group. This example separates the storage pool for DB area files from other storage pools.

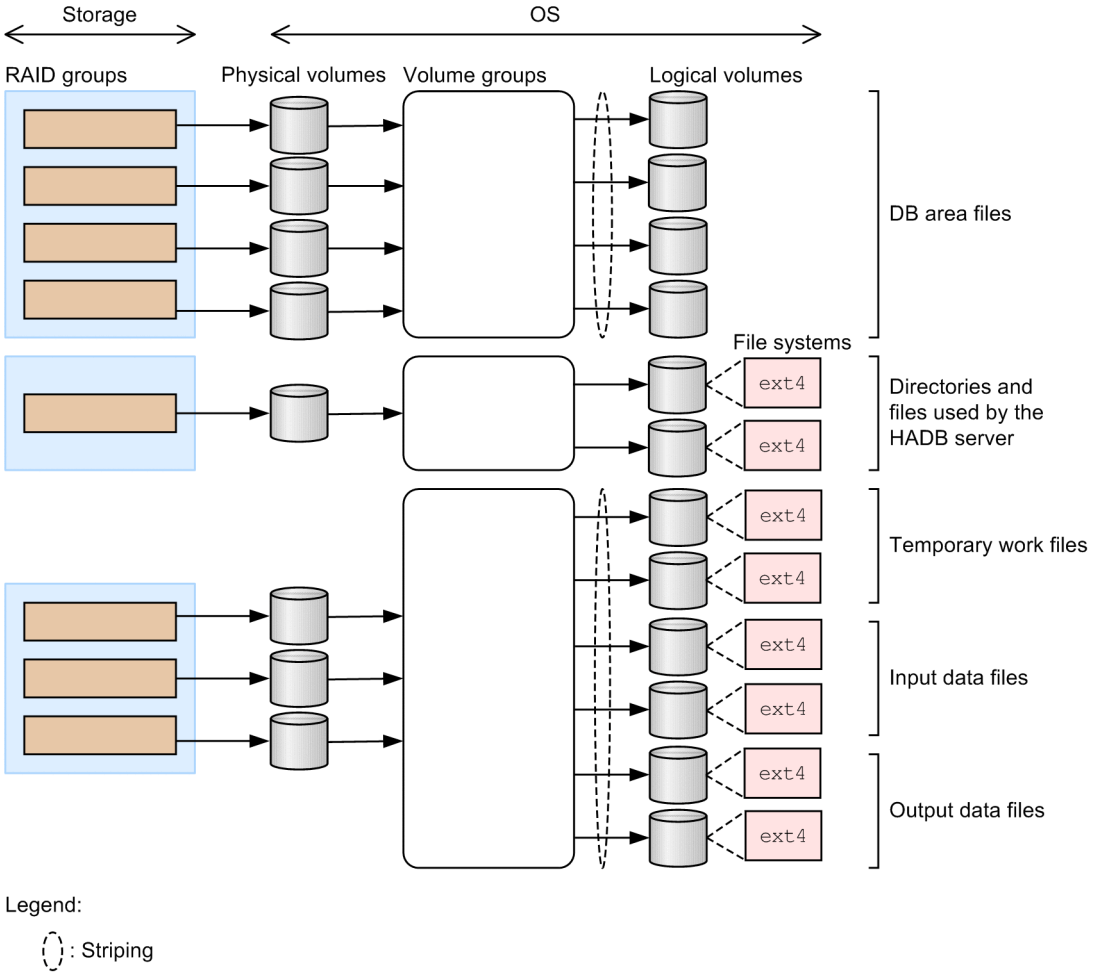
However, if there are not enough available RAID groups, use a single storage pool to apply priority to I/O operations on the DB area files.

For details about how to configure storage, see the documentation for the storage product.

■ **Example 2**

The following figure shows an example of a striping configuration that uses a logical volume manager (LVM) supported by the OS.

Figure 6-2: Example of a striping configuration that uses a logical volume manager (LVM) supported by the OS



This example registers four RAID groups for DB area files into one volume group and creates logical volumes from this volume group with a striping specification. When data is accessed in the DB area files, the I/O operations are distributed almost evenly among the four RAID groups. Such a configuration enables you to issue more I/O requests than when there is only one RAID group. This example separates the volume group for DB area files from other volume groups.

However, if there are not enough available RAID groups, use a single volume group to apply priority to I/O operations on the DB area files.

For details about how to configure LVMs, see the OS documentation.

6.1.2 Points to consider when setting up an LVM

This subsection explains the points to consider when setting up an LVM. For details about how to set up an LVM, see the documentation for the operating system you are using.

- Assign access privileges to the HADB administrator you set up in the OS in 8.2.1 Tasks that must be performed before installation so the HADB administrator can access the logical volume (LV).

  If the logical volume was created by the LVM, its access privilege information might be initialized each time the OS is restarted. Therefore, after restarting the OS, make sure that the access privilege has not been reinitialized.

- To improve the processing performance, use as many disks as possible for striping. Increasing the number of disks distributes the disk storage locations more widely, and therefore reduces the number of I/O operations on each disk, which in turn improves the processing performance.
- When a single DB area is configured across multiple logical volumes (DB area files), accesses to the logical volumes are distributed, resulting in improved processing performance. A guideline for determining the number of logical volumes to configure for a single DB area is 20 to 40 percent of the number of CPU cores in the machine in which the HADB server is installed. For example, if the number of CPU cores in the machine is 40, use 8 to 16 logical volumes to configure a single DB area.
- We recommend that you specify 4,096 kilobytes for the stripe size for LVMs. If you specify a different value, specify one that is at least the DB area page size, does not exceed 16,384 kilobytes, and is an integer multiple of the DB area page size.
- If the data storage locations are sufficiently distributed by storage functions, there is no need to use the LVM to implement a striping configuration just to distribute I/O operations.

  If a striping configuration that stores data across multiple RAID groups cannot be implemented by storage functions, use LVMs to implement striping. For example, if there are 60 disks and the maximum number of disks that can make up one RAID group is 15, create four RAID groups and configure striping for each RAID group.
- To prevent degradation of processing performance due to access contention, we recommend that the size of one logical volume (DB area file) does not exceed 16 terabytes.
- When using the multi-node function or a cold standby configuration, if you are using SCSI reservation for shared disk, specify SCSI reservation at the storage disk (RAID group) level. Areas subject to SCSI reservation must be on a separate disk from other areas. For details about areas subject to SCSI reservation, see the following topics:
  - When using the multi-node function: 16.2.4 Storage configuration
  - When using a cold standby configuration: 17.2.4 Storage configuration

6.1.3 Points to consider regarding power outages

Consider the following points when preparing against a power outage:

- Use the ext3, ext4, or XFS file system.
- Set the disk cache to write-through mode so that its content is not lost during a power outage.

6.2 Estimating the kernel parameters

This section explains how to estimate the values for kernel parameters (Linux operating system parameters).

Before you activate an HADB server, you must set the kernel parameters appropriately. If the kernel parameter values are too small, HADB might not operate correctly. The following table provides guidelines for specifying the kernel parameter values, the files in which to set the kernel parameters, and the applicable parameter names.

Table 6-2: Guidelines and locations for specifying kernel parameter values

No.	Kernel parameter	Specification guidelines	Settings file and parameter name

1	`shmmax`	Specifies, in bytes, the maximum size of a shared memory segment. Specify a value that is equal to or greater than the maximum value determined in 6.3.1 Determining the shared memory requirement.	**Settings file**
			`/etc/sysctl.conf`
			Parameter name
			`kernel.shmmax`

2	`shmmni`	Specifies the maximum number of shared memory segments for the entire system. Specify `2,000` or a larger value.	**Settings file**
			`/etc/sysctl.conf`
			Parameter name
			`kernel.shmmni`

3	`shmall`	Specifies the maximum number of shared memory pages for the entire system. Specify a value for `SHMALL` so that the value determined from the following formula is equal to or greater than the value specified for `SHMMAX`.
		Formula
		shared memory page size × `SHMALL`
			Settings file
			`/etc/sysctl.conf`
			Parameter name
			`kernel.shmall`

4	`threads-max`	Specifies the maximum number of real threads that HADB can use. Specify a value that is equal to or greater than the value determined from the following formula:
		max_users + *rthd_num* + 10 + *multi_num*#
		Explanation of variables:
		max_users:
		Value specified for the `adb_sys_max_users` operand in the server definition
		rthd_num:
		Value specified for the `adb_sys_rthd_num` operand in the server definition
		multi_num:
		Number of host names specified for the `adb_sys_multi_node_info` operand in the server definition × 2 + 3
		#
		Add this value when using the multi-node function.
			Settings file
			`/etc/sysctl.conf`
			Parameter name
			`kernel.threads-max`

5	`aio-max-nr`	Specify a value that is equal to or greater than the value determined from the formula in Figure 6-3: Determining the value of the aio-max-nr kernel parameter. Figure 6-3: Determining the value of the aio-max-nr kernel parameter appears after this table.	**Settings file**
			`/etc/sysctl.conf`
			Parameter name
			`fs.aio-max-nr`

6	`rmem_default`	Specifies the default value for the size of the receive buffer. Specify `33,554,432` or a larger value.	**Settings file**
			`/etc/sysctl.conf`
			Parameter name
			`net.core.rmem_default`

No.	Kernel parameter	Specification guidelines	Settings file and parameter name
7	rmem_max	Specifies the maximum size of the receive buffer. Specify 33,554,432 or a larger value.	**Settings file** /etc/sysctl.conf **Parameter name** net.core.rmem_max
8	wmem_default	Specifies the default value for the size of the send buffer. Specify 33,554,432 or a larger value.	**Settings file** /etc/sysctl.conf **Parameter name** net.core.wmem_default
9	wmem_max	Specifies the maximum size of the send buffer. Specify 33,554,432 or a larger value.	**Settings file** /etc/sysctl.conf **Parameter name** net.core.wmem_max
10	nr_hugepages	When you are applying HugePages to the shared memory, specify the number of pages to be allocated as HugePages. Specify a value that is equal to or greater than the value determined from the following formula: $\uparrow \frac{\text{Required amount of shared memory used by the HADB server}}{\text{page size per page of HugePages}} \uparrow$ For details about the amount of shared memory required by the HADB server, see 6.3.1 Determining the shared memory requirement. You can use the following command to determine the page size per page in HugePages. Pay attention to the unit for the numeric value. **Command to be executed** grep Hugepagesize /proc/meminfo **Execution result example** Hugepagesize: 2048 kB When the server machine's OS is started, HugePages reserves the number of pages of memory specified in this kernel parameter. If you specify for this kernel parameter a number of pages that exceeds the size of memory installed in the server machine on which the HADB server is installed, the OS cannot be started. Therefore, for this kernel parameter, you must specify a page number that does not exceed the size of memory installed in the server machine on which the HADB server is installed.	**Settings file** /etc/sysctl.conf **Parameter name** vm.nr_hugepages
11	hugetlb_shm_group	When you are starting the HADB server by applying HugePages to the shared memory as a non-superuser, you must add the user's group ID to hugetlb_shm_group of the Linux kernel capability. You use the OS's id command to acquire the user's group ID.	**Settings file** /etc/sysctl.conf **Parameter name** vm.hugetlb_shm_group
12	nofile	Specify a value that is equal to or greater than the value determined from the formula in Figure 6-4: Determining the value of the nofile kernel parameter. Figure 6-4: Determining the value of the nofile kernel parameter appears after this table.	**Settings file**# /etc/security/limits.conf **Parameter name** • soft nofile • hard nofile

No.	Kernel parameter	Specification guidelines	Settings file and parameter name
13	file-max	Specify a value that is at least the value specified for nofile.	**Settings file** /etc/sysctl.conf **Parameter name** fs.file-max
14	memlock	Specify the upper limit of the memory lock for the shared memory. Specify the memory size provided in the server machine on which the HADB server is installed.	**Settings file#** /etc/security/limits.conf **Parameter name** • soft memlock • hard memlock
15	nproc	Specify the upper limit of the number of real threads to be used by HADB. Specify a value that is equal to or greater than the value determined from the following formula: *max_users* + *rthd_num* + 10 + *multi_num*# Explanation of variables: *max_users*: Value specified for the adb_sys_max_users operand in the server definition *rthd_num*: Value specified for the adb_sys_rthd_num operand in the server definition *multi_num*: Number of host names specified for the adb_sys_multi_node_info operand in the server definition × 2 + 3 # Add this value when using the multi-node function.	**Settings file#** /etc/security/limits.conf The value to be specified in this file must also be specified for the following file: **If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)** /etc/security/limits.d/90-nproc.conf **If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)** /etc/security/limits.d/20-nproc.conf **Parameter name** • soft nproc • hard nproc
16	/sys/block/*device-name*/queue/nr_requests	Sets up the depth of the kernel's I/O queue. Specify 8192. Change the command-side queue depth queue_depth also. For details about how to set up the queue_depth parameter, see OS- and HBA-related documentation.	**Settings file** /sys/block/*device-name*/queue/nr_requests
17	/sys/block/*device-name*/queue/scheduler	Sets up the I/O scheduler for each device. Specify deadline.	**Settings file** /sys/block/*device-name*/queue/scheduler
18	/sys/block/*device-name*/queue/rq_affinity	If the device is a flash drive, specify 2.	**Settings file** /sys/block/*device-name*/queue/rq_affinity
19	/sys/block/*device-name*/queue/add_random	If the device is a flash drive, specify 0.	**Settings file** /sys/block/*device-name*/queue/add_random

No.	Kernel parameter	Specification guidelines	Settings file and parameter name
20	/sys/block/*device-name*/queue/rotational	If the device is a flash drive, specify 0.	**Settings file** /sys/block/*device-name*/queue/rotational

#

If the HADB server is set to be started automatically from Job Management Partner 1/Automatic Job Management System 3, the kernel parameter does not take effect even if it is set to /etc/security/limits.conf. You must use one of the following methods to specify the kernel parameter:

- Specify the kernel parameter in the environment setting parameters in the job execution environment.
- Specify the kernel parameter in the automated startup script of Job Management Partner 1/Automatic Job Management System 3.

For details, see *Precautions applying when the JP1/AJS3 service starts automatically* in *Notes on using Unix jobs* in the *Job Management Partner 1/Automatic Job Management System 3 System Design (Work Tasks) Guide*.

Use the following formula to determine the value of the aio-max-nr kernel parameter in Table 6-2: Guidelines and locations for specifying kernel parameter values.

Figure 6-3: Determining the value of the aio-max-nr kernel parameter

Formula

$$\left\{(max\_users + rthd\_num + 10 + multi\_num^{\#1}) \times (uthd\_num + 2)\right\}$$
$$+\left\{(write\_rthd + read\_buff\_num) \times 20\right\}$$
$$+\left\{max\_users \times (wrktbl\_clt\_blk\_num \times 4) \times sql\_rthd\_num\right\}$$
$$+ csvread\_aio\_num^{\#2} + auditread\_aio\_num^{\#3}$$

Explanation of variables:

max_users:
Value specified for the `adb_sys_max_users` operand in the server definition

rthd_num:
Value specified for the `adb_sys_rthd_num` operand in the server definition

multi_num:
Number of host names specified for the `adb_sys_multi_node_info` operand in the server definition × 2 + 3

uthd_num:
Value specified for the `adb_sys_uthd_num` operand in the server definition

write_rthd:
Whichever of the following values is largest:
- Value specified for the import option `adb_import_rthd_num` - 1
- ↓(value specified for the index rebuild option `adb_idxrebuild_rthd_num` - 1) ÷ 2↓
- ↓(value specified for the merge chunk option `adb_mergechunk_rthd_num` - 1) ÷ 2
- ↓(value specified for the unarchive chunk option `adb_unarcv_rthd_num` - 1) ÷ 2

read_buff_num:
Substitute 2.

wrktbl_clt_blk_num:
Whichever of the following values is largest:
- Value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition
- Value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition
- Value specified for the export option `adb_export_wrktbl_blk_num`

sql_rthd_num:
Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

csvread_aio_num:
Value determined by the following formula:
(max_users + rthd_num) × 2,048

auditread_aio_num:
Value determined by the following formula:
(*max_users* + *rthd_num*) × 2,048

#1
Add this value when using the multi-node function.

#2
Add this value in the following circumstances:
- When using the `ADB_CSVREAD` function
- When using an unarchivable multi-chunk table

#3
Add this value when using the `ADB_AUDITREAD` function.

Use the following formula to determine the value of the `nofile` kernel parameter in Table 6-2: Guidelines and locations for specifying kernel parameter values.

Figure 6-4: Determining the value of the nofile kernel parameter

Formula

(*max_users* + (*multi_num* + 3)[#1] + *dbarea_file_num* + 49)
+ (*max_users* + *rthd_num* + 10 + *multi_num*[#1]) × 129
+ *export_file_num* + *trc_file_num* + *optlog_file_num* + *csvread_file_num*[#2]
+ *arcvchk_file_num*[#3] + 1 + *syndict_file_num*[#4]
+ *auditread_file_num*[#5] + *convertaudittrailfile_file_num*[#6] + 1

Explanation of variables:

max_users:
Value specified for the `adb_sys_max_users` operand in the server definition

multi_num:
Number of host names specified for the `adb_sys_multi_node_info` operand in the server definition × 2 + 3

dbarea_file_num:
Total number of DB area files that constitute the DB area

rthd_num:
Value specified for the `adb_sys_rthd_num` operand in the server definition

export_file_num:
・Number of output data files specified in the output data path file for the `adbexport` command
・Total number of output data files (if multiple instances of the `adbexport` command are executed concurrently)

trc_file_num:
Substitute 9.

optlog_file_num:
Substitute 3.

csvread_file_num:
Value determined by the following formula:
(*max_users* + *rthd_num*) × 2,048

arcvchk_file_num:
Value determined by the following formula:
(*max_users* + *rthd_num*) × 2,048

syndict_file_num:
Value determined by the following formula:
max_users + 1

auditread_file_num:
Value determined by the following formula:
(*max_users* + *rthd_num*) × 2,048

convertaudittrailfile_file_num:
Substitute 3.

#1: Add this value if using the multi-node function.
#2: Add this value in the following circumstances:
・When using the `ADB_CSVREAD` function
・When using an archivable multi-chunk table
#3: Add this value if using an archivable multi-chunk table.
#4: Add this value if using a synonym dictionary to perform synonym searches.
#5: Add this value if using the `ADB_AUDITREAD` function.
#6: Add this value if using the `adbconvertaudittrailfile` command.

6.3 Estimating the HADB server's memory requirement

This section explains how to estimate the amount of memory required by the HADB server.

You can use the following formula to determine the maximum size of memory to be used by the HADB server. Specify a value that is equal to or greater than the value obtained from the formula for the `adb_sys_memory_limit` server definition operand. For details, see the explanation of the adb_sys_memory_limit operand in 7.2.2 Operands related to performance (set format).

Formula (kilobytes)

Maximum size of memory used by the HADB server =
shared-memory-management-area
+ *global-buffer-page*
+ *process-common-memory*
+ *real-thread-private-memory*[#1] × (*value-of-adb_sys_max_users-server-definition-operand* + 10)
+ (*real-thread-private-memory*[#1] × (2 × *number-of-nodes-used-by-multi-node-function* + 3))[#2]
+ MAX(900,000 , *real-thread-private-memory*[#1]) × *value-of-adb_sys_rthd_num-server-definition-operand*
+ *real-thread-private-memory*[#3]
+ *heap-memory*

#1

Applies to the variables in (4) Determining the real thread private memory requirement under 6.3.1 Determining the shared memory requirement, excluding the following:

- *RTHD_IMPORTSZ*
- *RTHD_IDXRBLDSZ*
- *RTHD_GETCOSTSZ*
- *RTHD_DBSTATUSSZ*
- *RTHD_EXPORTSZ*
- *RTHD_MERCHKSZ*
- *RTHD_ARCCKSZ*
- *RTHD_UNARCCKSZ*
- *RTHD_REORGSZ*
- *RTHD_CLTDEFMNG*
- *RTHD_SYNDICTSZ*
- *RTHD_AUDTRAILSZ*
- *RTHD_CONVERTAUDSZ*

#2

Add this value when you use the multi-node function.

#3

Applies to the following variables in (4) Determining the real thread private memory requirement under 6.3.1 Determining the shared memory requirement:

- *RTHD_IMPORTSZ*
- *RTHD_IDXRBLDSZ*
- *RTHD_GETCOSTSZ*
- *RTHD_DBSTATUSSZ*

- *RTHD_EXPORTSZ*
- *RTHD_MERCHKSZ*
- *RTHD_ARCCKSZ*
- *RTHD_UNARCCKSZ*
- *RTHD_REORGSZ*
- *RTHD_CLTDEFMNG*
- *RTHD_SYNDICTSZ*
- *RTHD_AUDTRAILSZ*
- *RTHD_CONVERTAUDSZ*

The memory used by the HADB server consists of **shared memory** and **process memory**.

■ **Shared memory types**

To determine the amount of shared memory, you must determine the amount for each of the following four types:

- Shared memory management area
- Global buffer page
- Process common memory
- Real thread private memory

■ **Process memory type**

- Heap memory

The HADB server uses these types of memory at the following times:

- During startup of the HADB server
- During normal operation (when an HADB client connects to the HADB server and executes an SQL statement)
- During execution of the `adbinit` command
- During execution of the `adbimport` command
- During execution of the `adbidxrebuild` command
- During execution of the `adbgetcst` command
- During execution of the `adbdbstatus` command
- During execution of the `adbexport` command
- During execution of the `adbstat` command
- During execution of the `adbmodarea` command
- During execution of the `adbmergechunk` command
- During execution of the `adbchgchunkcomment` command
- During execution of the `adbchgchunkstatus` command
- During execution of the `adbarchivechunk` command
- During execution of the `adbunarchivechunk` command
- During execution of the `adbreorgsystemdata` command
- During execution of the `adbclientdefmang` command

- During execution of the `adbsyndict` command
- During execution of the `adbconvertaudittrailfile` command

> **Note**
>
> If you do not specify the `adb_sys_memory_limit` server definition operand, specify the maximum size of process common memory in the `adb_sys_proc_area_max` server definition operand. Also specify the maximum size of real thread private memory per real thread in the `adb_sys_rthd_area_max` server definition operand.

6.3.1 Determining the shared memory requirement

This subsection explains how to determine the amount of shared memory the HADB server will use.

Use the following formula to determine the amount of shared memory required by the HADB server.

Formula

```
shared-memory =
     shared-memory-management-area + global-buffer-page + process-common-memory
     + real-thread-private-memory#1
     × (value-specified-for-adb_sys_max_users-operand-in-server-definition
       + value-specified-for-adb_sys_rthd_num-operand-in-server-definition + 10)
       + real-thread-private-memory#2
```

#1

Applies to the variables in (4) Determining the real thread private memory requirement, excluding the following:

- *RTHD_IMPORTSZ*
- *RTHD_IDXRBLDSZ*
- *RTHD_GETCOSTSZ*
- *RTHD_DBSTATUSSZ*
- *RTHD_EXPORTSZ*
- *RTHD_MERCHKSZ*
- *RTHD_ARCCKSZ*
- *RTHD_UNARCCKSZ*
- *RTHD_REORGSZ*
- *RTHD_CLTDEFMNG*
- *RTHD_SYNDICTSZ*
- *RTHD_AUDTRAILSZ*
- *RTHD_CONVERTAUDSZ*

#2

Applies to the following variables in (4) Determining the real thread private memory requirement:

- *RTHD_IMPORTSZ*
- *RTHD_IDXRBLDSZ*

- *RTHD_GETCOSTSZ*
- *RTHD_DBSTATUSSZ*
- *RTHD_EXPORTSZ*
- *RTHD_MERCHKSZ*
- *RTHD_ARCCKSZ*
- *RTHD_UNARCCKSZ*
- *RTHD_REORGSZ*
- *RTHD_CLTDEFMNG*
- *RTHD_SYNDICTSZ*
- *RTHD_AUDTRAILSZ*
- *RTHD_CONVERTAUDSZ*

The following subsections explain how to determine the requirements for the shared memory management area, global buffer page, process common memory, and real thread private memory.

(1) Determining the shared memory management area requirement

The shared memory management area is used in the cases described below. You must determine the requirement for the shared memory management area required in each of these cases.

- During startup of the HADB server

  See (1) Determining the shared memory management area requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.
- During execution of the `adbinit` command

  See (1) Determining the shared memory management area requirement (for executing the adbinit command) in 6.3.5 Determining the memory requirement for executing the adbinit command.

(2) Determining the global buffer page requirement

A global buffer page is used for starting the HADB server. To determine the global buffer page requirement, see (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(3) Determining the process common memory requirement

Use the formula shown below to determine the process common memory requirement.

Formula (megabytes)

Process common memory
(when the value of the *PROC_STARTSZ* variable does not exceed 4 gigabytes) =

$$\mathrm{MAX}\left\{4{,}000\ ,\ \left(\left\lceil\left(PROC\_STARTSZ + PROC\_EXECSZ\right) \div 1{,}024\right\rceil \times 1.01\right)\right\}$$

Process common memory
(when the value of the *PROC_STARTSZ* variable exceeds 4 gigabytes) =

$$\mathrm{MAX}\left\{4{,}000\ ,\ \left(\left\lceil\left(PROC\_STARTSZ^{\#} + PROC\_EXECSZ\right) \div 1{,}024\right\rceil \times 1.01\right) + \left(\left\lceil PROC\_STAINF \div 1{,}024\right\rceil \times 1.01\right)\right\}$$

#

Add the value obtained by subtracting the value of the *PROC_STAINF* variable from the value of the *PROC_STARTSZ* variable.

Explanation of variables

- *PROC_STARTSZ*

  Process common memory used for starting the HADB server

  Determine the value as explained in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

- *PROC_EXECSZ*

  Process common memory to be used during normal operation

  Determine the value as explained in (1) Determining the process common memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

- *PROC_STAINF*

  Process common memory used to store control information for statistical information

  Determine the value according to (p) Determining the variable PROC_STAINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(4) Determining the real thread private memory requirement

Use the formula shown below to determine the real thread private memory requirement.

Formula (megabytes)

Real thread private memory =

$$\left\lceil\left\{RTHD\_STARTSZ + \mathrm{MAX}\left(RTHD\_EXECSZ\ ,\ RTHD\_IMPORTSZ\ ,\ RTHD\_IDXRBLDSZ\ ,\ RTHD\_GETCOSTSZ\ ,\ RTHD\_DBSTATUSSZ\ ,\ RTHD\_EXPORTSZ\ ,\ RTHD\_MERCHKSZ\ ,\ RTHD\_ARCCKSZ\ ,\ RTHD\_UNARCCKSZ\ ,\ RTHD\_REORGSZ\ ,\ RTHD\_CLTDEFMNG\ ,\ RTHD\_SYNDICTSZ\ ,\ RTHD\_AUDTRAILSZ\ ,\ RTHD\_CONVERTAUDSZ\right)\right\} \div 1{,}024\right\rceil \times 1.01$$

Explanation of variables

- *RTHD_STARTSZ*

  Real thread private memory used for starting the HADB server

  Determine the value as explained in (4) Determining the real thread private memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

- *RTHD_EXECSZ*

  Real thread private memory to be used during normal operation

  Determine the value as explained in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

- *RTHD_IMPORTSZ*

  Real thread private memory to be used when executing the `adbimport` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbimport command) under 6.3.6 Determining the memory requirement for executing the adbimport command.

- *RTHD_IDXRBLDSZ*

  Real thread private memory to be used when executing the `adbidxrebuild` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbidxrebuild command) under 6.3.7 Determining the memory requirement for executing the adbidxrebuild command.

- *RTHD_GETCOSTSZ*

  Real thread private memory to be used when executing the `adbgetcst` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbgetcst command) under 6.3.8 Determining the memory requirement for executing the adbgetcst command.

- *RTHD_DBSTATUSSZ*

  Real thread private memory to be used when executing the `adbdbstatus` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing adbdbstatus command) under 6.3.9 Determining the memory requirement for executing the adbdbstatus command.

- *RTHD_EXPORTSZ*

  Real thread private memory to be used when executing the `adbexport` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (during execution of the adbexport command) under 6.3.10 Determining the memory requirement for executing the adbexport command.

- *RTHD_MERCHKSZ*

  Real thread private memory to be used when executing the `adbmergechunk` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbmergechunk command) under 6.3.13 Determining the memory requirement for executing the adbmergechunk command.

- *RTHD_ARCCKSZ*

  Real thread private memory used for executing the `adbarchivechunk` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbarchivechunk command) under 6.3.16 Determining the memory requirement for executing the adbarchivechunk command.

- *RTHD_UNARCCKSZ*

Real thread private memory used for executing the `adbunarchivechunk` command

Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbunarchivechunk command) under 6.3.17 Determining the memory requirement for executing the adbunarchivechunk command.

- *RTHD_REORGSZ*

  Real thread private memory used for executing the `adbreorgsystemdata` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbreorgsystemdata command) under 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command.

- *RTHD_CLTDEFMNG*

  Real thread private memory used for executing the `adbclientdefmang` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbclientdefmang command) under 6.3.19 Determining the memory requirement for executing the adbclientdefmang command.

- *RTHD_SYNDICTSZ*

  Real thread private memory used for executing the `adbsyndict` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbsyndict command) under 6.3.20 Determining the memory requirement for executing the adbsyndict command.

- *RTHD_AUDTRAILSZ*

  Real thread private memory used when executing the `adbaudittrail` command

  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbaudittrail command) under 6.3.21 Determining the memory requirement for executing the adbaudittrail command.

- *RTHD_CONVERTAUDSZ*

  Real thread private memory used for executing the `adbconvertaudittrailfile` command

  Determine the value as explained in 6.3.22 Determining the memory requirement for executing the adbconvertaudittrailfile command.

6.3.2 Determining the process memory requirement

Use the following formula to determine the process memory (heap memory) requirements.

Formula (megabytes)

$$\textit{process memory (heap memory)} = \left\lceil \left(\textit{HEAP\_STARTSZ} + \textit{HEAP\_EXECSZ} + \textit{HEAP\_IMPORTSZ} + \textit{HEAP\_IDXRBLDSZ} + \textit{HEAP\_EXPORTSZ} + \textit{HEAP\_MERCHKSZ} + \textit{HEAP\_ARCCKSZ} + \textit{HEAP\_UNARCCKSZ} + \textit{HEAP\_REORGSZ} + \textit{HEAP\_CLTDEFMNG} \right) \div 1{,}024 \right\rceil \times 1.01$$

Explanation of variables

- *HEAP_STARTSZ*

  Process memory (heap memory) used when starting the HADB server

  Determine the value as explained in (5) Determining the heap memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

- *HEAP_EXECSZ*

  Process memory (heap memory) used during normal operation

  Determine the value as explained in (3) Determining the heap memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

- *HEAP_IMPORTSZ*

  Process memory (heap memory) used when executing the `adbimport` command

  Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbimport command) under 6.3.6 Determining the memory requirement for executing the adbimport command.

- *HEAP_IDXRBLDSZ*

  Process memory (heap memory) used when executing the `adbidxrebuild` command

  Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbidxrebuild command) under 6.3.7 Determining the memory requirement for executing the adbidxrebuild command.

- *HEAP_EXPORTSZ*

  Process memory (heap memory) used when executing the `adbexport` command

  Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbexport command) under 6.3.10 Determining the memory requirement for executing the adbexport command.

- *HEAP_MERCHKSZ*

  Process memory (heap memory) used when executing the `adbmergechunk` command

  Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbmergechunk command) under 6.3.13 Determining the memory requirement for executing the adbmergechunk command.

- *HEAP_ARCCKSZ*

  Process memory (heap memory) used when executing the `adbarchivechunk` command

  Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbarchivechunk command) under 6.3.16 Determining the memory requirement for executing the adbarchivechunk command.

- *HEAP_UNARCCKSZ*

  Process memory (heap memory) used when executing the `adbunarchivechunk` command

  Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbunarchivechunk command) under 6.3.17 Determining the memory requirement for executing the adbunarchivechunk command.

- *HEAP_REORGSZ*

  Process memory (heap memory) used when executing the `adbreorgsystemdata` command

  Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbreorgsystemdata command) under 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command.

- *HEAP_CLTDEFMNG*

  Process memory (heap memory) used when executing the `adbclientdefmang` command

Determine the value as explained in (3) Determining the heap memory requirement (for executing the adbclientdefmang command) under 6.3.19 Determining the memory requirement for executing the adbclientdefmang command.

6.3.3 Determining the memory requirement for starting the HADB server

When you use the `adbstart` command to start the HADB server, the HADB server uses the types of memories listed below. Therefore, determine the requirement for each type of memory.

■ **Shared memory**

- Shared memory management area
- Global buffer page
- Process common memory
- Real thread private memory

■ **Process memory**

- Heap memory

The following subsections describe the formulas for determining the required amounts of each of these types of memory.

(1) Determining the shared memory management area requirement (for starting the HADB server)

Use the following formula to determine the shared memory management area requirement (*SHMMAN*) for starting the HADB server.

Formula (kilobytes)

SHMMAN =
96,000 + 968 × (*rthd_num* + *max_users* + 10 + *multi_num*)
+ (2,885 × *max_users*) + *SHM_LSGBUF* + *SHM_LSLBUF*

Explanation of variables

rthd_num
Value specified for the `adb_sys_rthd_num` operand in the server definition

max_users
Value specified for the `adb_sys_max_users` operand in the server definition

multi_num
Use the following formula to determine its value.

Formula

multi_num =
Number of host names specified for the adb_sys_multi_node_info operand in the server definition × 2 + 3

SHM_LSGBUF
Shared memory that stores the content displayed by the `adbls -d gbuf` command

Substitute the following value:

Value (kilobytes)

```
SHM_LSGBUF = 337
```

SHM_LSLBUF

Shared memory that stores the content displayed by the `adbls -d lbuf` command

Substitute the following value:

Value (kilobytes)

```
SHM_LSLBUF = 33
```

(2) Determining the global buffer page requirement (for starting the HADB server)

Use the following formula to determine the global buffer page requirement (*SHM_BUFGLOBAL*) for starting the HADB server.

Formula (kilobytes)

$$SHM\_BUFGLOBAL = \uparrow \frac{\left\{ MSTPAGES + WRKPAGES + \sum (BUFPAGES + SCANPAGES) \right\}}{1{,}024} \uparrow$$

Explanation of variables

MSTPAGES: Pages in the master directory DB area

Use the following formula to determine its value.

Formula (bytes)

$$MSTPAGES = RTHNUM \times 8{,}192$$

RTHNUM: See the description of the variable *RTHNUM* in (a) Determining the variable SCI under (3) Determining the process common memory requirement (for starting the HADB server).

WRKPAGES: Pages in the work table DB area

Use the following formula to determine its value.

Formula (bytes)

```
WRKPAGES = (wrk_page_num + 1) × wrk_page_size
```

- *wrk_page_num*
  Value specified for the `adb_dbbuff_wrktbl_glb_blk_num` operand in the server definition
- *wrk_page_size*
  Determine this value in bytes based on the explanation of the page size of the work table DB area in Table 6-3: DB area page size.

Σ(*BUFPAGES* + *SCANPAGES*): Sum total of the *BUFPAGES* and *SCANPAGES* variable values calculated for all specified `adbbuff` operands

This total value also includes the following `adbbuff` operands:

- adbbuff operand with the -o option specified
- adbbuff operand without the -n or -o option specified

The following explains each of these in detail.

adbbuff operand with the -o option specified

When the -o option is specified, DB areas that are not specified in the -n option of the adbbuff operand are classified according to page size and the adbbuff operand is defined automatically. Keep this in mind when determining the value of *BUFPAGES*. An example follows.

Example: When the following four DB areas are not specified in the -n option

■ Target DB areas

- ADBDIC, ADBSTBL, ADBUTBL01, ADBUTBL02 (page size: 4 kilobytes)
- ADBUTBL03 (page size: 32 kilobytes)

■ Server definition specification example

```
adbbuff -g SAMPLEBUF -o -p 100
```

In this case, the four DB areas are classified according to page size and are defined automatically. If any of the options (-p, -a) is specified in the adbbuff operand for which the -o option is specified, the option specification values are also defined automatically.

■ Example of automatic definition

```
adbbuff -g SAMPLEBUF#0000004096 -n ADBDIC, ADBSTBL, ADBUTBL01, ADBUTBL02 -p 100
adbbuff -g SAMPLEBUF#0000032768 -n ADBUTBL03 -p 100
```

Note: The underlined portions are the global buffer names that are added during automatic definition.

adbbuff operand without the -n or -o option specified

When there are DB areas that are not specified in the -n option of the adbbuff operand, and the -o option is not specified either, the DB areas are classified according to page size, as is the case when the -o option is specified, and the adbbuff operand is defined automatically. Keep this in mind when determining the value of *BUFPAGES*. An example follows.

Example: When the following four DB areas are not specified in the -n option, and the -o option is not specified either

■ Target DB areas

- ADBDIC, ADBSTBL, ADBUTBL01, ADBUTBL02 (page size: 4 kilobytes)
- ADBUTBL03 (page size: 32 kilobytes)

■ Server definition specification example

- Not specified.

In this case, the four DB areas are classified according to page size and are defined automatically. The value that is assumed when the -p option is omitted is defined.

■ Example of automatic definition

```
adbbuff -g ##ADBOTHER#0000004096 -n ADBDIC, ADBSTBL, ADBUTBL01, ADBUTBL02
adbbuff -g ##ADBOTHER#0000032768 -n ADBUTBL03
```

Note: The underlined portions are the global buffer names that are defined automatically.

BUFPAGES: Page area

Use the following formula to determine its value.

Formula (bytes)

```
BUFPAGES = (GLBPAGES + OPTTYPES) × page_size
```

GLBPAGES:

Sum total of the values specified for the various options (`-p`, `-a`) of the `adbbuff` operand in the server definition

OPTTYPES:

The number of `adbbuff` operand options (`-p`, `-a`) that are specified in the server definition

If all options are omitted, use 1 for *OPTTYPES*. For example, in the following specification example, *OPTTYPES* is 2, because the `-p` and `-a` options are specified.

Example

```
adbbuff -g SAMPLEBUF -n ADBUTBL01 -p 10 -a 10
```

page_size:

Page size of the DB area specified in the `-n` option of the `adbbuff` operand in the server definition (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size.

SCANPAGES: Table scan pages

Determine the value from the formula shown below. However, if the `-v` option and `-k` option of the `adbbuff` operand in the server definition are omitted, substitute `0` for *SCANPAGES*.

Formula (bytes)

```
SCANPAGES = SCANBLOCKS × 4,194,304
```

SCANBLOCKS:

Number of blocks in the table scan buffer

Use the following formula to determine the value:

$$SCANBLOCKS = \left\downarrow \frac{adbbuff\_opt\_v \times 1{,}048{,}576 - (rthd\_num \times 8{,}841 + 706)}{\frac{1{,}612{,}223{,}349}{page\_size} + 4{,}194{,}369} \right\downarrow$$

adbbuff_opt_v:

Value specified for *size-of-memory-used-by-table-scan-buffer* in the `-v` option of the `adbbuff` operand in the server definition

However, if the `-k` option of the `adbbuff` operand is specified in the server definition, use the following formula to determine the value:

```
value specified for -k option of adbbuff operand × 4
```

page_size:

Page size of the DB area specified in the `-n` option of the `adbbuff` operand in the server definition (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size.

Table 6-3: DB area page size

No.	DB area type	Page size (bytes)
1	Master directory DB area	4,096

No.	DB area type	Page size (bytes)
2	Dictionary DB area	
3	System-table DB area	
4	Work table DB area	↑*work_page_size* ÷ 32↑ × 32 × 1,024 For *work_page_size*, specify the following value: **If the adb_dbarea_wrk_page_size operand is specified in the server definition:** Value of the adb_dbarea_wrk_page_size operand in the server definition **If the adb_dbarea_wrk_page_size operand is not specified in the server definition** Value of the adb_init_wrk_page_size operand for the initialization option of the adbinit command
5	Data DB area	**If the DB area was created using the adbinit command** Use the following formula to determine the page size: *Specification value of the -p option specified in the adbinitdbarea initialization option* × 1,024 **If the DB area was created using the adbmodarea command** Use the following formula to determine the page size: *Specification value of the -p option specified in the adbaddarea DB area addition and modification option* × 1,024

Note

If a database has been built, you can also use a command to check the page size of the DB area as follows:

For the work table DB area:

- Use the adbls -d gbuf command to obtain information about the global buffers.
- Use the adbls -d lbuf command to obtain information about the buffer for the local work table.

For the data DB area:

- Use the adbdbstatus -d summary -c dbarea command to obtain the summary information about the DB area.
- Use the adbls -d gbuf command to obtain information about the global buffers.

rthd_num:

Value specified for the adb_sys_rthd_num operand in the server definition

(3) Determining the process common memory requirement (for starting the HADB server)

Use the following formula to determine the requirement for the process common memory (*PROC_STARTSZ*) for starting the HADB server.

Formula (kilobytes)

```
PROC_STARTSZ =
    SCI + RPCS + DICS + BUFGLOBAL + DBAREAINF + DBSYSINF + RECCTL + DBUPDINFMNG
  + RNGPREREADINF + RNGPREREADHANDLE + MODA + PROC_CNCTSZ + LSINF + DBHTRC
  + LSTCMNINF + LSTRTHDINF + PROC_STAINF + SYNDICTINF + AUDINF + CLMNZ
```

Explanation of variables

- *SCI*
  System management control setting
  See (a) Determining the variable SCI.
- *RPCS*
  Communication service management setting
  See (b) Determining the variable RPCS.
- *DICS*
  Area for retrieval from dictionary tables and system tables
  See (c) Determining the variable DICS.
- *BUFGLOBAL*
  Size of global buffers
  See (d) Determining the variable BUFGLOBAL.
- *DBAREAINF*
  DB area setting
  See (e) Determining the variable DBAREAINF.
- *DBSYSINF*
  System setting for starting HADB server
  See (f) Determining the variable DBSYSINF.
- *RECCTL*
  Recovery control area
  See (g) Determining the variable RECCTL.
- *DBUPDINFMNG*
  Update information control area for DB areas, tables, indexes, and chunks
  See (h) Determining the variable DBUPDINFMNG.
- *RNGPREREADINF*
  Pre-reading of range indexes
  See (i) Determining the variable RNGPREREADINF.
- *RNGPREREADHANDLE*
  Starting pre-reading of range indexes
  See (j) Determining the variable RNGPREREADHANDLE.
- *MODA*
  DB area modification control information
  See (k) Determining the variable MODA.
- *PROC_CNCTSZ*
  Memory used for connecting to the database

See (a) Determining the variable PROC_CNCTSZ in (1) Determining the process common memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

- *LSINF*
  `adbls` command control information
  See (l) Determining the variable LSINF.
- *DBHTRC*
  Storage area used for buffers, disk I/O, tracing communication between HADB servers, and storage areas used for tracing work table state transitions
  See (m) Determining the variable DBHTRC.
- *LSTCMNINF*
  Management information common to work tables
  See (n) Determining the variable LSTCMNINF.
- *LSTRTHDINF*
  Management information for the local work table
  See (o) Determining the variable LSTRTHDINF.
- *PROC_STAINF*
  Control information for statistical information
  See (p) Determining the variable PROC_STAINF.
- *SYNDICTINF*
  Synonym dictionary access information
  See (q) Determining the variable SYNDICTINF.
- *AUDINF*
  Audit trail management information
  See (r) Determining the variable AUDINF.
- *CLMNZ*
  Memory necessary if the updated-row columnizing facility is enabled
  See (s) Determining the variable CLMNZ.

(a) Determining the variable SCI

Use the following formula to determine the value of variable *SCI*.

Formula (kilobytes)

$$SCI = \uparrow (30{,}352 + CON + RTH + UTH + TRN + DEF + MLT^{\#1} + CGR^{\#2} + CMI^{\#3}) \div 1{,}024 \uparrow$$

#1
Add this value when you use the multi-node function.

#2
Add this value when you use the client-group facility.

#3
Add this value when you use centralized management of client definitions.

Explanation of the variables

CON: Connection control setting

Use the following formula to determine its value.

Formula (bytes)

$$CON = 6{,}608 \times max\_users$$

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

RTH: Real thread control setting

Use the following formula to determine its value.

Formula (bytes)

$$RTH = RTHNUM \times 1{,}576$$

RTHNUM: Total number of real threads

Formula

$$RTHNUM = 7 + rthd\_num + CTHNUM + MLTNUM^{\#}$$

#

Add this value when you use the multi-node function.

rthd_num

Value specified for the `adb_sys_rthd_num` operand in the server definition

CTHNUM: Number of connection threads

Formula

$$CTHNUM = 3 + max\_users$$

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

MLTNUM: Number of threads exclusively for the multi-node function

Determine this value when you use the multi-node function.

Formula

$$MLTNUM = multi\_node\_num \times 2 + 3$$

multi_node_num

Number of host names specified in the `adb_sys_multi_node_info` operand in the server definition

UTH: Pseudo real thread control setting

Use the following formula to determine its value.

Formula (bytes)

$$UTH = UTHNUM \times RTHNUM \times 1{,}696$$

UTHNUM: Total number of pseudo-threads

Formula

$$UTHNUM = 2 + uthd\_num$$

uthd_num: Value specified for the `adb_sys_uthd_num` operand in the server definition

RTHNUM

See the variable *RTHNUM* in the description of the variable *RTH* in (a) Determining the variable SCI.

TRN: Transaction management setting

Use the following formula to determine its value.

Formula (bytes)

$$TRN = 96 + (88 \times max\_users)$$

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

DEF: Definition analysis result setting

Use the following formula to determine this value.

Formula (bytes)

$$DEF = 3,701 + 475 \times dbbuff\_defnum$$

dbbuff_defnum: Number of `adbbuff` operands specified in the server definition

MLT: Multi-node control information

Determine this value when you use the multi-node function.

Formula (bytes)

$$MLT = 14,656 + (14,056 + 792 \times RTHNUM) \times multi\_node\_num$$

multi_node_num

Number of host names specified in the `adb_sys_multi_node_info` operand in the server definition

CGR: Client group control information

Determine this value when you use the client-group facility.

Formula (bytes)

$$CGR = 160 + 104 \times cltgrp\_num$$

cltgrp_num

Number of client groups specified in the `adbcltgrp` operand in the server definition

CMI: Information for centralized management of client definitions

Determine this value when you use centralized management of client definitions.

Formula (bytes)

$$CMI = (\uparrow(50,376 + 760 \times (cltmng\_num - 1)) \div 32 \uparrow \times 32 + 64) + 224 \times authid\_num$$

cltmng_num

Total number of `adbclientmang` operands (for client-managing definition) specified in the client management definition file

authid_num

Total number of authorization identifiers specified for the `adbclientmang` operands in the client-managing definition

(b) Determining the variable RPCS

Use the following formula to determine the value of variable *RPCS*.

Formula (kilobytes)

$$RPCS = \left\lceil \left\{ (1{,}024 \times 2) + (262{,}656 \times (CTHNUM + MLTNUM + 2)) + (4 \times ipv4\_num) \right\} \div 1{,}024 \right\rceil + 512$$

Explanation of variables

CTHNUM

See the description of the variable *CTHNUM* in (a) Determining the variable SCI.

MLTNUM

See the description of the variable *MLTNUM* in (a) Determining the variable SCI.

ipv4_num

Number of valid IPv4 addresses of the local node (addresses)

(c) Determining the variable DICS

Use the following formula to determine the value of variable *DICS*.

Formula (kilobytes)

```
DICS = tbldef_cache_size + 84
```

Explanation of variables

tbldef_cache_size: Value specified for the `adb_sql_tbldef_cache_size` operand in the server definition

(d) Determining the variable BUFGLOBAL

Use the following formula to determine the value of variable *BUFGLOBAL*.

Formula (kilobytes)

$$BUFGLOBAL = \lceil (BUFDEF + BUFCTL + BUFINT + BUFPGE + BUFTBL + BUFUPDLISTCTL^{\#}) \div 1{,}024 \rceil \times 1.05$$

#

Add this value when you use the multi-node function.

Explanation of variables

BUFDEF: `adbbuff` operand analysis

Use the following formula to determine its value.

Formula (bytes)

```
BUFDEF = 3,640 + 608 × BUFOPE_NUM
```

BUFOPE_NUM: Number of times the adbbuff operand in the server definition is defined

This total value also includes the following adbbuff operands:

- adbbuff operand with the -o option specified
- adbbuff operand without the -n or -o option specified

The following explains each of these in detail.

adbbuff operand with the -o option specified

When the -o option is specified, DB areas that are not specified in the -n option of the adbbuff operand are classified according to page size, and the adbbuff operand is defined automatically. Keep this in mind when determining the value of *BUFOPE_NUM*. An example follows.

Example: When the following four DB areas are not specified in the -n option

▪ **Target DB areas**

- ADBDIC, ADBSTBL, ADBUTBL01, ADBUTBL02 (page size: 4 kilobytes)
- ADBUTBL03 (page size: 32 kilobytes)

▪ **Server definition specification example**

```
adbbuff -g SAMPLEBUF -o
```

In this case, the four DB areas are classified according to page size and are defined automatically. Therefore, *BUFOPE_NUM* becomes 2,

▪ **Example of automatic definition**

```
adbbuff -g SAMPLEBUF#0000004096 -n ADBDIC, ADBSTBL, ADBUTBL01, ADBUTBL02
adbbuff -g SAMPLEBUF#0000032768 -n ADBUTBL03
```

Note: The underlined portions are the global buffer names that are added during automatic definition.

adbbuff operand without the -n or -o option specified

When there are DB areas that are not specified in the -n option of the adbbuff operand, and the -o option is not specified either, DB areas are classified according to page size, as is the case when the -o option is specified, and the adbbuff operand is defined automatically. Keep this in mind when determining the value of *BUFOPE_NUM*. An example follows.

Example: When a DB area (ADBDIC, ADBSTBL) is not specified in the -n option, and the -o option is not specified either

▪ **Target DB areas**

- ADBDIC, ADBUTBL01, ADBUTBL02 (page size: 4 kilobytes)
- ADBUTBL03 (page size: 32 kilobytes)

▪ **Server definition specification example**

```
adbbuff -g SAMPLEBUF01 -n ADBUTBL01, ADBUTBL02
adbbuff -g SAMPLEBUF02 -n ADBUTBL03
```

In this case, ADBDIC and ADBSTBL, which have not been specified, are defined automatically. Therefore, *BUFOPE_NUM* becomes 4, which is for ADBDIC plus the two that are specified.

▪ **Example of automatic definition**

```
adbbuff -g ##ADBOTHER#0000004096 -n ADBDIC, ADBSTBL
```

Note: The underlined portions are the global buffer names that are defined automatically.

BUFCTL: Global buffer control area

Use the following formula to determine its value.

Formula (bytes)

$$BUFCTL = 568 \times \Sigma OPTTYPES + 88 \times DBAREA\_NUM + 256 \times BUFOPT\_V\_NUM + 18,896 + 240 \times RTHNUM$$

Σ*OPTTYPES*: Total number of options (`-p`, `-a`) specified in each `adbbuff` operand

This total value also includes the following `adbbuff` operands:

- `adbbuff` operand with the `-o` option specified
- `adbbuff` operand without the `-n` or `-o` option specified

For details, see the description of the variable Σ(*BUFPAGES* + *SCANPAGES*) in (2) Determining the global buffer page requirement (for starting the HADB server).

An example of how to determine the number of option specifications follows.

Example

When the `adbbuff` operand is defined as shown in the following, the total number of specified options is 5 (1 + 2 + 1 + 1), including the automatically defined `ADBDIC` and `ADBSTBL`.

▪ Server definition specification example

```
adbbuff -g SAMPLEBUF1 -n ADBUTBL01 -p 100
adbbuff -g SAMPLEBUF2 -n ADBUTBL02 -p 100 -a 100
adbbuff -g SAMPLEBUF3 -n ADBUTBL03#
```

#: The `-p` option is assumed.

▪ Example of automatic definition

```
adbbuff -g ##ADBOTHER#0000004096 -n ADBDIC, ADBSTBL#
```

Note: The underlined portions are the global buffer names that are defined automatically.

#: The `-p` option is assumed.

DBAREA_NUM:

Total number of DB areas

BUFOPT_V_NUM:

Number of defined `adbbuff` operands with the `-v` option or `-k` option specified

RTHNUM:

See the description of the variable *RTHNUM* in (a) Determining the variable SCI.

BUFINT: Temporary area for multiplexed initialization

Use the following formula to determine its value.

Formula (bytes)

$$BUFINT = 480 + 80 \times \Sigma BUFWRK\_NUM + 8 \times rthd\_num + INITQUE$$

rthd_num:

Value specified for the `adb_sys_rthd_num` operand in the server definition

INITQUE: Initialization processing queue

Use the following formula to determine its value.

Formula (bytes)

```
INITQUE = 80 × RTHNUM × RTHNUM + 632 × RTHNUM + 960 + 64 × ΣBUFWRK_NUM
```

RTHNUM: See the description of the variable *RTHNUM* in (a) Determining the variable SCI.

Σ*BUFWRK_NUM*: Sum total of the *BUFWRK_NUM* variables calculated for each `adbbuff` operand

This total value also includes the following `adbbuff` operands:

- `adbbuff` operand with the `-o` option specified
- `adbbuff` operand without the `-n` or `-o` option specified

For details, see the description of the variable Σ(*BUFPAGES* + *SCANPAGES*) in (2) Determining the global buffer page requirement (for starting the HADB server).

BUFWRK_NUM: Use the following formula to determine its value.

Formula (count)

$$BUFWRK\_NUM = 1 + OPTTYPES + \left\uparrow \frac{adbbuff\_opt\_p}{500,000} \right\uparrow + \left\uparrow \frac{adbbuff\_opt\_a}{500,000} \right\uparrow + BUFOPT\_V + SCAN\_INIT$$

OPTTYPES:

See the description of the variable *OPTTYPES* in (2) Determining the global buffer page requirement (for starting the HADB server).

adbbuff_opt_p:

Value specified for the `-p` option of the `adbbuff` operand

adbbuff_opt_a:

Value specified for the `-a` option of the `adbbuff` operand

BUFOPT_V: Whether the `-v` option or `-k` option of the `adbbuff` operand is specified

Substitute `1` if the `-v` option or `-k` option is specified; otherwise substitute `0`.

SCAN_INIT: Area for multiplexed initialization of the table scan buffer

If the `-v` option or `-k` option of the `adbbuff` operand is specified, substitute the value specified for the `adb_sys_rthd_num` operand in the server definition. Substitute `0` if neither the `-v` option nor the `-k` option is specified.

BUFPGE: Page management area

Use the following formula to determine its value.

Formula (bytes)

```
BUFPGE = ΣTYPEBUF + MSTBUF + WRKBUF
```

Σ*TYPEBUF*: Sum total of the *TYPEBUF* variables determined for the options (`-p`, `-a`) specified in each `adbbuff` operand

This total value also includes the following `adbbuff` operands:

- `adbbuff` operand with the `-o` option specified
- `adbbuff` operand without the `-n` or `-o` option specified

For details, see the description of the variable Σ(*BUFPAGES* + *SCANPAGES*) in (2) Determining the global buffer page requirement (for starting the HADB server).

TYPEBUF: Use the following formula to determine its value.

Formula (bytes)

```
TYPEBUF = GCLBUF(page_num) + BUFMEM(page_num)
```

page_num:

Value specified for any of the options (`-p`, `-a`) specified in the `adbbuff` operand in the server definition

GCLBUF: Use the following formula to determine its value.

Formula (bytes)

```
GCLBUF(X) = 192 × RTHNUM + 192 + X × (176 + BUFLOG)
```

RTHNUM:

See the description of the variable *RTHNUM* in (a) Determining the variable SCI.

X:

Number of pages to be targeted

BUFLOG: Log area

Use the following formula to determine its value.

Formula (bytes)

```
BUFLOG = ↓(((16 + ((RTHNUM - 6) ÷ 8 + 1)) + 15) ÷ 16)↓ × 16
```

BUFMEM: Use the following formula to determine its value.

Formula (bytes)

```
BUFMEM(X) =
    480 × RTHNUM × RTHNUM + (584 + 24 × UTHNUM) × RTHNUM
    + (352 + 16 × RTHNUM) × HASHTBL_NUM + 80 × X + 24 × UTHNUM + 368
```

*RTHNUM***:**

See the description of the variable *RTHNUM* in (a) Determining the variable SCI.

UTHNUM:

See the description of the variable *UTHNUM* in (a) Determining the variable SCI.

X:

Number of pages to be targeted

HASHTBL_NUM: Use the following formula to determine its value.

Formula

```
HASHTBL_NUM = X ÷ 4
```

MSTBUF: Use the following formula to determine its value.

Formula (bytes)

```
MSTBUF = RTHNUM × 2
```

RTHNUM:

See the description of the variable *RTHNUM* in (a) Determining the variable SCI.

WRKBUF:

Value specified for the `adb_dbbuff_wrktbl_glb_blk_num` operand in the server definition

BUFTBL: Table scan buffer

Use the following formula to determine its value.

Formula (bytes)

$$BUFTBL = \Sigma SCANBUF$$

$\Sigma SCANBUF$: Number of table scan buffers for batch reading specified in each `adbbuff` operand

This is the sum total of the *SCANBUF* variables calculated for each `-v` or `-k` option specified for the `adbbuff` operand.

SCANBUF: Use the following formula to determine its value.

Formula (bytes)

$$SCANBUF = SCANBLOCKS \times \left(64 + \frac{4{,}194{,}304}{page\_size} \times 384\right) + rthd\_num \times 8{,}896 + 320$$

For `adbbuff` operands for which neither the `-v` option nor `-k` option is specified, assume 0 for *SCANBUF*.

- *SCANBLOCKS*

  Number of blocks in the table scan buffer

  Determine the value according to the description of the variable *SCANBLOCKS* in (2) Determining the global buffer page requirement (for starting the HADB server).

- *page_size*

  Page size of the DB area specified in the `-n` option of an `adbbuff` operand specified with the `-v` option or `-k` option (bytes)

  Determine the value as explained in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server).

- *rthd_num*

  Value specified for the `adb_sys_rthd_num` operand in the server definition

BUFUPDLISTCTL

Area for managing the update list among nodes

Substitute the following value.

Value (bytes)

$$BUFUPDLISTCTL = 576$$

(e) Determining the variable DBAREAINF

Use the following formula to determine the variable *DBAREAINF*:

Formula (kilobytes)

$$DBAREAINF = 10{,}064 + 64 \times dbarea\_num + 11 \times dbarea\_file\_num + (table\_num + index\_num) \times 1{,}876 + \left\{\left\lceil \frac{32 + \left(16 + dbarea\_file\_num \times \left\lceil \frac{SEGSIZE}{16} \right\rceil \times 8\right) \times 600}{1{,}024} \right\rceil\right\} + (max\_users + rthd\_num) \times 11 + 3 \times dbarea\_file\_num \times column\_table\_num$$

Explanation of variables

dbarea_num

Number of all DB areas

dbarea_file_num

Number of all DB area files

table_num

Total number of user-defined base tables

column_table_num

Total number of user-defined base tables that are column store tables

index_num

Total number of user-defined indexes

SEGSIZE

Segment size in the DB area in which DB area files are defined (pages)

Use the following formula to determine the segment size:

```
SEGSIZE = 4,194,304 ÷ page_size
```

- *page_size*

  Page size in the DB area in which DB area files are defined (bytes)

  Determine the value as explained in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server).

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

rthd_num

Value specified for the `adb_sys_rthd_num` operand in the server definition

(f) Determining the variable DBSYSINF

For the variable *DBSYSINF*, substitute the following value.

Value (kilobytes)

DBSYSINF = 1,308

(g) Determining the variable RECCTL

Use the following formula to determine the value of variable *RECCTL*.

Formula (kilobytes)

RECCTL = 2,177 + (65 + 64 × *log_usrbuf_num*) × (*log_usrfile_num*) + *max_users*

Explanation of variables

log_usrbuf_num: Value specified for the `adb_log_usrbuf_num` operand in the server definition

log_usrfile_num: Value specified for the `adb_log_usrfile_num` operand in the server definition

max_users: Value specified for the `adb_sys_max_users` operand in the server definition

(h) Determining the variable DBUPDINFMNG

Substitute the following value for the variable *DBUPDINFMNG*.

Note that this value changes depending on whether the multi-node function is used.

Value (kilobytes)

- If the multi-node function is not used
  DBUPDINFMNG = 89
- If the multi-node function is used
  DBUPDINFMNG = 369

(i) Determining the variable RNGPREREADINF

Use the formulas shown below to determine the variable *RNGPREREADINF*. The formula to be used depends on the specification of the `adb_sql_rngidx_preread` operand in the server definition.

If `NO` is specified for the `adb_sql_rngidx_preread` operand, the variable *RNGPREREADINF* becomes 0.

- When `ALL` is specified in the `adb_sql_rngidx_preread` operand

  Formula (kilobytes)

  $$RNGPREREADINF = \left\lceil \frac{\left\{484 + (uthd\_num + 2) \times 8\right\}}{1{,}024} \right\rceil$$

  Explanation of variables

  uthd_num: Value specified for the `adb_sys_uthd_num` operand in the server definition

- When range index names are specified in the `adb_sql_rngidx_preread` operand

  Formula (kilobytes)

  $$RNGPREREADINF = \left\lceil \frac{\left\{484 + 224 \times preread\_rng\_num + (uthd\_num + 2) \times 8\right\}}{1{,}024} \right\rceil$$

  Explanation of variable

  preread_rng_num: Number of range index names specified in the `adb_sql_rngidx_preread` operand

  uthd_num: Value specified for the `adb_sys_uthd_num` operand in the server definition

(j) Determining the variable RNGPREREADHANDLE

Substitute the following value for the variable *RNGPREREADHANDLE*.

Value (kilobytes)

RNGPREREADHANDLE = 1

(k) Determining the variable MODA

Use the following formula to determine the value of variable *MODA*.

Formula (kilobytes)

```
MODA = 9,216 + ↑wrk_page_size ÷ 1,024↑
```

Explanation of variables

wrk_page_size

Page size of the work table DB area (bytes)

Determine the value as explained in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server).

(l) Determining the variable LSINF

Substitute the following value for the variable *LSINF*.

Value (kilobytes)

LSINF = 1

(m) Determining the variable DBHTRC

Use the following formula to determine the value of variable *DBHTRC*.

Formula (kilobytes)

DBHTRC = 4,570 × *RTHNUM*

Explanation of variables

RTHNUM

See the description of the variable *RTHNUM* in (a) Determining the variable SCI.

(n) Determining the variable LSTCMNINF

Use the following formula to determine the value of variable *LSTCMNINF*.

Formula (kilobytes)

LSTCMNINF ＝ ↑（4,128＋8,200 × *max_users*）÷1,024↑

Explanation of variables

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

(o) Determining the variable LSTRTHDINF

Use the following formula to determine the value of variable *LSTRTHDINF*.

Formula (kilobytes)

LSTRTHDINF ＝ ↑（8,200 × *RTHNUM*）÷1,024↑

Explanation of variables

RTHNUM

See the description of the variable *RTHNUM* in (a) Determining the variable SCI.

(p) Determining the variable PROC_STAINF

Use the following formula to determine the value of variable *PROC_STAINF*.

Formula (kilobytes)

```
PROC_STAINF= ↑max_users× (443,472×128+16) ÷1,024↑
```

Explanation of variables

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

(q) Determining the variable SYNDICTINF

Substitute the following value for the variable *SYNDICTINF*.

Value (kilobytes)

```
SYNDICTINF= 76,951
```

(r) Determining the variable AUDINF

Use the following formula to determine the value of variable *AUDINF*. Add this value when the audit trail facility is enabled. Disabling the audit trail facility releases the value determined for variable *AUDINF*.

Formula (kilobytes)

```
AUDINF = 1 + PROC_AUDINFSZ
```

Explanation of variables

PROC_AUDINFSZ: Audit trail information

Use the following formula to determine the value:

Formula (kilobytes)

```
PROC_AUDINFSZ = 2 + (↑sql_text_size÷1,024↑)#1
              + (↑PARAM_INFO_SIZE÷1,024↑)#2
              + OBJECT_INFO_SIZE#3
              + system_info_size
```

#1

Add this value when executing an SQL statement.

#2

Add this value when executing an SQL statement that includes a dynamic parameter.

#3

Add this value when executing an SQL statement or a command.

sql_text_size

SQL text length (bytes)

PARAM_INFO_SIZE

Parameter storage area (bytes)

See the description of the variable *PARAM_INFO_SIZE* in (c) Determining the variable RTHD_EXESQLSZ under (2) Determining the real thread private memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

OBJECT_INFO_SIZE

Object information (kilobytes)

Use the following formula to determine the value:

```
OBJECT_INFO_SIZE= 1 x obj_num×user_num
```

obj_num

Number of objects specified in the SQL statement or command

If `ALL TABLES` is specified in a `GRANT` statement or `REVOKE` statement, this value is the number of tables to which `ALL TABLES` applies.

user_num

Number of privilege grantees specified in the `GRANT` statement or `REVOKE` statement

If the SQL statement or command does not specify any privilege grantees, a value of `1` is used.

system_info_size

System information (kilobytes)

Use the following formula to determine the memory requirement when starting the HADB server:

```
system_info_size = ↑size-of-server-definition-file# ÷ 1,024↑
```

#

Specify the size of the server definition file in bytes.

Use the following formula to determine the memory requirement for situations other than HADB server startup:

- When executing the `adbclientdefmang` command

  ```
  system_info_size = 2 x ↑size-of-client-management-definition-file# ÷ 1,024↑
  ```

  #

  Specify the size of the client management definition file in bytes.

- Other than when executing the `adbclientdefmang` command

  ```
  system_info_size = 0
  ```

(s) Determining the variable CLMNZ

Substitute the following value for the variable *CLMNZ*.

Value (kilobytes)

```
CLMNZ = 5,120 + 1
```

(4) Determining the real thread private memory requirement (for starting the HADB server)

Use the following formula to determine the real thread private memory requirement (*RTHD_STARTSZ*) for starting the HADB server.

Formula (kilobytes)

$$RTHD\_STARTSZ = RTI + DICL + DICL2^{\#1} + RTHD\_STAINF + MLTMNG^{\#2} + TRCMON + AUDTHDINF^{\#3}$$

#1

Add this value when you upgrade the HADB server version.

#2

Add this value when you use the multi-node function.

#3

Add this value when the audit trail facility is enabled.

Explanation of variables

RTI

Real thread management information

Use the following formula to determine its value.

Formula (kilobytes)

$$RTI = \uparrow(AIO + DBI) \div 1{,}024\uparrow$$

AIO

Asynchronous I/O management information

Use the following formula to determine its value.

Formula (bytes)

$$AIO = 32 \times UTHNUM$$

UTHNUM

For details, see the description of the variable *UTHNUM* in (a) Determining the variable SCI under (3) Determining the process common memory requirement (for starting the HADB server).

DBI

Use the following formula to determine its value.

Formula (bytes)

$$DBI = 336 \times UTHNUM$$

DICL

Substitute the following value.

Value (kilobytes)

$$DICL = 46$$

DICL2

Substitute the following value.

Value (kilobytes)

```
DICL2＝ 7,790
```

RTHD_STAINF

Control information for statistical information

Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_STAINF＝
    （394＋0.6×tables×6） ×（2＋max_sql_concurrent_exec_num）
```

tables

Sum of the numbers of tables specified in all SQL statements that are executed in one transaction

max_sql_concurrent_exec_num

Maximum number of SQL statements to be executed at the same time during a transaction

MLTMNG

Multi-node function management information

Use the following formula to determine its value.

Formula (kilobytes)

```
MLTMNG＝
  ↑(2,120＋↑(1,216×authid_num)÷4,096↑×4,096
   ＋(8,536＋760×UTHNUM)×multi_node_num)÷1,024↑
```

authid_num

Total number of authorization identifiers specified for the `adbclientmang` operands in the client-managing definition

multi_node_num

Number of host names specified in the `adb_sys_multi_node_info` operand in the server definition

TRCMON

Monitor out thread management information for SQL tracing

Use the following formula to determine its value.

Formula (kilobytes)

```
TRCMON＝ 98,318＋(RTHD_EXPSQLTRCSZ×3)#
```

#

Memory allocated when the SQL trace buffer is extended

RTHD_EXPSQLTRCSZ

Real thread private memory used when the SQL trace buffer is extended

Determine the value according to (i) Determining the variable RTHD_EXPSQLTRCSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

AUDTHDINF

Audit trail management information

Use the following formula to determine the value. Disabling the audit trail facility releases the value determined for variable *AUDTHDINF*.

Formula (kilobytes)

```
AUDTHDINF = ↑(272,631,408 + CTHNUM x 260,000) ÷ 1,024↑
```

CTHNUM

For details, see the description of the variable *CTHNUM* in (a) Determining the variable SCI under (3) Determining the process common memory requirement (for starting the HADB server).

(5) Determining the heap memory requirement (for starting the HADB server)

Use the following formula to determine the heap memory requirement (*HEAP_STARTSZ*) for starting the HADB server.

Formula (kilobytes)

HEAP_STARTSZ = 161 x *UTHNUM* x *RTHNUM*

Explanation of variables

UTHNUM

For details, see the description of the variable *UTHNUM* in (a) Determining the variable SCI under (3) Determining the process common memory requirement (for starting the HADB server).

RTHNUM

For details, see the description of the variable *RTHNUM* in (a) Determining the variable SCI under (3) Determining the process common memory requirement (for starting the HADB server).

6.3.4 Determining the memory requirement during normal operation

During normal operation following HADB server startup (when an HADB client connects to the HADB server and executes an SQL statement), the HADB server uses the types of memory listed below. Determine the requirement for each type of memory.

■ Shared memory

- Process common memory (*PROC_EXECSZ*)
- Real thread private memory (*RTHD_EXECSZ*)

■ Process memory

- Heap memory (*HEAP_EXECSZ*)

The following subsections describe the formula for determining each of these memory requirements.

(1) Determining the process common memory requirement (during normal operation)

Use the following formula to determine the process common memory requirement for normal operation (*PROC_EXECSZ*).

Formula (kilobytes)

```
PROC_EXECSZ=
    PROC_CNCTSZ+PROC_TOTALSQLSSZ+PROC_EXECSQLSZ
  +PROC_BUFWORK_CTL+PROC_DEFSQLSZ+PROC_IMPORTSZ
  +PROC_IDXRBLDSZ+PROC_GETCOSTSZ+PROC_DBSTATUSSZ
  +PROC_MODASZ+PROC_EXPORTSZ+PROC_MERGCSZ
  +PROC_UPDSZ+PROC_CHGCCMSZ+PROC_CHGCSTSZ
  +PROC_UPDLISTSZ#+PROC_ARCCKSZ+PROC_UNARCCKSZ
  +PROC_REORGSZ+PROC_SYNDICTSZ+PROC_AUDTRAILSZ
```

#

Add this value when you use the multi-node function.

Explanation of variables

- *PROC_CNCTSZ*

  Process common memory used for connecting to the database

  See (a) Determining the variable PROC_CNCTSZ.

- *PROC_TOTALSQLSSZ*

  Process common memory used for preprocessing SQL statements

  See (b) Determining the variable PROC_TOTALSQLSSZ.

- *PROC_EXECSQLSZ*

  Process common memory used for executing SQL statements

  See (c) Determining the variable PROC_EXECSQLSZ.

- *PROC_BUFWORK_CTL*

  Process common memory used for accessing work tables

  See (d) Determining the variable PROC_BUFWORK_CTL.

- *PROC_DEFSQLSZ*

  Process common memory used for executing definition SQL statements

  See (e) Determining the variable PROC_DEFSQLSZ.

- *PROC_IMPORTSZ*

  Process common memory used for executing the `adbimport` command

  Determine the value as explained in (1) Determining the process common memory requirement (for executing the adbimport command) under 6.3.6 Determining the memory requirement for executing the adbimport command.

- *PROC_IDXRBLDSZ*

  Process common memory used for executing the `adbidxrebuild` command

  Determine the value as explained in (1) Determining the process common memory requirement (for executing the adbidxrebuild command) under 6.3.7 Determining the memory requirement for executing the adbidxrebuild command.

- *PROC_GETCOSTSZ*

  Process common memory used for executing the `adbgetcst` command

  Determine the value as explained in (1) Determining the process common memory requirement (for executing the adbgetcst command) under 6.3.8 Determining the memory requirement for executing the adbgetcst command.

- *PROC_DBSTATUSSZ*

  Process common memory used for executing the `adbdbstatus` command

Determine the value as explained in (1) Determining the process common memory requirement (for executing the adbdbstatus command) under 6.3.9 Determining the memory requirement for executing the adbdbstatus command.

- *PROC_MODASZ*

  Process common memory used for executing the `adbmodarea` command

  See (1) Determining the process common memory requirement (for executing the adbmodarea command) in 6.3.12 Determining the memory requirement for executing the adbmodarea command.

- *PROC_EXPORTSZ*

  Process common memory used for executing the `adbexport` command

  Determine the value as explained in 6.3.10 Determining the memory requirement for executing the adbexport command.

- *PROC_MERGCSZ*

  Process common memory used for executing the `adbmergechunk` command

  Determine the value as explained in (1) Determining the process common memory requirement (for executing the adbmergechunk command) under 6.3.13 Determining the memory requirement for executing the adbmergechunk command.

- *PROC_UPDSZ*

  Process common memory used when a definition SQL statement or update SQL statement is executed for the first time after a connection to the database is established

  Determine the value as explained in (f) Determining the variable PROC_UPDSZ.

- *PROC_CHGCCMSZ*

  Process common memory used during execution of the `adbchgchunkcomment` command

  Determine the value as explained in 6.3.14 Determining the memory requirement for executing the adbchgchunkcomment command.

- *PROC_CHGCSTSZ*

  Process common memory used during execution of the `adbchgchunkstatus` command

  Determine the value as explained in 6.3.15 Determining the memory requirement for executing the adbchgchunkstatus command.

- *PROC_UPDLISTSZ*

  Process common memory used for using the multi-node function

  See (g) Determining the variable PROC_UPDLISTSZ.

- *PROC_ARCCKSZ*

  Process common memory used for executing the `adbarchivechunk` command

  See (1) Determining the process common memory requirement (for executing the adbarchivechunk command) in 6.3.16 Determining the memory requirement for executing the adbarchivechunk command.

- *PROC_UNARCCKSZ*

  Process common memory used for executing the `adbunarchivechunk` command

  See (1) Determining the process common memory requirement (for executing the adbunarchivechunk command) in 6.3.17 Determining the memory requirement for executing the adbunarchivechunk command.

- *PROC_REORGSZ*

  Process common memory used for executing the `adbreorgsystemdata` command

  See (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) in 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command.

- *PROC_SYNDICTSZ*

Process common memory used for executing the `adbsyndict` command

See (1) Determining the process common memory requirement (for executing the adbsyndict command) in 6.3.20 Determining the memory requirement for executing the adbsyndict command.

- *PROC_AUDTRAILSZ*

  Process common memory used during execution of the `adbaudittrail` command

  See (1) Determining the process common memory requirement (for executing the adbaudittrail command) in 6.3.21 Determining the memory requirement for executing the adbaudittrail command.

(a) Determining the variable PROC_CNCTSZ

The variable *PROC_CNCTSZ* is required when connecting to databases. Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_CNCTSZ=
  BUFCNCT+SQLWQUE+SQLACTB+RTHDLOG+2
```

Explanation of variables

BUFCNCT: Connection buffer

Use the following formula to determine its value.

Formula (kilobytes)

```
BUFCNCT =
       ↑(56 + 8,216 × rthd_num) × max_users ÷ 1,024↑
```

rthd_num

Use the following formula to determine the value:

```
value-specified-for-adb_sys_rthd_num-operand-in-server-definition + 1
```

However, use `0` for the variable *rthd_num* if there is no `adbbuff` operand in the server definition for which the `-v` option or `-k` option is specified.

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

SQLWQUE: SQL statement execution queue

Use the following formula to determine its value.

Formula (kilobytes)

```
SQLWQUE = ↑8 × max_users ÷ 1,024↑
```

max_users: Value specified for the `adb_sys_max_users` operand in the server definition

SQLACTB: SQL statement table access information

Use the following formula to determine its value.

Formula (kilobytes)

```
SQLACTB=
  ↑56 × max_table_num ÷ 1,024↑
```

max_table_num: Maximum number of tables accessed by each SQL statement executed in the target connection

RTHDLOG

Log management area for processing real threads

Use the following formula to determine its value.

Formula (kilobytes)

$$RTHDLOG = \uparrow(48 + 8 \times rthd\_num) \div 1{,}024\uparrow$$

rthd_num

Value specified for the `adb_sys_rthd_num` operand in the server definition

(b) Determining the variable PROC_TOTALSQLSSZ

The variable *PROC_TOTALSQLSSZ* is required when preprocessing SQL statements. Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_TOTALSQLSSZ = MAXSQLWRK + TOTALSQLSCT + PREDICSZ + PROC_AUDINFSZ
```

Explanation of the variables

MAXSQLWRK

Work area used for a transaction during preprocessing of SQL statements

Use the following formula to determine its value.

Formula (kilobytes)

```
MAXSQLWRK = (max_sql(SQLPSZ) + 20) × max_users
```

max_sql()

Substitute the maximum value obtained when calculating *SQLPSZ*

max_users

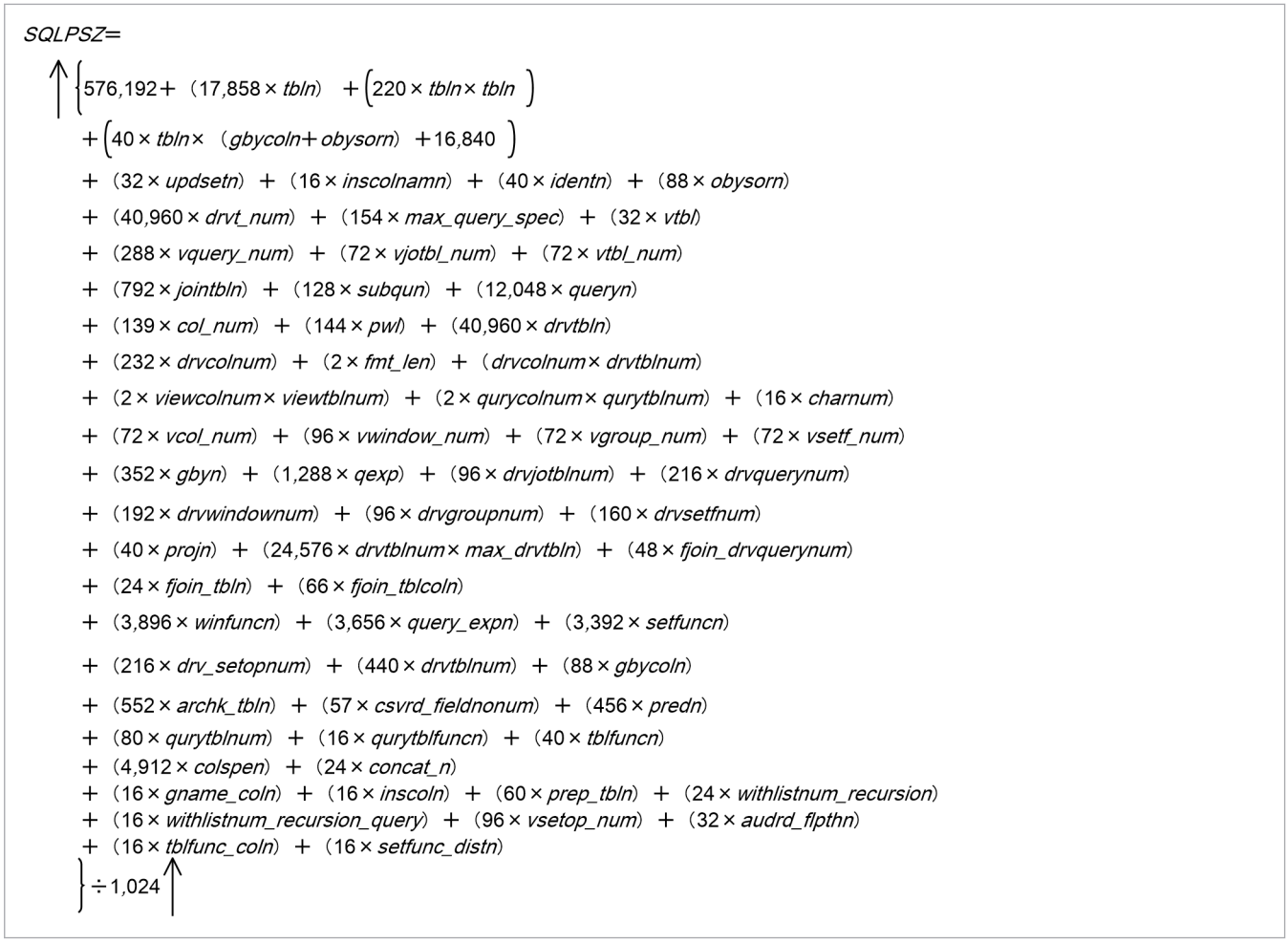
Value specified for the `adb_sys_max_users` operand in the server definition

SQLPSZ

Work areas used when preprocessing SQL statements

Use the following formula to determine its value.

Formula (kilobytes)



tbln

Number of tables specified in the `FROM` clause

gbycoln

Number of columns specified in the `GROUP BY` clause

obysorn

Number of sort keys specified in the `ORDER BY` clause

updsetn

Number of `SET` clauses in the `UPDATE` statement

inscolnamn

Number of columns to be inserted by the `INSERT` statement

identn

Number of identifiers

drvt_num

Number of derived tables

max_query_spec

Maximum number of query specifications that can be specified in an SQL statement

vtbl

Number of viewed tables in an SQL statement

vquery_num
Number of query specifications specified in the view definition

vjotbl_num
Number of joined tables specified in the view definition

vtbl_num
Number of base tables specified in the view definition

jointbln
Number of joined tables

subqun
Number of subqueries

queryn
Number of query specifications

col_num
Number of selection expressions

pwl
Number of `WITH` clauses

drvtbln
Number of derived tables specified in the `FROM` clause

drvcolnum
Number of columns in derived tables

fmt_len
Format size of the `CONVERT` scalar function (bytes)

drvtblnum
Number of derived tables

viewcolnum
Number of columns in viewed tables

viewtblnum
Number of viewed tables

qurycolnum
Number of columns in query names

qurytblnum
Number of query names

charnum
Number of bytes in each SQL statement

vcol_num
Number of columns specified in the view definition

vwindow_num
Number of window functions specified in the view definition

vgroup_num
Number of `GROUP BY` clauses specified in the view definition

vsetf_num
Number of set functions specified in the view definition

gbyn
Number of `GROUP BY` clauses

qexp
Number of set operations

drvjotblnum
Number of joined tables specified in a derived table

drvquerynum
Number of queries specified in a derived table

drvwindownum
Number of window functions specified in a derived table

drvgroupnum
Number of `GROUP BY` clauses specified in a derived table

drvsetfnum
Number of set functions specified in a derived table

projn
Number of value expressions

max_drvtbln
Maximum number of tables that can be specified in the `FROM` clause

fjoin_drvquerynum
Number of queries specified in a derived table contained in `FULL OUTER JOIN`

fjoin_tbln
Number of table references in `FULL OUTER JOIN`

fjoin_tblcoln
Number of columns in table references in `FULL OUTER JOIN`

winfuncn
Number of window functions

query_expn
Number of query expressions

setfuncn
Number of set functions (+2 in the case of `AVG`)

drv_setopnum
Number of set operations specified in a derived table

archk_tbln
Number of archivable multi-chunk tables

csvrd_fieldnonum
Number of field data numbers specified in the `ADB_CSVREAD` function

predn
Number of predicates

qurytblfuncn
Number of table function derived tables specified for query names

tblfuncn
Number of table function derived tables

colspen
Number of column specifications

concat_n
Number of concatenation operations

gname_coln
Number of grouping column column names

inscoln
Number of inserted columns

prep_tbln
Number of pre-processing tables

withlistnum_recursion
Number of `WITH` list elements of recursive queries

withlistnum_recursion_query
Number of query specifications in recursive queries

vsetop_num
Number of set operations specified in the view definition

audrd_flpthn
Number of audit trail file path names specified in the `ADB_AUDITREAD` function

tblfunc_coln
Number of columns in table function derived tables

setfunc_distn
Number of `DISTINCT` set functions

TOTALSQLSCT
Total preprocessing results control area created by a transaction during preprocessing of SQL statements

When SQL statements are executed from the JDBC driver, this is the total preprocessing results control area for preprocessing of SQL statements created at the same time inside a single connection.

Use the following formula to determine its value.

Formula (kilobytes)

```
TOTALSQLSCT = sum_sql(SQLPSSZ + SQLSSZ + APATHVIEW) × max_users
```

max_users
Value specified for the `adb_sys_max_users` operand in the server definition

sum_sql ()
Substitute the maximum value among the calculated results for the variable *SQLSSZ*.

When SQL statements are executed from the JDBC driver, substitute the maximum value among the values determined for the SQL statements to be executed in a single connection.

SQLPSSZ
Preprocessing results control area acquired for preprocessing of SQL statements

Use the following formula to determine its value.

Formula (kilobytes)

```
SQLPSSZ = ↑184 ÷ 1,024↑
```

SQLSSZ

Preprocessing results control area used for preprocessing of SQL statements

Use the following formula to determine its value.

Formula (kilobytes)

$$
\begin{aligned}
SQLSSZ = \uparrow \{ & 27{,}392 + (768^{\#} \times tblcoln) + (2{,}224 \times tblidxn) + (8{,}350 \times tbln) \\
& + (1{,}192 \times col\_num) + (8 \times insvaln) + (240 \times inscoln) + (173 \times updcoln) \\
& + (2{,}288 \times colspen) + (1{,}640 \times gbycoln) + (72 \times operan) + (16 \times liken) + (128 \times inn) \\
& + (98 \times scafuncn) + (5{,}688 \times setfuncn) + (96 \times notn) + (96 \times orn) \\
& + (24 \times param\_num) + (16 \times sregn) + (136 \times rown) + (64 \times jointbln) \\
& + (704 \times inscolnamn) + (2{,}000 \times subqun) + (616 \times ord\_num) + (6{,}682 \times queryn) \\
& + (144 \times logicn) + (536 \times predn) + (128 \times base\_num) + (244 \times sel\_num) \\
& + (72 \times pwg) + (160 \times scl) + (8 \times sgc) + (8 \times ssk) + (16 \times srw) \\
& + (432 \times exp\_num) + (2 \times view\_size) + \left(sum\_view\_object\,(view\_size) \times 2.5 \right) \\
& + (1{,}272 \times qexp\_coln) + (88 \times gname\_coln) + \left(24 \times default\_coln \right) \\
& + (2{,}568 \times setfuncn) + (16 \times act\_num) + (16 \times decode\_revn) \\
& + (2 \times fmt\_len) + (88 \times obysorn) + (16 \times pw\_when) \\
& + (24 \times projn) + (2{,}168 \times fulljoin\_num) + (920 \times fj\_join\_num) \\
& + (208 \times gbyn) + (384 \times qexp) + (3{,}648 \times query\_expn) + (28{,}296 \times winfuncn) \\
& + (576 \times winfuncn \times winfuncn) + (576 \times col\_num \times winfuncn) + (576 \times gbycoln \times winfuncn) \\
& + (576 \times setfuncn \times winfuncn) + (24 \times withlistnum) \\
& + (16 \times likeregn) + (624 \times tblfuncn) + (80 \times multisetn) + (40 \times rowvaluecon) \\
& + (32 \times tablevaluecon) + \left((8 \times rowvaluecon + 200) \times rowvaluecon\_coln \right) \\
& + (112 \times gexp\_coln) + (8 \times fucall\_paramn) + (64 \times csvrd\_n) \\
& + (5 \times csvrd\_fieldnonum) + (24 \times multiset\_projn) + (2{,}336 \times archk\_tbln) \\
& + (5 \times archk\_coln) + (304 \times drvtblnum) + (16 \times comp\_dec\_num) + (104 \times contains\_n) \\
& + (13 \times def\_char\_datesn) + (402 \times drvcolnum) + (256 \times tblfunc\_coln) \\
& + (16 \times pw\_when) + (regex\_num) + (8 \times scalfunc\_paramn) + (128 \times setfuncn0) \\
& + (24 \times winf\_part\_projn) + (1{,}565 \times audrd\_n) + (16 \times ltdecode\_revn) \\
& + (8 \times randcursornum) + (16 \times randrownum) \\
\} & \div 1{,}024 \uparrow
\end{aligned}
$$

#

For a table that stores cost information, replace this value with 32,776.

tblcoln

Number of columns in the table being processed

tblidxn

Number of indexes defined for the table being processed

tbln

Number of tables specified in the FROM clause

col_num
Number of selection expressions

insvaln
Number of insertion values

inscoln
Number of inserted columns

updcoln
Number of updated columns

colspen
Number of column specifications

gbycoln
Number of grouping columns specified in the `GROUP BY` clause

operan
Number of arithmetic operations

liken
Number of `LIKE` predicates

inn
Number of `IN` predicates

scafuncn
Number of scalar functions

setfuncn
Number of set functions (+2 in the case of `AVG`)

notn
Number of `NOT`s

orn
Number of `OR`s

param_num
Number of dynamic parameters

sregn
Total number of datetime information acquisition functions and user information acquisition functions

rown
Number of `ROW`s

jointbln
Number of joined tables

inscolnamn
Number of columns to be inserted by the `INSERT` statement

subqun
Number of subqueries

ord_num
Number of columns specified in the `ORDER BY` clause

queryn

Number of query specifications

logicn

Number of AND, OR, and NOT logical operations

predn

Number of predicates

base_num

Number of underlying tables specified in the view definition

sel_num

Number of selection expressions with query specification specified in the view definition

pwg

Number of labeled durations

scl

Number of columns (sum total of the number of column specifications and the number of column specifications after expansion with * specification)

sgc

Number of grouping columns

ssk

Number of sort keys specified in the ORDER BY clause

srw

Number of DECIMAL types and TIMESTAMP types contained in tables targeted by ROW

exp_num

Total number of value expressions

sum_view_object ()

Sum total of the view object sizes of viewed tables specified in an SQL statement for the variable in parentheses (bytes)

view_size

View object size (value of the VIEW_OBJECT_SIZE column of the SQL_VIEWS dictionary table (base table)) (bytes)

qexp_coln

Number of columns in a query expression within a set operation

gname_coln

Number of grouping column column names

default_coln

Number of columns for which the DEFAULT clause is specified

act_num

Total number of THENs and ELSEs in a CASE expression

decode_revn

Number of return values for the DECODE scalar function

fmt_len

Format size of the CONVERT scalar function (bytes)

obysorn
Number of sort items specified in the `ORDER BY` clause

pw_when
Number of `WHEN`s specified in a `CASE` expression

projn
Number of value expressions

fulljoin_num
Number of times `FULL OUTER JOIN` is specified in SQL statements

fj_join_num
Number of joined tables specified in `FULL OUTER JOIN`

gbyn
Number of `GROUP BY` clauses

qexp
Number of set operations

query_expn
Number of query expressions

winfuncn
Number of window functions

withlistnum
Number of elements in the `WITH` list

likeregn
Number of the `LIKE_REGEX` predicates

tblfuncn
Number of table function derived tables

multisetn
Number of multiset value expressions

rowvaluecon
Number of row value constructors

tablevaluecon
Number of table value constructors

rowvaluecon_coln
Number of columns of a row value constructor

gexp_coln
Number of group value expressions

fucall_paramn
Number of function call arguments

csvrd_n
Number of `ADB_CSVREAD` functions

csvrd_fieldnonum
Number of field data numbers specified in the `ADB_CSVREAD` function

multiset_projn

Number of multiset elements in a multiset value expression

archk_tbln

Number of archivable multi-chunk tables

archk_coln

Number of derived columns in an archivable multi-chunk table

drvtblnum

Number of derived tables

comp_dec_num

Number of operations that specify `DECIMAL` type data and `INTEGER` or `SMALLINT` type data in comparison operands

contains_n

Number of `CONTAINS` scalar functions

def_char_datesn

Number of predefined character-string representations in datetime data

drvcolnum

Number of columns in derived tables

tblfunc_coln

Number of columns in table function derived tables

regex_num

Number of `LIKE_REGEX` predicates specified

scalfunc_paramn

Number of scalar function arguments

setfuncn0

Number of set functions

winf_part_projn

Number of value expressions in window partition clauses

audrd_n

Number of `ADB_AUDITREAD` functions

ltdecode_revn

Number of return values for the `LTDECODE` scalar function

randcursornum

Number of `RANDOMCURSOR` scalar functions

randrownum

Number of `RANDOMROW` scalar functions

APATHVIEW

Access path information control area that is created for displaying an access path

Use the following formula to determine its value.

Formula (kilobytes)

```
APATHVIEW =
   ↑((9,728 + 1,024 × scl) × tbln + 2,200) ÷ 4,096↑ × 5
```

tbln

Number of tables specified in the `FROM` clause

scl

Number of columns (sum total of the number of column specifications and the number of column specifications after expansion with * specification)

PREDICSZ

Substitute the following value.

Value (kilobytes)

$$PREDICSZ = 19,320$$

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(c) Determining the variable PROC_EXECSQLSZ

The variable *PROC_EXECSQLSZ* is required when executing SQL statements. Use the following formula to determine its value.

Formula (kilobytes)

$$PROC\_EXECSQLSZ = TRANSNAP + RTHDUPDINF + DBUPDINF^{\#}$$

#

Add this value when executing the `PURGE CHUNK` or `TRUNCATE TABLE` statement.

Explanation of variables

TRANSNAP

Transaction information snapshot

Use the following formula to determine its value.

Formula (kilobytes)

$$TRANSNAP = \uparrow(112 + (88 \times max\_users)) \div 1,024\uparrow$$

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

RTHDUPDINF

Processing real thread update information

Use the following formula to determine its value.

Formula (kilobytes)

$$RTHDUPDINF=$$
$$\uparrow(648 + (168 \times rthd\_num)) \div 1,024\uparrow$$

rthd_num

Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

If the adb_sql_exe_max_rthd_num operand is specified in the client definition, assign that value (the value of the adb_sql_exe_max_rthd_num operand in the client definition). However, if the value of the adb_sql_exe_max_rthd_num operand in the client definition is greater than the value of the adb_sql_exe_max_rthd_num operand in the server definition, assign the value of the adb_sql_exe_max_rthd_num operand in the server definition.

For details, see the explanation of the adb_sql_exe_max_rthd_num operand in 7.2.2 Operands related to performance (set format).

DBUPDINF

DB area, table, index, and chunk update information

Use the following formula to determine this value. Note that the formula to be used differs depending on whether the multi-node function is used.

Formula (kilobytes)

- If the multi-node function is not used
  DBUPDINF = ↑((40 + ↑(*archive_file_num* / 1,024)↑ x 122,888) / 1,024)↑
- If the multi-node function is used
  DBUPDINF = 129 + ↑((40 + ↑(*archive_file_num* / 1,024)↑ x 122,888) / 1,024)↑

archive_file_num

Specify the number of archive files for the archived chunk.

Use the adbdbstatus command to output the summary information of archived chunks, and then check the value of Archive_file_num for each chunk. For details about Archive_file_num, see the following section in the manual *HADB Command Reference*: *List of items that are output in the summary information of archived chunks* in *Items that are output in the summary information of archived chunks* in *adbdbstatus (Analyze the Database Status)*.

(d) Determining the variable PROC_BUFWORK_CTL

The variable *PROC_BUFWORK_CTL* is required when a work table is used. Use the following formula to determine its value.

Formula (kilobytes)

PROC_BUFWORK_CTL = *max_users* + *rthd_num*

Explanation of variables

max_users: Value specified for the adb_sys_max_users operand in the server definition

rthd_num: Value specified for the adb_sys_rthd_num operand in the server definition

(e) Determining the variable PROC_DEFSQLSZ

The variable *PROC_DEFSQLSZ* is required when executing definition SQL statements. Use the following value.

The value to substitute differs depending on whether the multi-node function is used.

Value to substitute when the multi-node function is not used (kilobytes)

PROC_DEFSQLSZ = 2,261

Value to substitute when the multi-node function is used (kilobytes)

```
PROC_DEFSQLSZ = 7,208
```

Note:

If you define a base table or an index for a base table, the amount of process common memory (DB area information) that is required increases. Determine the value of the variable *DBAREAINF* again, according to (e) Determining the variable DBAREAINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(f) Determining the variable PROC_UPDSZ

Add the variable *PROC_UPDSZ* when you execute a definition SQL statement or update SQL statement for the first time after database connection. Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_UPDSZ = BUFUPD × upd_users
```

Explanation of variables

upd_users

Number of concurrent executions of a connection (application program or command) for acquiring process common memory for managing updated pages

BUFUPD

Process common memory for managing updated pages

Use the following formula to determine its value.

Formula (kilobytes)

```
BUFUPD =
    ↑(56 + BUFHIST + BUFFLU + BUFBLK) ÷ 1,024↑
```

The following table shows the timing at which process common memory for managing updated pages is allocated.

Table 6-4: Memory allocation timing

No.	Execution type		Allocated or not
1	• Application program execution • `adbsql` command execution	• Connection in which a definition SQL statement is executed • Connection in which an update SQL statement is executed	Y#1
2		Connection in which only a retrieval SQL statement is executed	N#2, #3
3	`adbinit` command execution		Y
4	`adbimport` command execution		Y
5	`adbidxrebuild` command execution		Y
6	`adbgetcst` command execution		Y#4
7	`adbdbstatus` command execution		N
8	`adbstat` command execution		N
9	`adbmodarea` command execution		Y

No.	Execution type	Allocated or not
10	`adbexport` command execution	N#2
11	`adbmergechunk` command execution	Y
12	`adbchgchunkcomment` command execution	Y#4
13	`adbmodbuff` command execution	N
14	`adbarchivechunk` command execution	Y
15	`adbunarchivechunk` command execution	Y

Legend:

Y: Process common memory for managing updated pages is allocated.

N: Process common memory for managing updated pages is not allocated.

#1

If an error occurs during preprocessing, process common memory for managing updated pages is not allocated.

#2

Process common memory for managing updated pages is not allocated for writing data to a work table.

#3

If a definition SQL statement or update SQL statement is executed even once during a connection, the connection buffer remains allocated.

#4

Process common memory for managing updated pages is allocated in order to update the system tables.

BUFHIST

Use the following formula to determine its value.

Formula (bytes)

```
BUFHIST =
    116 × RTHNUM + 44,052 + BUFMEM(RTHNUM + 257) + BUFMEM(RTHNUM + 4,097)
```

BUFMEM

For details, see the description of the variable *BUFMEM* in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

BUFFLU

Use the following formula to determine its value.

Formula (bytes)

```
BUFFLU =
    8 × uthd_num + 96 × RTHNUM + 393,312
```

uthd_num

Value specified for the `adb_sys_uthd_num` operand in the server definition

BUFBLK

Use the following formula to determine its value.

Formula (bytes)

```
BUFBLK =
    33,734,656 + 1,024 × BUFLOG
```

BUFLOG

For details, see the description of the variable *BUFLOG* in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(g) Determining the variable PROC_UPDLISTSZ

Add the variable *PROC_UPDLISTSZ* when you use the multi-node function. Use the following formula to determine the value.

Formula (kilobytes)

PROC_UPDLISTSZ = 147 × *max_users*

Explanation of variables

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

(2) Determining the real thread private memory requirement (during normal operation)

Use the following formula to determine the real thread private memory requirement for normal operation (*RTHD_EXECSZ*).

Formula (kilobytes)

```
RTHD_EXECSZ = RTHD_CNCTSZ + RTHD_DEFSQLSZ + RTHD_EXESQLSZ + RTHD_ROLLBKSZ
            + RTHD_DBEXTSZ + RTHD_WORKBUF + RTHD_EXESQLDICSZ
            + RTHD_COMMUSZ + RTHD_MODASZ + RTHD_EXPSQLTRCSZ
            + RTHD_AUDINFSZ + RTHD_COLUMNIZESZ
```

Explanation of variables

- *RTHD_CNCTSZ*
  Real thread private memory used when connecting to a database
  Determine the value as explained in (a) Determining the variable RTHD_CNCTSZ.
- *RTHD_DEFSQLSZ*
  Real thread private memory used when executing definition SQL statements
  Determine the value as explained in (b) Determining the variable RTHD_DEFSQLSZ.
- *RTHD_EXESQLSZ*
  Real thread private memory used when executing data manipulation SQL statements
  Determine the value as explained in (c) Determining the variable RTHD_EXESQLSZ.
- *RTHD_ROLLBKSZ*
  Real thread private memory used during rollback processing or when restarting the HADB server
  Determine the value as explained in (d) Determining the variable RTHD_ROLLBKSZ.

- *RTHD_DBEXTSZ*
  Real thread private memory used when extending a DB area
  Determine the value as explained in (e) Determining the variable RTHD_DBEXTSZ.
- *RTHD_WORKBUF*
  Real thread private memory used when processing work tables
  Determine the value as explained in (f) Determining the variable RTHD_WORKBUF.
- *RTHD_EXESQLDICSZ*
  Real thread private memory used when preprocessing data manipulation SQL statements
  Determine the value as explained in (g) Determining the variable RTHD_EXESQLDICSZ.
- *RTHD_COMMUSZ*
  Real thread private memory used for communication processing
  Determine the value as explained in (h) Determining the variable RTHD_COMMUSZ.
- *RTHD_MODASZ*
  Real thread private memory used for executing the `adbmodarea` command
  Determine the value as explained in (2) Determining the real thread private memory requirement (for executing the adbmodarea command) under 6.3.12 Determining the memory requirement for executing the adbmodarea command.
- *RTHD_EXPSQLTRCSZ*
  Real thread private memory used when the SQL trace buffer is extended
  Determine the value as explained in (i) Determining the variable RTHD_EXPSQLTRCSZ.
- *RTHD_AUDINFSZ*
  Real thread private memory used when the audit trail facility is enabled
  Determine the value as explained in (j) Determining the variable RTHD_AUDINFSZ.
- *RTHD_COLUMNIZESZ*
  Real thread private memory used when the updated-row columnizing facility is enabled
  Determine the value as explained in (k) Determining the variable RTHD_COLUMNIZESZ.

(a) Determining the variable RTHD_CNCTSZ

The variable *RTHD_CNCTSZ* is added when connecting to a database. Use the following formula to determine its value.

Formula (kilobytes)

RTHD_CNCTSZ = *TRANSNAP* + *SQLTRC*

Explanation of variable

TRANSNAP
For details, see the description of the variable *TRANSNAP* in (c) Determining the variable PROC_EXECSQLSZ under (1) Determining the process common memory requirement (during normal operation).

SQLTRC
Management information for SQL tracing
Assign 65,536 kilobytes.

(b) Determining the variable RTHD_DEFSQLSZ

The variable *RTHD_DEFSQLSZ* is required when executing definition SQL statements. Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_DEFSQLSZ=
    17+SZ_ALTER_TABLE+SZ_ALTER_VIEW+SZ_CREATE_INDEX+SZ_CREATE_TABLE
      +SZ_CREATE_VIEW+SZ_DROP_INDEX+SZ_DROP_TABLE
      +SZ_DROP_VIEW+SZ_GRANT+SZ_REVOKE
```

Explanation of variable

SZ_ALTER_TABLE

Substitute the following value when you change the definition of a base table.

Value (kilobytes)

```
SZ_ALTER_TABLE= 377
```

SZ_ALTER_VIEW

Substitute the following value when you change the definition of a viewed table.

Value (kilobytes)

```
SZ_ALTER_VIEW= 266
```

SZ_CREATE_INDEX

Substitute the following value when you define an index.

Substitute the following value also when you include a uniqueness constraint definition or chunk-archive specification in the `CREATE TABLE` statement.

Value (kilobytes)

```
SZ_CREATE_INDEX= 42
```

SZ_CREATE_TABLE

Substitute the following value when you define a base table.

Value (kilobytes)

```
SZ_CREATE_TABLE= 671
```

SZ_CREATE_VIEW

Substitute the following value when you define a viewed table.

Value (kilobytes)

```
SZ_CREATE_VIEW= 217
```

SZ_DROP_INDEX

Substitute the following value when you delete an index.

Substitute the following value also when an index is deleted as a result of processing after the `DROP USER`, `DROP SCHEMA`, or `DROP TABLE` statement is executed.

Value (kilobytes)

```
SZ_DROP_INDEX= 39
```

SZ_DROP_TABLE

Substitute the value shown below when you delete a base table.

Substitute the following value also when a base table is deleted as a result of processing after the `DROP USER` or `DROP SCHEMA` statement is executed.

Value (kilobytes)

```
SZ_DROP_TABLE = 467
```

SZ_DROP_VIEW

Substitute the following value when you delete a viewed table.

Substitute the following value also when a viewed table is deleted as a result of processing after the `DROP USER`, `DROP SCHEMA`, or `DROP TABLE` statement is executed.

Value (kilobytes)

```
SZ_DROP_VIEW= 191
```

SZ_GRANT

Substitute the following value when you grant a privilege.

Value (kilobytes)

```
SZ_GRANT= 33,178
```

SZ_REVOKE

Substitute the following value when you revoke a privilege.

Substitute the following value also when a privilege is revoked as a result of processing after the `DROP USER`, `DROP SCHEMA`, or `DROP TABLE` statement is executed.

Value (kilobytes)

```
SZ_REVOKE= 33,410
```

(c) Determining the variable RTHD_EXESQLSZ

The variable *RTHD_EXESQLSZ* is required when executing data manipulation SQL statements. Use the following formula to determine its value.

Formula (kilobytes)

RTHD_EXESQLSZ=
MAX（*max_sql*(*SQL_CNCT_SIZE*)×*max_sql_concurrent_exec_num*， *max_sql*(*PROC_THD_SIZE*)）
＋*BUF_DLYSZ*＋*ARC_DIR_PATH* #

#

Add this value when executing the `PURGE CHUNK` or `TRUNCATE TABLE` statement for an archivable multi-chunk table.

Explanation of variables

max_sql()

For each SQL statement to be executed, calculate the result for the variable in the parentheses, and then substitute the maximum calculated value.

max_sql_concurrent_exec_num

Maximum number of SQL statements that will be executed at the same time during a transaction

SQL_CNCT_SIZE: Data manipulation SQL control area

Use the following formula to determine its value.

Formula (kilobytes)

```
SQL_CNCT_SIZE = STATEMENT_SIZE + CNCT_THD_SIZE + 0.3
```

STATEMENT_SIZE: Statement handle control area

Use the following formula to determine its value.

Formula (kilobytes)

$$STATEMENT\_SIZE = \uparrow(2{,}496 + PREPARE\_INFO\_SIZE + PARAM\_INFO\_SIZE + APATHVIEW) \div 1{,}024\uparrow$$

PREPARE_INFO_SIZE: SQL statement preprocessing information

Use the following formula to determine its value.

Formula (bytes)

```
PREPARE_INFO_SIZE = col_num × 214 + param_num × 80 + idx_num × 512
```

col_num: Number of selection expressions specified in the SQL statement

param_num: Number of dynamic parameters specified in the SQL statement

idx_num: Number of indexes used by the SQL statement

PARAM_INFO_SIZE: Parameter storage area

Use the following formula to determine its value.

Formula (bytes)

$$PARAM\_INFO\_SIZE = \sum_{i=1}^{param\_num} param\_size_{(i)}$$

param_num: Number of dynamic parameters specified in the SQL statement

param_size(i): Maximum data length of each dynamic parameter

APATHVIEW: Access path information control area that is created for displaying an access path

See the description of the variable *APATHVIEW* in (b) Determining the variable PROC_TOTALSQLSSZ under (1) Determining the process common memory requirement (during normal operation).

CNCT_THD_SIZE: SQL execution control area

Use the following formula to determine its value.

Formula (kilobytes)

$$CNCT\_THD\_SIZE =$$

$$\left\lceil \left\{ 3{,}776 + 752 \times tbl\_num + 2{,}976 \times sql\_rthd\_num + 200 \times setop\_num + query\_num \right.\right.$$

$$\times \left(1{,}360 + 496 \times sql\_rthd\_num + 16 \times (sql\_rthd\_num \times sql\_rthd\_num) + 160 \times window\_num \right)$$

$$+ tbl\_num \times sql\_rthd\_num \times (96 + 192 \times sql\_rthd\_num) + 112 \times drvtbl\_num$$

$$+ (72 + 16 \times sql\_rthd\_num) \times tblfunc\_num$$

$$+ RESULT\_WORK\_SIZE + OPE\_WORK\_SIZE + \sum (HASHGRP\_SIZE)$$

$$+ HASHTBL\_CNCT\_SIZE + (SCAN\_WORK\_SIZE + 88) \times 5 \times tbl\_num$$

$$+ IN\_PRD\_SIZE + LIKE\_PRD\_SIZE + REG\_SIZE$$

$$+ \sum (LMTWRK\_SIZE) + COR\_QUERY\_SIZE + TXT\_WORK\_SIZE + REGEXP\_WORK\_SIZE$$

$$+ \mathrm{MAX}(CSVREAD\_WORK\_SIZE)$$

$$+ 384 \times tvc\_num + 8 \times query\_num \times \mathrm{MIN}(1 , \mathrm{MAX}(tvc\_num , in\_query\_num))$$

$$+ 256 \times in\_query\_num + RECURSIVE\_QUERY\_CNCT\_SIZE + RANDOMCURSOR\_WORK\_SIZE$$

$$\left. + RANDOM\_NUM\_WORK\_SIZE + RNGIDX\_INFO\_SIZE \right\} \div 1{,}024 \left\rceil \right.$$

Important

■ Notes about specifying viewed tables and query names

When specifying a viewed table or query name in an SQL statement, estimate the memory requirements as if an internal derived table corresponding to each viewed table or query name were applied. For a recursive query name that references a target recursive query within a recursive query, estimate memory requirements assuming the work table that stores the results of the recursive query.

Example:

- SQL statement that defines the viewed table

```
CREATE VIEW "V1" AS SELECT * FROM "T1","T2" WHERE "T1"."C1"="T2"."C2"
```

- SQL statement to be executed

```
SELECT * FROM "V1","T3"
           WHERE "V1"."C1"="T3"."C3"
```

- SQL statement for which memory requirements are to be estimated

```
SELECT * FROM (SELECT * FROM "T1","T2" WHERE "T1"."C1"="T2"."C2"),"T3"
           WHERE "V1"."C1"="T3"."C3"
```

■ Notes when specifying an archivable multi-chunk table

When specifying an archivable multi-chunk table in an SQL statement, estimate memory requirements as if the archivable multi-chunk table were converted by equivalent exchange to a derived table. For details about the equivalent exchange of archivable multi-chunk tables, see

Equivalent exchange of SQL statements that search archivable multi-chunk tables in the *HADB Application Development Guide*.

tbl_num: Number of tables specified

If you specify `FULL OUTER JOIN`, add the number of times `FULL OUTER JOIN` is specified.

sql_rthd_num

Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

setop_num: Number of set operators specified in SQL statements

If you specify `FULL OUTER JOIN`, add the number of times `FULL OUTER JOIN` is specified.

query_num: Number of query specifications

Determine this value from the number of `FROM` clauses specified in SQL statements.

If you specify `FULL OUTER JOIN`, add the number obtained by doubling the number of times `FULL OUTER JOIN` is specified.

window_num: Number of window functions specified in SQL statements

If multiple `DISTINCT` set functions are specified with different arguments in one query specification, also add the number of those set functions - 1.

drvtbl_num: Number of derived tables specified in SQL statements

If you specify `FULL OUTER JOIN`, add the number of times `FULL OUTER JOIN` is specified.

tblfunc_num

Number of table function derived tables

Σ(*HASHGRP_SIZE*)

Sum total of the *HASHGRP_SIZE* values calculated for the individual queries that are specified in SQL statements (bytes)

Σ(*LMTWRK_SIZE*):

Sum total of the *LMTWRK_SIZE* values calculated for the individual work tables that are created (bytes)

For the following issues, see *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*:

- Whether an SQL statement in which the `LIMIT` clause is specified creates a work table
- Number of work tables that are created

If the `LIMIT` clause is used to specify the maximum number of rows to be acquired from the results of a set operation, the value specified for the `LIMIT` clause is applied to each of the query specifications that comprise the set operation.

MAX(CSVREAD_WORK_SIZE):

Determine in bytes the value of the *CSVREAD_WORK_SIZE* variable for each table function derived table for which the `ADB_CSVREAD` function is specified. Then substitute the largest among the determined values.

tvc_num:

Number of table value constructors specified in SQL statements

in_query_num:

Total number of predicates that satisfy the following conditions:

- `IN` predicate

A table subquery is specified on the right side, and the column specified on the left-side value expression is the first of the B-tree indexed columns to be used for a search.

- Quantified predicate

  The column specified on the left-side value expression in = ANY or = SOME is the first of the B-tree indexed columns to be used for a search.

RESULT_WORK_SIZE: SQL execution result storage area

Use the formula shown below to determine its value. To calculate the data length, see Table 6-9: Data length of each data type.

Formula (bytes)

```
RESULT_WORK_SIZE =
    ↑(40 + sum_col_size + row_size
    + 2 × (sum_grp_size + sum_window_spec_size)
    + 3 × (sum_set_func_size + sum_dist_col_size + sum_window_func_size)
    + 8 × (all_col_num + row_num + 2 × (grp_col_num + window_spec_col_num)
    + 3 × (set_func_num + dist_col_num + window_func_num))
    + sum_inpred_size + sum_reg_size
    + 8 × group_query_num + 12 × list_num
    + 16 × join_num + supquery_num
    + sum_setop_size × 3 + full_join_drvc_size × 3
    + (func_filepath_size + 4) × (csvread_num + auditread_num)) ÷ 16↑ × 16
```

- *sum_col_size*

  Sum total of the data lengths of all specified columns

- *row_size*

  Sum total of the row lengths of the table from which data is acquired using ROW

- *sum_grp_size*

  Sum total of the data lengths of the grouping columns

- *sum_window_spec_size*

  Sum total of the data lengths of the columns specified in window specifications

- *sum_set_func_size*

  Sum total of the data lengths of all set function results

- *sum_dist_col_size*

  Sum total of the data lengths of the columns specified as arguments in the DISTINCT set function

- *sum_window_func_size*

  Sum total of the data length values of the results of window functions

- *all_col_num*

  Number of all columns specified

- *row_num*

  Number of ROWs specified

- *grp_col_num*

  Number of grouping columns specified in the GROUP BY clause

- *window_spec_col_num*

  Number of columns specified in window specifications

- *set_func_num*

  Number of set functions

- *dist_col_num*
  Number of columns specified as arguments of the DISTINCT set function
- *window_func_num*
  Number of window functions
- *sum_inpred_size*
  Sum total of the sizes of the result creation areas of all IN predicates specified (29 + 8 × *the-number-of-value-expressions-specified-on-the-right-side-of-the-IN-predicate*). The value is 0 if no IN predicate is specified.
- *sum_reg_size*
  Sum total of the data length for all datetime information acquisition functions and user information acquisition functions specified
- *group_query_num*
  Number of query specifications in which the GROUP BY clause is specified
- *list_num*
  Number of work tables
  Substitute the combined total of the following values:
  • + 1 if the ORDER BY clause is specified
  • Number of query specifications in which the GROUP BY clause is specified
  • Number of query specifications in which SELECT DISTINCT is specified
  • Number of times the DISTINCT set function is specified
  • Number of times the inverse distribution function is specified
  • Number of times the window function is specified
  • Number of query specifications in which multiple table references are specified in the FROM clause
  • Number of derived tables
  • Number of viewed tables specified
  • Number of WITH clauses
  • Number of joined tables specified (to be counted as 2 in the case of FULL OUTER JOIN)
  • Number of subqueries specified
  • Number of set operations specified + 1
- *join_num*
  Number of joined tables
  Count each table as 2 in the case of FULL OUTER JOIN.
- *supquery_num*
  Number of subqueries
- *sum_setop_size*
  For each set operation, total the data lengths of the selection expressions that result from the set operation. Then, substitute the value obtained by adding up these lengths.
- *full_join_drvc_size*
  For each query specification in which FULL OUTER JOIN is specified, determine the data length of the selection expression. Then, substitute the value obtained by adding up these lengths.
- *func_filepath_size*
  Maximum length of file path names specified in ADB_CSVREAD functions and ADB_AUDITREAD functions

- *csvread_num*
  Number of ADB_CSVREAD functions
- *auditread_num*
  Number of ADB_AUDITREAD functions

OPE_WORK_SIZE: Operation control area

Use the formula shown below to determine its value. However, its value is 0 if no arithmetic operations, logical operations, CASE expressions, or scalar functions are specified. To calculate the data length, see Table 6-9: Data length of each data type.

Formula (bytes)

$$OPE\_WORK\_SIZE = \left\{ \left\downarrow \frac{(571 + 32 \times arith\_num + 24 \times concat\_num + 24 \times logical\_num + case\_size + sclfunc\_size)}{512} \right\downarrow \right\} \times 512 + 40$$

arith_num: Maximum number of arithmetic operators (+, −, * or /) in each value expression

concat_num: Maximum total number of concatenation operators (+ or ||) and CONCAT scalar functions in value expressions

logical_num: Maximum number of logical operators in search conditions

case_size: Sum total of the result length values of CASE expressions

sclfunc_size: Sum total of the result length values of scalar functions

HASHGRP_SIZE: Hash group area

Use the formula shown below to determine its value. However, its value is 0 if 0 is specified for the adb_sql_exe_hashgrp_area_size operand in the server definition or if no GROUP BY clause or set function is specified in the SQL statement. To calculate the data length, see Table 6-9: Data length of each data type.

Formula (bytes)

$$HASHGRP\_SIZE = 160 + 24 \times (grp\_col\_num + set\_func\_num) + sum\_grp\_size + sum\_set\_func\_size + hashgrp\_byte + \downarrow(hashgrp\_byte \times 4 - MIN(8{,}192 \times 128, \downarrow(hashgrp\_byte \times 10) \div 25\downarrow)) \div (\uparrow(16 + \uparrow(sum\_grp\_size) \div 4\uparrow \times 4 + sum\_set\_func\_size) \div 8\uparrow \times 8)\downarrow$$

- *grp_col_num*
  Number of grouping columns specified in the GROUP BY clause
- *set_func_num*
  Number of set functions
- *sum_grp_size*
  Sum total of the data lengths of the grouping columns
- *sum_set_func_size*
  Sum total of the data lengths of all set function results
- *hashgrp_byte*
  Value specified for the hash group area size
  Use the following formula to determine its value:

value-specified-for-adb_sql_exe_hashgrp_area_size-operand-in-server-definition × 1,024

HASHTBL_CNCT_SIZE: Management area when retrieval is performed using a hash table

Use the formula shown below to determine its value if the SQL statement contains any of the items listed here. Note that the value is 0 if the SQL statement contains none of these items. The value is also 0 when the maximum number of SQL processing real threads or the hash table area size is 0.

- Query specification that uses the equal sign (=) to join multiple tables
- Subquery
- `GROUP BY` clause
- `DISTINCT` set function
- `SELECT DISTINCT`
- `UNION` or `UNION DISTINCT`

Formula (bytes)

$$HASHTBL\_CNCT\_SIZE=$$
$$192+160\times sql\_rthd\_num+\left\{1{,}752+\left\lceil\frac{hashtbl\_row\_num}{2}\right\rceil+64\times hashtbl\_col\_num\right.$$
$$\left.+(2{,}320+392\times bucket\_num+24\times sql\_rthd\_num)\times sql\_rthd\_num\right\}$$
$$\times hashtbl\_num+HASHFLT\_CNCT\_SIZE+HASHRNG\_CNCT\_SIZE+GRSETS\_CNCT\_SIZE$$

- *sql_rthd_num*

  Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

- *hashtbl_row_num*

  Maximum row length in hash tables

  For details, see 6.25.3 Number of work tables created during retrieval using hash tables.

- *hashtbl_col_num*

  Substitute the maximum value from among the values determined according to the specifications within the SQL statements.

  ■ Query specification that uses the equal sign (=) to join multiple tables

  Maximum number of columns in a hash table. When you are determining the number of columns in a hash table in each query specification, for each of the tables joined by the equal sign (=), determine the total number of columns specified in selection expressions and search conditions. Then determine the total value, excluding the minimum value, for each table for which you determined the number of columns.

  ■ Subquery

  For each subquery, determine the sum total of the values listed below. Use the largest value among these summed values as the subquery value.

  • Number of columns specified in a subquery selection expression

  • Number of set functions specified in a subquery

  • Number of columns specified for predicates that include external reference columns within the search condition

  ■ GROUP BY clause and the DISTINCT set function

  For each query in which the `GROUP BY` clause and the `DISTINCT` set function are specified, determine the sum total of the values listed below. Use the largest value among these summed values as the value for the `GROUP BY` clause and the `DISTINCT` set function.

  • Number of grouping columns specified in the `GROUP BY` clause

• Number of set functions

■ SELECT DISTINCT

Determine the following value for each query specification that includes `SELECT DISTINCT`. Use the largest of the determined values as the value of `SELECT DISTINCT`.

• Number of columns specified in a selection expression

■ UNION or UNION DISTINCT

Determine the following value for each query expression body in which `UNION` or `UNION DISTINCT` is specified. Use the largest of the determined values as the value of `UNION` or `UNION DISTINCT`.

• Number of columns in the table that is derived by the query expression body

- *bucket_num*

  Number of work tables created during retrieval using hash tables

  For details, see 6.25.3 Number of work tables created during retrieval using hash tables.

- *hashtbl_num*

  Number of hash tables

  For details, see 6.25.3 Number of work tables created during retrieval using hash tables.

HASHFLT_CNCT_SIZE: Hash filter management area

Determine this value if either of the following items is included in the SQL statement. If neither of the items is included, this value is 0. This value is also 0 if the size of the hash filter area (value specified for the `adb_sql_exe_hashflt_area_size` operand in the server definition) is 0.

- Query specification that uses the equal sign (=) to join multiple tables
- Subquery

Formula (bytes)

```
HASHFLT_CNCT_SIZE = 256 + 256 × hashflt_num
```

- *hashflt_num*

  Total number of hash filters in the SQL statement

  Determine the total of all the following values:

  • Number of join conditions that use = specifications in query specifications that join multiple tables by using equal signs (=)

  • Number of subqueries that do not include any external reference columns

  • Number of = conditions that include external reference columns in subqueries that include external reference columns

HASHRNG_CNCT_SIZE: Management area for evaluation of range index conditions in hash retrieval

Determine this value if either of the following items is included in the SQL statement. If neither of the items is included, this value is 0.

- Query specification that uses the equal sign (=) to join multiple tables
- Subquery

Formula (bytes)

```
HASHRNG_CNCT_SIZE = 64 + 136 × hash_retrieval_rngidx_num + 56 × hash_cond_rngidx
_num
```

hash_retrieval_rngidx_num

Total number of hash retrieval processes that use a range index in the SQL statement

Determine the total of all the following values. If this total number is 0, the value of *HASHRNG_CNCT_SIZE* is 0.

- The value determined from the following formula for the query specifications in which a range-indexed column is specified for the equal sign (=), of the query specifications that use the equal sign (=) to join multiple tables:

  ```
  total-number-of-tables-joined-by-using-equal-sign - total-number-of-query-specifications-joined-by-using-equal-sign
  ```

- Number of quantified predicates for which a range-indexed column is specified on the left side
- Number of `IN` predicates for which a range-indexed column is specified on the left side and a table subquery is specified on the right side
- Number of subqueries (other than the preceding two subqueries) that satisfy all of the following conditions:
  - An external reference column is specified by using the equal sign (=) in a search condition
  - The specified external reference column is a range-indexed column

hash_cond_rngidx_num

Total number of conditions that use a range index for the hash retrieval processes in the SQL statement

Determine the total of all the following values:

- Number of = join conditions in which range-indexed columns are specified in the query specifications that use the equal sign (=) to join multiple tables
- Number of quantified predicates for which a range-indexed column is specified on the left side
- Number of `IN` predicates for which a range-indexed column is specified on the left side and a table subquery is specified on the right side
- Number of = conditions (other than the preceding two subqueries) that satisfy all of the following conditions:
  - An external reference column is specified by using the equal sign (=) in a search condition
  - The specified external reference column is a range-indexed column

GRSETS_CNCT_SIZE: Work area for the `DISTINCT` set function

Determine this value if the SQL statement includes a query specification in which multiple `DISTINCT` set functions are specified with different arguments. The value of this variable is 0 if the SQL statement does not include a query specification in which multiple `DISTINCT` set functions are specified with different arguments.

Formula (bytes)

```
GRSETS_CNCT_SIZE =
 (616 + 136 × dset_func_num + sql_rthd_num × (664 + 72 × dset_func_num)
   ) × dset_query_num
```

dset_func_num

Number of `DISTINCT` set functions specified with different arguments

sql_rthd_num

The maximum number of SQL processing real threads (the value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition).

dset_query_num

Number of query specifications in which `DISTINCT` set functions are specified with different arguments

IN_PRD_SIZE: `IN` predicate and quantified predicate execution management area

Use the formula shown below to determine its value. However, the value is 0 if no `IN` predicate or quantified predicate is specified. The variable *IN_PRD_SIZE* is 0 also if the variable *in_pred_num* is 0.

Formula (bytes)

```
IN_PRD_SIZE = 48 + in_pred_num × (96 + sql_rthd_num × 48)
```

sql_rthd_num: Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

in_pred_num: Total number of predicates that satisfy the following conditions:

- `IN` predicate

  The column specified on the left-side value expression is the first of the B-tree indexed columns to be used for a search.

- Quantified predicate

  The column specified on the left-side value expression in = `ANY` or = `SOME` is the first of the B-tree indexed columns to be used for a search.

LIKE_PRD_SIZE: `LIKE` predicate execution control area

Use the formula shown below to determine its value. Note that the value is 0 if no `LIKE` predicate is specified. The value is also 0 if the column specified as the match value of all `LIKE` predicates is not included in the B-tree indexed columns to be used for a search. To calculate the data length, see Table 6-9: Data length of each data type.

Formula (bytes)

```
LIKE_PRD_SIZE = 16 + like_param_num × (64 + like_param_size × 2)
```

- *like_param_num*

  Number of `LIKE` predicates in which the dynamic parameter is specified for the pattern character string. If the column specified as the match value of a `LIKE` predicate is not included in the B-tree indexed columns to be used for a search, that `LIKE` predicate is excluded from the count.

- *like_param_size*

  Maximum data length assumed by the dynamic parameter specified as the pattern character string. If the column specified as the match value of a `LIKE` predicate is not included in the B-tree indexed columns to be used for a search, that `LIKE` predicate is excluded from the count.

REG_SIZE: Management area of the datetime information acquisition function and the user information acquisition function

Use the formula shown below to determine its value. Note that the value is 0 if neither a datetime information acquisition function nor a user information acquisition function are specified. To calculate the data length, see Table 6-9: Data length of each data type.

Formula (bytes)

```
REG_SIZE = 24 × date_spec + 32 × time_spec + 32 × timestmp_spec + 120 × user_spe
c + 24
```

- *date_spec*

  A value of 1 if `CURRENT_DATE` is specified; 0 otherwise

- *time_spec*

  A value of 1 if `CURRENT_TIME` is specified; 0 otherwise

- *timestmp_spec*

  A value of 1 if `CURRENT_TIMESTAMP` is specified; 0 otherwise

- *user_spec*

  A value of 1 if `CURRENT_USER` is specified; 0 otherwise

LMTWRK_SIZE

Work table control area

The value is 0 if a `LIMIT` clause is not specified or no work table is created.

Formula (bytes)

```
LMTWRK_SIZE =
  ↑(96 × ↑(offset_num + limit_num) ÷ ↓(wrk_page_size - 64) ÷ ROWSZ↓↑
  + (4 + ROWSZ) × (offset_num + limit_num)) ÷ 1,024↑
```

- *offset_num*

  Offset of the first row to return (*offset*) as specified in a `LIMIT` clause

- *limit_num*

  Maximum number of rows to return (*row-count*) as specified in a `LIMIT` clause

- *wrk_page_size*

  Page size of the work table DB area (bytes)

  See Table 6-3: DB area page size in (2) Determining the global buffer page requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

- *ROWSZ*

  Row length of a work table (bytes)

  Determine the row length of the work table based on the formula for the variable *ROWSZ* in 5.9.2 Determining the number of pages for base rows that are needed for storing work tables. However, when the variable $col\_size_{(i)}$ needs to be determined and the column is of the variable-length data type, substitute *n* (definition length) for *d* (actual data length).

COR_QUERY_SIZE

Work area for subqueries that contain external reference columns

Substitute 16,777,216 bytes. If no subquery containing external reference columns is specified, assign 0.

TXT_WORK_SIZE

Work area for text indexes

Use the following formula to determine its value. If no text index is defined or used, assign 0.

Formula (bytes)

$$TXT\_WORK\_SIZE = 28 \times scan\_str\_num$$

- *scan_str_num*

  Number of characters in the pattern character string specified in the `LIKE` predicate that uses a text index

CSVREAD_WORK_SIZE

Retrieval work area for the `ADB_CSVREAD` function

Determine this value from the following formula. If the `ADB_CSVREAD` function will not be used, assign `0`.

Formula (bytes)

$$CSVREAD\_WORK\_SIZE = 269{,}280 + decomp\_areasize + 32 \times col\_num + \sum_{i=1}^{col\_num} temp\_coldata\_areasize_{(i)}$$

decomp_areasize

Decompression area size (bytes)

Size of area for decompressing input data files. The value to be substituted differs according to the value specified for the compression format option `COMPRESSION_FORMAT` of the `ADB_CSVREAD` function.

To determine the decompression area size, see the following table.

Table 6-5: List of the values that can be specified as the decompression area size

No.	Value of COMPRESSION_FORMAT	Decompression area size (bytes)
1	`GZIP`	112
2	`NONE`	40

col_num

Number of field data items that are to be retrieved from input data files

temp_coldata_areasize(i)

Temporary storage area size (bytes)

The size of the area that is used to temporarily store the data of the i-th field retrieved from the input data file. The value of *i* corresponds to the sequence number of the column name list of the table-function derived table.

For the temporary storage area size, use the value shown in the following table.

Table 6-6: Temporary storage area size for each data type

No.	Classification	data type		Temporary storage area size (bytes)
1	Numeric data	`INTEGER`		24
2		`SMALLINT`		16
3		`DECIMAL`		48
4		`DOUBLE PRECISION`		512
5	Character string data	`CHAR`		((2 + *defined length* × 2 + 7) ÷ 8) × 8
6		`VARCHAR`		((2 + *defined length* × 2 + 7) ÷ 8) × 8
7	Datetime data	`DATE`		16
8		`TIME`		24
9		`TIMESTAMP`		40
10	Binary data	`BINARY`	Input data file in hexadecimal format	((2 + *defined length* × 2 + 7) ÷ 8) × 8
11			Input data file in binary format	((2 + *defined length* × 8 + 7) ÷ 8) × 8
12		`VARBINARY`	Input data file in hexadecimal format	((2 + *defined length* × 2 + 7) ÷ 8) × 8
13			Input data file in binary format	((2 + *defined length* × 8 + 7) ÷ 8) × 8

REGEXP_WORK_SIZE

Size of the work area that is used to evaluate regular expressions (bytes)

If the `LIKE_REGEX` predicate is not specified, substitute 0.

The formula for calculating the value of the variable *REGEXP_WORK_SIZE* differs depending on whether a zero-character repetition is included in the regular expression specified as the pattern character string for the `LIKE_REGEX` predicate.

A zero-character repetition is a specification that might refer to a zero character as a result of combining meta characters and repetition factors. For example, if you specify `*`, `?`, `{0,}`, and `{0,x}` (*x*: 1 or a larger integer) in a nested form, the specification becomes a zero-character repetition.

A specification example that becomes a zero-character repetition

```
(a*)+, (c{0,2})*, (e?){3}
```

The following shows the formula for determining the variable *REGEXP_WORK_SIZE*.

If a zero-character repetition is not included in the regular expression:

REGEXP_WORK_SIZE= （*push_instr_num*+*scan_str_num*+2）×14

If a zero-character repetition is included in the regular expression:

REGEXP_WORK_SIZE= （*push_instr_num*×*ostrrep_instr_num*+1）×（*scan_str_num*+1）×14

Important

If a zero-character repetition is included in the regular expression, a large amount of memory might be consumed. However, even if there is a specification that might refer to a zero character, a large amount of memory is not consumed unless quantifiers or repetition factors are nested.

push_instr_num

Use the following formula to determine the value.

push_instr_num = *A* + *B* + *C*

Explanation of variables
A: Total number of the following special characters in the regular expression specified as the pattern character string: `*`, `+`, `?`, and `|`
B: Total number of regular character sets
C: Number of repetitions specified for repetition factors

For details about how to determine the variable *C* (number of repetitions specified for a repetition factor), see the following table.

Table 6-7: Determining the number of repetitions specified for a repetition factor

No.	Specification of a repetition factor	Value to be substituted for the variable C	Example
1	The upper and lower limits are both specified.	Value obtained by upper-limit - lower-limit	If `{10,20}` is specified, substitute 10.
2	The following conditions are met: • Only the lower limit is specified. • The lower limit is not followed by a comma.	Lower-limit	If `{10}` is specified, substitute 10.
3	The following conditions are met: • Only the lower limit is specified. • The lower limit is followed by a comma.	Value obtained by 256 - lower-limit	If `{10,}` is specified, substitute 246.

scan_str_num

Number of characters in the character string specified as the pattern character string

ostrrep_instr_num

Total value of zero-character repetitions

The following shows examples of regular expressions and formulas for determining the variable *REGEXP_WORK_SIZE*.

Regular expression specification example 1

```
KFAA11[0-9]{3}-E
```

Formula example (bytes)

REGEXP_WORK_SIZE = (4 + 16 + 2) x 14 = 308

Explanation
- 4 is substituted for *push_instr_num*.
- 16 is substituted for *scan_str_num*.

Regular expression specification example 2

```
KFAA6000[2-9]-E.*uid=12345.*
```

Formula example (bytes)

REGEXP_WORK_SIZE = (4 + 28 + 2) x 14 = 476

Explanation
- 3 is substituted for *push_instr_num*.
- 28 is substituted for *scan_str_num*.

RECURSIVE_QUERY_CNCT_SIZE

Recursive query management area (bytes)

If the SQL statements will not contain recursive queries, specify 0.

$$RECURSIVE\_QUERY\_CNCT\_SIZE =$$
$$(512 + 72 \times sql\_rthd\_num) \times recursive\_query\_num$$
$$+ (recurmb\_query\_num + recurmb\_setop\_num + recurmb\_list\_num + recurmb\_subquery\_num)$$
$$\times 32 + (392 + 16 \times sql\_rthd\_num) \times recurmb\_hashtbl\_num$$

sql_rthd_num

The maximum number of SQL processing real threads (the value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition).

recursive_query_num

Number of recursive queries specified in SQL statements

recurmb_query_num

Number of `FROM` clauses specified in recursive members

recurmb_setop_num

Number of set operators specified in recursive members

recurmb_list_num

Sum total of all the following specified in recursive members

- Number of times the window function is specified
- Number of query specifications in which multiple table references are specified in the `FROM` clause
- Number of joined tables specified
- Number of subqueries specified

recurmb_subquery_num

Number of subqueries specified in recursive members

recurmb_hashtbl_num

Number of hash tables specified in recursive members

For details, see *Work tables created when SQL statements are executed* in the *HADB Application Development Guide*.

RANDOMCURSOR_WORK_SIZE

Work area for scalar function RANDOMCURSOR

Substitute the following value if scalar function RANDOMCURSOR is included in the SQL statement. Substitute 0 if scalar function RANDOMCURSOR is not included in the SQL statement.

Value (bytes)

```
RANDOMCURSOR_WORK_SIZE = 3,176
```

RANDOM_NUM_WORK_SIZE

Work area for generating pseudorandom numbers (bytes)

```
RANDOM_NUM_WORK_SIZE = RANDOM_NUM_UNIFORM_WORK_SIZE
                     + RANDOM_NUM_NORMAL_WORK_SIZE
                     + RANDOMROW_WORK_SIZE
```

RANDOM_NUM_UNIFORM_WORK_SIZE

Work area for generating pseudorandom numbers according to the uniform distribution

If the SQL statement includes a RANDOM scalar function, substitute the following value. If the SQL statement does not include a RANDOM scalar function, substitute 0.

Value (bytes)

```
RANDOM_NUM_UNIFORM_WORK_SIZE = 3,176
```

RANDOM_NUM_NORMAL_WORK_SIZE

Work area for generating pseudorandom numbers according to the normal distribution

If the SQL statement includes a RANDOM_NORMAL scalar function, substitute the following value. If the SQL statement does not include a RANDOM_NORMAL scalar function, substitute 0.

Value (bytes)

```
RANDOM_NUM_NORMAL_WORK_SIZE = 6,288
```

RANDOMROW_WORK_SIZE

Work area for scalar function RANDOMROW

Substitute the following value if scalar function RANDOMROW is included in the SQL statement. Substitute 0 if scalar function RANDOMROW is not included in the SQL statement.

Value (bytes)

```
RANDOMROW_WORK_SIZE = 3,176
```

PROC_THD_SIZE: SQL process real thread control area

Use the formula shown below to determine its value. For variables *RESULT_WORK_SIZE*, *OPE_WORK_SIZE*, *HASHGRP_SIZE*, *COR_QUERY_SIZE*, *REGEXP_WORK_SIZE*, and *RANDOM_NUM_WORK_SIZE*, substitute the values determined from the formulas described earlier.

Formula (kilobytes)

$$
\begin{aligned}
&PROC\_THD\_SIZE = \\
&\uparrow \Big\{ 3{,}456 + tbl\_num \\
&\quad \times \left(400{,}400 + 672 \times uthd\_num + 256 \times tbl\_num + idx\_col\_num \times (26{,}624 + 2{,}048 \times idx\_size) \right) \\
&\quad + 40 \times sql\_rthd\_num + 544 \times uthd\_num + RESULT\_WORK\_SIZE \\
&\quad \times \Big(1{,}024 \times tbl\_num + (tbl\_num + 2 \times setop\_num) \times 8 \times \mathrm{MAX}(128,\ uthd\_num) \\
&\quad \times (cor\_query\_num \times uthd\_num + 2) + 4{,}096 \times (drvtbl\_num + in\_query\_num) + 256 \times tblfunc\_num + 11 \Big) \\
&\quad + OPE\_WORK\_SIZE \times (8 \times uthd\_num) \\
&\quad + \Sigma\_[i=cor\_query](HASHGRP\_SIZE_{(i)}) \times uthd\_num \times sql\_rthd\_num \\
&\quad + \Sigma\_[j=not\_cor\_query](HASHGRP\_SIZE_{(j)}) \\
&\quad + HASHTBL\_PROC\_SIZE + (SCAN\_WORK\_SIZE + 88) \times uthd\_num \times 5 \times tbl\_num \\
&\quad + \Big(3{,}104 + sql\_rthd\_num \times \left(2{,}208 + tbl\_num \times (96 + 192 \times sql\_rthd\_num) \right) \\
&\quad + max\_cor\_query(inprd\_cor\_query\_num) \times (96 + sql\_rthd\_num \times 48) \\
&\quad + 200 \times max\_cor\_query(setop\_cor\_query\_num) + 160 \times max\_cor\_query(window\_cor\_query\_num) \\
&\quad + 16 \times max\_cor\_query(limit\_cor\_query\_num) + 208 \times sql\_rthd\_num \Big) \\
&\quad \times cor\_query\_num + \Big((786{,}432 + 163{,}840 \times sql\_rthd\_num) \times (drvtbl\_num + in\_query\_num) \Big) \\
&\quad + \sum (drvtbl\_row\_len \times \mathrm{MIN}(drvtbl\_row\_num\ ,\ 1{,}044{,}480)) \\
&\quad + \sum (in\_query\_drvc\_len \times \mathrm{MIN}(in\_query\_drvc\_num\ ,\ 1{,}044{,}480)) \\
&\quad + 120 \times 8 \times uthd\_num \times leftouter\_num + \sum (LMTWRK\_SIZE) + COR\_QUERY\_SIZE \\
&\quad + TXT\_WORK\_SIZE + (536 + sql\_rthd\_num \times 40) \times tblfunc\_num + REGEXP\_WORK\_SIZE \times uthd\_num \\
&\quad + \sum (tblfunc\_filename\_len + 5) \\
&\quad + \sum \Big((tblfunc\_row\_len + 4) \times \mathrm{MIN}(tblfunc\_row\_num\ ,\ 65{,}025) \Big) + \sum (CSVREAD\_WORK\_SIZE) \\
&\quad + \sum (AUDITREAD\_WORK\_SIZE) + SCAN\_INFO\_SIZE \\
&\quad + \Big(8 \times query\_num \times \mathrm{MIN}(1\ ,\ \mathrm{MAX}(tvc\_num\ ,\ in\_query\_num)) + 320 \times tvc\_num + 256 \times in\_query\_num \Big) \\
&\quad \times uthd\_num + RANDOM\_NUM\_WORK\_SIZE \Big\} \div 1{,}024 \uparrow
\end{aligned}
$$

Important

- Notes when specifying viewed tables and query names

  When specifying a viewed table or query name in an SQL statement, estimate the memory requirements as if an internal derived table corresponding to each viewed table or query name were applied. For a recursive query name that references a target recursive query within a recursive query, estimate memory requirements assuming the work table that stores the results of the recursive query.

- Notes when specifying an archivable multi-chunk table

  When specifying an archivable multi-chunk table in an SQL statement, estimate memory requirements as if the archivable multi-chunk table were converted by equivalent exchange to a derived table. For details about the equivalent exchange of archivable multi-chunk tables, see *Equivalent exchange of SQL statements that search archivable multi-chunk tables* in the *HADB Application Development Guide*.

tbl_num: Number of tables specified

If you specify `FULL OUTER JOIN`, add the number of times `FULL OUTER JOIN` is specified.

sql_rthd_num

Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

idx_col_num

Number of B-tree-indexed columns and text-indexed columns to be used for a search

idx_size

Sum of the defined lengths of all B-tree-indexed columns and text-indexed columns that will be used

uthd_num

Value specified for the `adb_sys_uthd_num` operand in the server definition

Σ_[i =cor_query](HASHGRP_SIZE(i)):

Sum total of the *HASHGRP_SIZE* values calculated for all subqueries that contain external reference columns in SQL statements (bytes)

Σ_[j =not_cor_query](HASHGRP_SIZE(j)):

Sum total of the *HASHGRP_SIZE* values calculated for query specifications that are not subqueries containing external reference columns in SQL statements (bytes)

HASHTBL_PROC_SIZE: Real thread management area during retrieval using hash tables

Determine this value if the SQL statement contains any of the items listed below. Note that the value is 0 if the SQL statement contains none of these items. The value is also 0 when the maximum number of SQL processing real threads or the hash table area size is 0.

- Query specification that uses the equal sign (=) to join multiple tables
- Subquery
- `GROUP BY` clause
- `DISTINCT` set function
- `SELECT DISTINCT`
- `UNION` or `UNION DISTINCT`

Formula (bytes)

$$HASHTBL\_PROC\_SIZE = \left\lceil \frac{hashtbl\_byte}{sql\_rthd\_num} \right\rceil + 128 \times uthd\_num + (RESULT\_WORK\_SIZE + 144) \times clt\_wrk\_page\_num + (3 \times RESULT\_WORK\_SIZE + 13{,}488 + 32 \times sql\_rthd\_num) \times hashtbl\_num + HASHFLT\_PROC\_SIZE + HASHRNG\_PROC\_SIZE$$

- *hashtbl_byte*

  Value determined from the following formula:

  $$value\ specified\ for\ the\ adb\_sql\_exe\_hashtbl\_area\_size\ operand\ in\ the\ server\ definition \times 1{,}024 \times 1{,}024$$

- *sql_rthd_num*

  Value specified for the `adb_sql_exe_max_rthd_num` operand in the client definition

- *uthd_num*

  Value specified for the `adb_sys_uthd_num` operand in the server definition

- *clt_wrk_page_num*

Value specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the client definition (Value specified for the export option adb_export_wrktbl_blk_num when the adbexport command is to be executed)

- *hashtbl_num*

  Number of hash tables

  For details, see 6.25.3 Number of work tables created during retrieval using hash tables.

HASHFLT_PROC_SIZE: Hash filter management area for the SQL processing real thread

Determine this value if either of the following items is included in the SQL statement. If neither of the items is included, this value is 0. This value is also 0 if the size of the hash filter area (value specified for the adb_sql_exe_hashflt_area_size operand in the server definition) is 0.

- Query specification that uses the equal sign (=) to join multiple tables
- Subquery

Formula (bytes)

```
HASHFLT_PROC_SIZE = ↑hashflt_byte ÷ sql_rthd_num↑ + 136 + 192 × hashflt_num
```

- *hashflt_byte*

  Value determined from the following formula:

  ```
  Value-specified-for-adb_sql_exe_hashflt_area_size-in-server-definition × 1,024 × 1,024
  ```

- *sql_rthd_num*

  Value specified for the adb_sql_exe_max_rthd_num operand in the client definition

- *hashflt_num*

  Total number of hash filters in the SQL statement

  Determine the total of all the following values:

  • Number of join conditions that use = specifications in query specifications that join multiple tables by using equal signs (=)

  • Number of subqueries that do not include any external reference columns

  • Number of = conditions that include external reference columns in subqueries that include external reference columns

HASHRNG_PROC_SIZE: Management area for evaluation of range index conditions for the SQL processing real thread in hash retrieval

Determine this value if either of the following items is included in the SQL statement. If neither of the items is included, this value is 0.

- Query specification that uses the equal sign (=) to join multiple tables
- Subquery

Formula (bytes)

```
HASHRNG_PROC_SIZE = 64 + 72 × hash_retrieval_rngidx_num + 64 × hash_cond_rngidx_num
                  + Σ(↑(hash_cond_rngidx_len) ÷ 16↑ × 32)
```

hash_retrieval_rngidx_num

Total number of hash retrieval processes that use a range index in the SQL statement

Determine the total of all the following values. If this total number is 0, the value of *HASHRNG_PROC_SIZE* is 0.

- The value determined from the following formula for the query specifications in which a range-indexed column is specified for the equal sign (=), of the query specifications that use the equal sign (=) to join multiple tables:

  ```
  total-number-of-tables-joined-by-using-equal-sign - total-number-of-query-specifications-joined-by-using-equal-sign
  ```

- Number of quantified predicates for which a range-indexed column is specified on the left side
- Number of `IN` predicates for which a range-indexed column is specified on the left side and a table subquery is specified on the right side
- Number of subqueries (other than the preceding two subqueries) that satisfy all of the following conditions:
  - An external reference column is specified by using the equal sign (=) in a search condition
  - The specified external reference column is a range-indexed column

hash_cond_rngidx_num

Total number of conditions that use a range index for the hash retrieval processes in the SQL statement

Determine the total of all the following values:

- Number of = join conditions in which range-indexed columns are specified in the query specifications that use the equal sign (=) to join multiple tables
- Number of quantified predicates for which a range-indexed column is specified on the left side
- Number of `IN` predicates for which a range-indexed column is specified on the left side and a table subquery is specified on the right side
- Number of = conditions (other than the preceding two subqueries) that satisfy all of the following conditions:
  - An external reference column is specified by using the equal sign (=) in a search condition
  - The specified external reference column is a range-indexed column

Σ(↑(*hash_cond_rngidx_len*) ÷ 16↑ × 32)

Determine the value for each condition that uses a range index for the hash retrieval processes in the SQL statement, and then sum the determined values.

hash_cond_rngidx_len

Determine the following value for each condition that uses a range index for hash retrieval:

- The larger of the data lengths of value expressions specified on both sides of the = join condition for which range-indexed columns are specified in the query specifications that use the equal sign (=) to join multiple tables
- The larger of the following values for the quantified predicate for which a range-indexed column is specified on the left side:
  - Total value of data lengths of results of selection expressions for the table subquery specified on the right side
  - Column length of the range-index column specified on the left side
- The larger of the following values for the `IN` predicates for which a range-indexed column is specified on the left side and a table subquery is specified on the right side:
  - Total value of data lengths of results of selection expressions for the table subquery specified on the right side
  - Column length of the range-index column specified on the left side

- The larger of the data lengths of results of value expressions on both sides of the = condition specified in the search condition in the subquery (that is not any of the preceding two subqueries and satisfies the following two conditions):

  • An external reference column is specified by using the equal sign (=) in the search condition in the subquery

  • The specified external reference column is a range-indexed column

setop_num: Number of set operators specified in SQL statements

If you specify `FULL OUTER JOIN`, add the number of times `FULL OUTER JOIN` is specified.

drvtbl_num: Number of derived tables specified in SQL statements

If you specify `FULL OUTER JOIN`, add the number of times `FULL OUTER JOIN` is specified.

in_query_num: Sum total of the number of `IN` predicates and quantified predicates that satisfy the following conditions:

- `IN` predicate

  A table subquery is specified on the right side, and the column specified on the left-side value expression is the first of the B-tree indexed columns to be used for a search.

- Quantified predicate

  The column specified on the left-side value expression in = `ANY` or = `SOME` is the first of the B-tree indexed columns to be used for a search.

inprd_cor_query_num:

Sum total of the number of `IN` predicates and quantified predicates that are specified in subqueries that contain external reference columns, and that satisfy the condition specified in the variable *in_query_num*.

max_cor_query(X):

For each subquery that contains external reference columns in an SQL statement, substitute the maximum value of *X* specified in the subquery.

setop_cor_query_num:

Number of set operators specified in subqueries that contain external reference columns

window_cor_query_num: Number of window functions specified in subqueries that contain external reference columns

In subqueries that contain external reference columns, if multiple `DISTINCT` set functions are specified with different arguments in one query specification, also add the number of those set functions - 1.

limit_cor_query_num:

Number of `LIMIT` clauses specified in subqueries that contain external reference columns

cor_query_num:

Sum total of the number of subqueries that contain external reference columns and the number of subqueries contained in those subqueries

Σ(*drvtbl_row_len* × MIN(*drvtbl_row_num*, 1,044,480)):

If you specify derived tables, determine the value of the expression *drvtbl_row_len* × MIN(*drvtbl_row_num*, 1,044,480) for each derived table. Then, add up the determined values.

- *drvtbl_row_len*

  Substitute the value obtained by adding up the data lengths of the selection expressions of the derived tables.

  To calculate the data length, see Table 6-9: Data length of each data type.

- *drvtbl_row_num*

Substitute the number of rows in the derived table. If you cannot determine the number of rows, substitute 1,044,480.

Σ(*in_query_drvc_len* × MIN(*in_query_drvc_num*, 1,044,480)):

Determine the value of the expression *in_query_drvc_len* × MIN(*in_query_drvc_num*, 1,044,480) for each predicate targeted by the variable *in_query_num*. Then, add up the determined values.

- *in_query_drvc_len*

  Substitute the data length of the selection expression in the subquery of the predicate targeted by the variable *in_query_num*.

  To calculate the data length, see Table 6-9: Data length of each data type.

- *in_query_drvc_num*

  Substitute the number of rows in the subquery of the predicate targeted by the variable *in_query_num*. If you cannot determine the number of rows, substitute 1,044,480.

leftouter_num

Number of times `LEFT OUTER JOIN` is specified

Σ(*LMTWRK_SIZE*):

Sum total of the *LMTWRK_SIZE* values calculated for the individual work tables that are created (bytes)

For the following issues, see *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*:

- Whether an SQL statement in which the `LIMIT` clause is specified creates a work table
- Number of work tables that are created

If the `LIMIT` clause is used to specify the maximum number of rows to be acquired from the results of a set operation, the value specified for the `LIMIT` clause is applied to each of the query specifications that comprise the set operation.

tblfunc_num

Number of table function derived tables

Σ(*tblfunc_filename_len* + 5)

Determine the value of the expression *tblfunc_filename_len* + 5 for each table function derived table. Then, add up the determined values.

tblfunc_filename_len

Length of the file name (full path) of the file to be accessed

Σ((*tblfunc_row_len* + 4) × MIN(*tblfunc_row_num*, 65,025))

Determine the value of the expression (*tblfunc_row_len* +4) × MIN(*tblfunc_row_num*, 65,025) for each table function derived table. Then, add up the determined values.

tblfunc_row_len

Sum of the data lengths for all columns in table function derived tables

To calculate the data length, see Table 6-9: Data length of each data type.

tblfunc_row_num

Number of rows in table function derived tables

If you cannot determine the number of rows, substitute 65,025.

Σ(*CSVREAD_WORK_SIZE*)

Determine in bytes the value of the *CSVREAD_WORK_SIZE* variable for each table function derived table for which the `ADB_CSVREAD` function is specified. Then, add up the determined values.

Σ(*AUDITREAD_WORK_SIZE*)

Determine in bytes the value of the *AUDITREAD_WORK_SIZE* variable for each table function derived table for which the `ADB_AUDITREAD` function is specified. Then, add up the determined values.

query_num

Number of queries

Determine this value from the number of `FROM` clauses specified in SQL statements.

If you specify `FULL OUTER JOIN`, add the number obtained by doubling the number of times `FULL OUTER JOIN` is specified.

tvc_num

Number of table value constructors specified in SQL statements

SCAN_WORK_SIZE: Search work area

Use the following formula to determine its value.

Formula (bytes)

```
SCAN_WORK_SIZE = MAX(SCB_SIZE) + 24
```

MAX(SCB_SIZE): The maximum size of the work area for internal retrieval processing

Determine the value of the *SCB_SIZE* variable for each table to be searched. Then, substitute the largest of the determined values.

SCB_SIZE: Work area for internal retrieval processing

Use the following formula to determine the value:

Formula (bytes)

```
SCB_SIZE =
    472+MAX(256, ↑KEYSZ÷8↑×8)+64×idx_col_num
    +↑(KEYSZ×2+10×logical_num)÷8↑×8+56×(scan_str_num+1)
    +64+(txt_nest_num+txt_contains_str)×8+CS_WORK_SIZE
```

KEYSZ: Key length of the B-tree index to be used for searching

For details, see 5.8.4 Determining the key length (KEYSZ) of a B-tree index.

idx_col_num

Number of B-tree indexed columns to be used for a search

logical_num

Maximum number of logical operators in a search condition

scan_str_num

Substitute either of the following values:

- Number of characters in the pattern character string specified in the `LIKE` predicate that uses a text index
- Number of characters in the search string specified for the scalar function `CONTAINS`

  Note that if you will perform a word-based leading-match word search with the `WORDCONTEXT_PREFIX` word-context search specification specified in a `CONTAINS` scalar function, substitute a value equivalent to the number of words specified in the search string × 32.

If the pattern character string consists of one character or more than 1,000 characters, assign 1,000.

txt_nest_num

Total of the following numbers of nests:#

- Number of nested `OR` conditions in the `LIKE` predicate that uses a text index

- Number of nested `OR` conditions in the scalar function `CONTAINS`

txt_contains_str

Number of characters in the search string specified for the scalar function `CONTAINS`#

#

The maximum of the total of the values specified for the variables *txt_nest_num* and *txt_contains_str* is 1,001. If the total value exceeds 1,001, substitute 1,001.

CS_WORK_SIZE: Work area for retrieval from column store tables

Use the following formula to determine the value: Note that the value is 0 if data will not be retrieved from any column store tables.

Formula (bytes)

```
CS_WORK_SIZE =
    2,544 + 144 × (MAX(col_num , 3) + in_projcol_num)
    + 29 × tbl_col_num × MIN (in_projcol_num , 1) + ↑ tbl_col_num ÷ 8 ↑ × 2
    + 6 × tbl_col_num + 27 × SEGSIZE + (page_size − 56) × ↑ 16,384 ÷ page_size ↑
    + 65,568 × pred_num + branch_col_size + Σ (DECOMPRESS_WORK_SIZE)
    + ↑ 10 × grp_col_num ÷ 8 ↑ × 8 + Σ ( ↑ grp_col_size ÷ 8 ↑ × 8)
    + GROUPING_IN_SGMT_WORK_SIZE
```

col_num

Number of selection expressions

in_projcol_num

The number of column names specified in the value expression on the right side of the `IN` predicate

If no `IN` predicate is specified or there are no column names specified in the value expression on the right side of the `IN` predicate, substitute `0`.

tbl_col_num

Number of columns in the table

SEGSIZE

Segment size in the DB area in which DB area files are defined (pages)

Use the following formula to determine the segment size:

```
SEGSIZE = 4,194,304 ÷ page_size
```

- *page_size*

  Page size of the DB area defined for the table (bytes)

  See Table 6-3: DB area page size in (2) Determining the global buffer page requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

pred_num

Number of predicates

If no search conditions are specified, substitute `1`.

branch_col_size

Maximum definition length of columns whose data type is character string data or binary data (bytes)

If there is a column with a definition length of 128 or longer among those defined in selection expressions or search conditions, substitute the maximum definition length.

Σ(*DECOMPRESS_WORK_SIZE*)

Determine the value of the variable *DECOMPRESS_WORK_SIZE* for each column specified in selection expressions and search conditions. Then, add up the determined values.

grp_col_num

Number of columns specified in the `GROUP BY` clause

Σ(↑*grp_col_size* ÷ 8↑ × 8)

Determine the value of the expression ↑*grp_col_size* ÷ 8↑ × 8 for each column specified in the `GROUP BY` clause, and then add up the determined values.

grp_col_size

Data length of each column specified in the `GROUP BY` clause

To calculate the data length, see Table 6-9: Data length of each data type.

GROUPING_IN_SGMT_WORK_SIZE

Grouping work area in the segment

Use the following formula to determine this value. If the `GROUP BY` clause is not specified, substitute 0.

Formula (bytes)

$$GROUPING\_IN\_SGMT\_WORK\_SIZE = (grp\_col\_num + set\_func\_num) \times 258 + \sum_{i=1}^{grp\_col\_num} (grouping\_col\_data\_size_{(i)} \times 126 + 9) + \sum_{i=1}^{set\_func\_num} ((grouping\_set\_func\_size_{(i)} + 22) \times 126) + 263{,}305$$

grp_col_num

Number of columns specified in the `GROUP BY` clause

set_func_num

Number of set functions

grouping_col_data_size(i)

Data length of each column

Determine the data length from the following table.

Table 6-8: Value to be substituted for grouping_col_data_size

No.	Classification	data type		Data length (bytes)
1	Numeric data	`INTEGER`		8
2		`SMALLINT`		
3		`DECIMAL`	1 ≤ *precision* ≤ 4	8
4			5 ≤ *precision* ≤ 8	
5			9 ≤ *precision* ≤ 16	
6			17 ≤ *precision* ≤ 38	16
7		`DOUBLE PRECISION`		8
8	Character string data	`CHAR`		*definition-length* + 7
9		`VARCHAR`	1 ≤ *definition-length* ≤ 32,000	*definition-length* + 9
10			32,001 ≤ *definition-length* ≤ 64,000	*definition-length* + 11

No.	Classification	data type	Data length (bytes)
11	Datetime data	DATE	8
12		TIME(*p*)	3 + ↑*p* ÷ 2↑ + 7
13		TIMESTAMP(*p*)	7 + ↑*p* ÷ 2↑ + 7
14	Binary data	BINARY	*definition-length* + 7
15		VARBINARY	*definition-length* + 9

Legend:

p: Number of digits in the fractional seconds (0, 3, 6, 9 or 12. The default value is 0.)

grouping_set_func_size(i)

Data length of each set function result

Determine the data length of each set function result as explained in Table 6-8: Value to be substituted for grouping_col_data_size according to the data type of the set function result.

DECOMPRESS_WORK_SIZE

Decompression work area

Use the following formula to determine the value:

Formula (bytes)

$$DECOMPRESS\_WORK\_SIZE = \uparrow \frac{col\_data\_size(i)+1}{8} \uparrow \times 16$$

col_data_size(i)

Data length of each column

To calculate the data length, see Table 6-9: Data length of each data type.

TXT_WORK_SIZE

Work area for text indexes

Use the following formula to determine its value. If no text index is defined or used, assign 0.

Formula (bytes)

$$TXT\_WORK\_SIZE = 28 \times scan\_str\_num + (txt\_nest\_num \times 5 + 2) \times 40 + 32{,}768$$

AUDITREAD_WORK_SIZE

Retrieval work area for the ADB_AUDITREAD function (bytes)

Use the following formula to determine its value. If the ADB_AUDITREAD function will not be used, assign 0.

```
AUDITREAD_WORK_SIZE = (AUDREADSZ + AUDBLOCKSZ) × 1,024
```

AUDREADSZ

Audit trail file management area

Substitute the following value. If the ADB_AUDITREAD function will not be used, assign 0.

Value (kilobytes)

```
AUDREADSZ = 1
```

AUDBLOCKSZ

Audit trail file work area

Substitute the following value. If the ADB_AUDITREAD function will not be used, assign 0.

Value (kilobytes)

```
AUDBLOCKSZ = 262,144
```

SCAN_INFO_SIZE: Table retrieval management area (bytes)

```
SCAN_INFO_SIZE = Σ SCAN_INFO_DATA_SIZE + Σ SCAN_INFO_HASH_SIZE
```

Σ *SCAN_INFO_DATA_SIZE*

Determine the value of the *SCAN_INFO_DATA_SIZE* variable for each retrieval-target table, and then add up the determined values.

Σ *SCAN_INFO_HASH_SIZE*

If the results derived from hash joins correspond to the outermost table of a nested loop join, calculate the value of the *SCAN_INFO_HASH_SIZE* variable for each hash join. If the results derived from hash execution of a subquery correspond to the outermost table of a nested loop join, calculate the value of the *SCAN_INFO_HASH_SIZE* variable for each hash execution of the subquery. Specify the total of the calculated values.

SCAN_INFO_DATA_SIZE: Table retrieval management area for base tables and joined tables (bytes)

```
SCAN_INFO_DATA_SIZE = tbl_access_num × (TBL_ACCESS_INFO_SIZE + OUTERJOIN_INFO_SIZE
)
```

tbl_access_num: Number of table retrievals

Substitute whichever of the following values applies:

- If the table is not joined: 1
- If the table is joined by a nested loop join
  - If the table is the outermost of the nested loop join: 1
  - If the table is an inner table of the nested loop join: The number of rows in the outer table of that inner table
- If the table is joined by a hash join
  - If the table is the outer table of the hash join: 1
  - If the table is the inner table of the hash join: 1

If the table is specified in a subquery that includes an external reference column, and the processing method of that subquery is nested loops work table execution or nested loops row value execution, specify the preceding value multiplied by *uthd_num* × *sql_rthd_num* as the number of table retrievals.

uthd_num

Value specified for the adb_sys_uthd_num operand in the server definition

sql_rthd_num

Value specified for the adb_sql_exe_max_rthd_num operand in the server definition

Example of calculating number of table retrievals

```
SELECT * FROM "T1" A,"T2" B,"T3" C
           WHERE A."C1"=B."C1" AND B."C2"=C."C2"
```

When executing this SELECT statement where tables T1, T2, and T3 are joined by a nested loop join in that order from the outside in, the table retrieval counts are as follows:

- Retrieval count of table T1: 1
- Retrieval count of table T2: The number of lines in the results obtained by searching table T1

- Retrieval count of table T3: The number of lines in the results obtained by searching tables T1 and T2

TBL_ACCESS_INFO_SIZE: Table retrieval information area

Substitute whichever of the following values applies:

- If the retrieval method of the target table is table scan: *TBL_SCI_MEM*
- If the retrieval method of the target table is index scan using a B-tree index or key scan: *IDX_SCI_MEM*
- If the retrieval method of the target table is index scan using a text index: *IDX_SCI_TXT_MEM*
- If the retrieval method of the target table is work table scan: *WRKTBL_SCI_MEM*

TBL_SCI_MEM: The table retrieval information area during table scan (bytes)

```
TBL_SCI_MEM = (336 + RESULT_WORK_SIZE) × (dbarea_file_num + chunk_num + SGTBL - 2)
```

dbarea_file_num

Number of DB area files for the DB areas that store the table to be searched

chunk_num

Number of chunks in table to be searched (or 1 if the table to be searched is a single-chunk table)

SGTBL

Number of segments in table to be searched

For the formula for determining the value of *SGTBL*, see the following subsection in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the retrieval-target table.

- Single-chunk table that is a row store table
  (a) Determining the variable SGROWTBL (for a single-chunk table)
- Single-chunk table that is a column store table
  (b) Determining the variable SGCOLUMNTBL (for a single-chunk table)
- Multi-chunk table that is a row store table
  (f) Determining the variable SGROWTBL (for a multi-chunk table)
- Multi-chunk table that is a column store table
  (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table)

IDX_SCI_MEM: Table retrieval information area for retrieval using a B-tree index (bytes)

```
IDX_SCI_MEM = IDX_CHUNK_SCI_MEM + IDX_SCAN_IDX_MEM
```

IDX_CHUNK_SCI_MEM

Table retrieval information area for chunks during retrieval using a B-tree index (bytes)

```
IDX_CHUNK_SCI_MEM = 368 + RESULT_WORK_SIZE
```

Note that *IDX_CHUNK_SCI_MEM* = 0 if either of the following conditions are met:

- The table to be searched is a multi-chunk table consisting of one chunk
- The table to be searched is a single-chunk table

IDX_SCAN_IDX_MEM

Table retrieval information area for all chunks during retrieval using a B-tree index (bytes)

$$IDX\_SCAN\_IDX\_MEM = \sum_{s=1}^{chunk\_num} IDX\_SCAN\_IDX\_MEM(s)$$

Note that *IDX_SCAN_IDX_MEM* = 0 if the number of chunks to be searched is 0.

chunk_num

Number of chunks to be searched

IDX_SCAN_IDX_MEM(s)

Table retrieval information area for B-tree index of s-th chunk (bytes)

- When there are two or more rows# in the *s*-th chunk that satisfy the search condition

$$IDX\_SCAN\_IDX\_MEM(s) = PIDX\_MEM \times \left\{ r + \sum_{k=1}^{n} PIDX(k) \right\} + PDUP\_MEM \times PDUP$$

- When there is one row# or fewer in the *s*-th chunk that satisfies the search condition

$$IDX\_SCAN\_IDX\_MEM(s) = 0$$

#: When calculating the number of rows that satisfy the search condition, include row data that was invalidated by the deletion of rows by a `DELETE` statement or the update of rows by an `UPDATE` statement.

r

2 if there are two or more types of key values of rows that satisfy the search condition, and 0 if there is one type.

n

The minimum value of k where *PIDX(k)* = 1.

PIDX(k)

Determine this value by using *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

For the variables *key_num*, *dup_key_num*, and *key_dup* used in the formula for determining *PIDX(k)*, determine the average number of key types and average number of duplicate keys for keys that satisfy the search conditions. When calculating the number of key types and duplicate keys, include invalid row data for rows that are deleted by a `DELETE` statement or updated by an `UPDATE` statement. If rows are being repeatedly added by `INSERT` statements or repeatedly modified by `UPDATE` statements, or there is variation in the number of duplicate keys or the key length, use a value that is 1.2 to 1.5 times the value determined by the formula.

PDUP

Determine this value by using *Formula 2 (for determining PDUP)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

For the variables *dup_key_num* and *dup_key_dup* used in the formula for determining *PDUP*, determine the average number of key types and average number of duplicate keys for keys that satisfy the search conditions. When calculating the number of key types and duplicate keys, include invalid row data for rows that are deleted by a `DELETE` statement or updated by an `UPDATE` statement. If rows are being repeatedly added by `INSERT` statements or repeatedly modified by `UPDATE` statements, or there is variation in the number of duplicate keys or the key length, use a value that is 1.2 to 1.5 times the value determined by the formula.

PIDX_MEM

Table retrieval information area for upper or leaf page of a B-tree index (bytes)

$$PIDX\_MEM = 432 + RESULT\_WORK\_SIZE + 24 \times (idx\_col\_num - 1) + 2 \times idx\_col\_num + 2 \times \sum_{i=1}^{idx\_col\_num} key\_size_{(i)}$$

idx_col_num

Number of indexed columns in a B-tree index

key_size(i)

Key length of i-th indexed column

Calculate *key_size(i)* as explained in 5.8.4 Determining the key length (KEYSZ) of a B-tree index.

Interpret the "Actual data length" in Table 5-18: B-tree index key length as the defined length of the data.

PDUP_MEM

Table retrieval information area for low-ID directory pages or low-ID list page of a B-tree index (bytes)

$$PDUP\_MEM = 368 + RESULT\_WORK\_SIZE + 2 \times idx\_col\_num + \sum_{i=1}^{idx\_col\_num} key\_size_{(i)}$$

idx_col_num

Number of indexed columns in a B-tree index

key_size(i)

Key length of i-th indexed column

Calculate *key_size(i)* as explained in 5.8.4 Determining the key length (KEYSZ) of a B-tree index.

Interpret the "Actual data length" in Table 5-18: B-tree index key length as the defined length of the data.

IDX_SCI_TXT_MEM

Table retrieval information area for retrieval using a text index (bytes)

```
IDX_SCI_TXT_MEM = IDX_CHUNK_TXT_MEM + IDX_SCAN_TXT_MEM
```

IDX_CHUNK_TXT_MEM

Table retrieval information area for chunks during retrieval using a text index (bytes)

$$IDX\_CHUNK\_TXT\_MEM = 328 + 8 \times \mathrm{MIN}(TXT\_SCAN\_NODE\_NUM,\ 1{,}001) + 8 \times \mathrm{MIN}(scan\_str\_num,\ 1{,}001) + RESULT\_WORK\_SIZE$$

Note that *IDX_CHUNK_TXT_MEM* = 0 if either of the following conditions are met:

- The table to be searched is a multi-chunk table consisting of one chunk
- The table to be searched is a single-chunk table

scan_str_num

Number of characters in the search character string specified in the `LIKE` predicate, `LIKE_REGEX` predicate, or `CONTAINS` scalar function that uses a text index (specify 1,001 if the number exceeds 1,001)

TXT_SCAN_NODE_NUM

Number of nodes used for text index searches

```
TXT_SCAN_NODE_NUM = scan_nest_num + sortcode_str_num + TXT_SYNONYM_NUM
```

scan_nest_num

Number of nested OR conditions in the `LIKE` predicate, `LIKE_REGEX` predicate, or `CONTAINS` scalar function that uses a text index

sortcode_str_num

Number of characters in the search string when performing correction search in a `CONTAINS` scalar function

TXT_SYNONYM_NUM

Average number of synonyms in synonym groups when using synonym search in a `CONTAINS` scalar function

$$TXT\_SYNONYM\_NUM = \uparrow all\_synonym\_num \div synonym\_group\_num \uparrow$$

all_synonym_num

The number of synonyms registered in the synonym dictionary specified when performing a synonym search in a `CONTAINS` scalar function

This value is the total number of synonyms in all synonym groups. Specify 0 if you do not perform synonym searches.

synonym_group_num

The number of synonym groups registered in the synonym dictionary specified when performing a synonym search in a `CONTAINS` scalar function

IDX_SCAN_TXT_MEM

Table retrieval information area for all chunks during retrieval using a text index (bytes)

$$IDX\_SCAN\_TXT\_MEM = \sum_{s=1}^{chunk\_num} IDX\_SCAN\_TXT\_MEM(s)$$

Note that *IDX_SCAN_TXT_MEM* = 0 if the number of chunks to be searched is 0.

chunk_num

Number of chunks to be searched

IDX_SCAN_TXT_MEM(s)

Table retrieval information area for text index of s-th chunk (bytes)

$$IDX\_SCAN\_TXT\_MEM(s) = 16 \times \left\{ \begin{array}{l} 328 + 8 \times \mathrm{MIN}(TXT\_SCAN\_NODE\_NUM\ ,\ 1{,}001) \\ +56 \times \mathrm{MIN}(scan\_str\_num\ ,\ 1{,}001) + RESULT\_WORK\_SIZE \end{array} \right\}$$

WRKTBL_SCI_MEM

Table retrieval information area for work table retrieval (bytes)

$$WRKTBL\_SCI\_MEM = (304 + RESULT\_WORK\_SIZE) \times WRKTBL\_PGNUM$$

WRKTBL_PGNUM

Number of pages of work table to be searched

Calculate *WP(i)* as explained in 5.9.2 Determining the number of pages for base rows that are needed for storing work tables.

SCAN_INFO_HASH_SIZE

Table retrieval management area when retrieval is performed using a hash table (bytes)

$$\begin{array}{l} SCAN\_INFO\_HASH\_SIZE = \\ \quad hashtbl\_row\_len \times \mathrm{MIN}(hashtbl\_row\_num\ ,\ 1{,}044{,}480) \\ \quad +(RESULT\_WORK\_SIZE + 200 + 40 \times sql\_rthd\_num) \times \uparrow \mathrm{MIN}(hashtbl\_row\_num\ ,\ 1{,}044{,}480) \div 255 \uparrow \end{array}$$

hashtbl_row_len

Hash table row length

Calculate the hash table row length as explained in 6.25.3 Number of work tables created during retrieval using hash tables.

hashtbl_row_num

Number of result rows derived from a hash join or hash execution of a subquery

If you cannot determine the number of rows, specify 1,044,480.

sql_rthd_num

The maximum number of SQL processing real threads (the value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition).

OUTERJOIN_INFO_SIZE

Outer join management area (bytes)

```
OUTERJOIN_INFO_SIZE = 120 + 16 × sql_rthd_num + RESULT_WORK_SIZE
```

Note that *OUTERJOIN_INFO_SIZE* = 0 if all of the following conditions are met:

- The table to be searched is not specified on the left side of the outer join
- The table to be searched is not specified on the right side of the outer join

sql_rthd_num

The maximum number of SQL processing real threads (the value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition).

BUF_DLYSZ

Area that is used to report the updated pages

Substitute the following value.

Value (kilobytes)

■ When the multi-node function is used and the `INSERT`, `UPDATE`, or `DELETE` statement is executed
BUF_DLYSZ = 41

■ In other cases
BUF_DLYSZ = 0

ARC_DIR_PATH

Area that is used to manage the path name of the archive directory to be deleted

Use the formula shown below to determine its value.

Formula (kilobytes)

$$ARC\_DIR\_PATH = \uparrow archivedir\_path\_num \times 513 \div 1{,}024 \uparrow$$

archivedir_path_num

If you execute the `TRUNCATE TABLE` statement, substitute 1.

If you execute the `PURGE CHUNK` statement, substitute the number of chunks to be deleted.

RNGIDX_INFO_SIZE

Range index information area for chunks

Use the following formula to determine its value.

Formula (bytes)

$$RNGIDX\_INFO\_SIZE = 32 + 8 \times rngidx\_num + \sum_{i=1}^{rngidx\_num} (16 + 8 \times (chunk\_num_{(i)} - 1) + RNGIDX\_INFO\_ENTRY\_SIZE_{(i)} \times chunk\_num_{(i)})$$

rngidx_num

Total number of range indexes that are used for retrieval in tables that are searched by using B-tree indexes, among the range indexes that are used to skip chunks during execution of an SQL statement

If one or more SQL statements are executed more than once for the same range index, assume the total number to be 1. If the total number is 0, substitute 0 for *RNGIDX_INFO_SIZE*.

Note that you can determine this value from the display result of the access path information by using the following procedure.

1. In the display result details view of the access path information, check *Information related to table retrieval methods* and *Information related to indexes*. Find all items that meet either of the following conditions:
   - The table retrieval method is `INDEX SCAN`, and `INDEX TYPE` of the first index information in *Information related to indexes* is `B-TREE` (B-tree index).
   - The table retrieval method is `KEY SCAN`.
2. Find all items that meet the condition in step 1 and for which *Information related to indexes* meets both of the following conditions:
   - `INDEX TYPE` is `RANGE` (range index).
   - `SKIP COND` is `CHUNK`.
3. Determine the total number of `INDEX NAME` values in the *Information related to indexes* that meets the conditions in step 2, eliminating duplication in names. Use the determined value for *rngidx_num*.

   In the following example of the display result of access path information, there are two `INDEX NAME` values: `T1RIX1` (appearing twice) and `T2RIX1` (appearing once). Therefore, *rngidx_num* is 2.

■ Example of display results of access path information

```
<<Detail   >>

QUERY : 1
     5 SET OPERATION
       SET OPERATION TYPE : UNION DISTINCT

QUERY : 1
     9 NESTED LOOP JOIN
       JOIN TYPE          : INNER JOIN

QUERY : 1
    10 KEY SCAN(ADBUSER.T1)
       INDEX NAME         : T1BIX1
        INDEX TYPE        : B-TREE
        INDEX COLUMN      : C1 ASC (>)
        INDEX COLUMN      : C2 ASC (none)
       INDEX NAME         : T1RIX1
        INDEX TYPE        : RANGE
        SKIP COND         : CHUNK
        INDEX COLUMN      : C1

QUERY : 1
    13 KEY SCAN(ADBUSER.T2)
       INDEX NAME         : T2BIX1
        INDEX TYPE        : B-TREE
        INDEX COLUMN      : C1 ASC (=)
        INDEX COLUMN      : C2 ASC (none)
       INDEX NAME         : T2RIX1
        INDEX TYPE        : RANGE
        SKIP COND         : CHUNK
        INDEX COLUMN      : C1

QUERY : 2
    18 TABLE SCAN(ADBUSER.T3)
       INDEX NAME         : T3RIX1
        INDEX TYPE        : RANGE
        SKIP COND         : CHUNK AND SEGMENT
        INDEX COLUMN      : C1

QUERY : 3
    25 KEY SCAN(ADBUSER.T1)
       INDEX NAME         : T1BIX1
        INDEX TYPE        : B-TREE
        INDEX COLUMN      : C1 ASC (=)
        INDEX COLUMN      : C2 ASC (=)
       INDEX NAME         : T1RIX1
        INDEX TYPE        : RANGE
        SKIP COND         : CHUNK
        INDEX COLUMN      : C1
```

Retrieval results of items for which *Information related to table retrieval methods* meets the condition in step 1

Retrieval results of items for which *Information related to indexes* meets the conditions in step 2, among the retrieval results of items meeting the condition in step 1

For details about access paths, see *How to use access paths (how to use SQL statement execution plans)* in the *HADB Application Development Guide*.

chunk_num(i)

Number of chunks created for the table in which the i-th range index is defined

Exclude the number of archive chunks. In the case of a single-chunk table, the value of this variable is 1.

RNGIDX_INFO_ENTRY_SIZE(i)

Range index information entry size

Use the following formula to determine its value.

Formula (bytes)

$$RNGIDX\_INFO\_ENTRY\_SIZE_{(i)} = 16 + \left(\frac{2 \times col\_size_{(i)} - 1}{8} + 1 \right) \times 8$$

col_size(i)

Data length of a column for which a range index is defined (bytes)

For details about the data length of each column, see Table 6-9: Data length of each data type.

Determine the data length from the following table.

Table 6-9: Data length of each data type

No.	Classification	Data type		Data length (bytes)
1	Numeric data	INTEGER		8
2		SMALLINT		4
3		DECIMAL	1 ≤ *precision* ≤ 4	2
4			5 ≤ *precision* ≤ 8	4
5			9 ≤ *precision* ≤ 16	8
6			17 ≤ *precision* ≤ 38	16
7		DOUBLE PRECISION		8
8	Character string data	CHAR		*definition-length*
9		VARCHAR	1 ≤ *definition-length* ≤ 32,000	*definition-length* + 2
10			32,001 ≤ *definition-length* ≤ 64,000	*definition-length* + 4
11	Datetime data	DATE		4
12		TIME (*p*)		3 + ↑*p* ÷ 2↑
13		TIMESTAMP (*p*)		7 + ↑*p* ÷ 2↑
14	Binary data	BINARY		*definition-length*
15		VARBINARY		*definition-length* + 2

Legend:

p: Number of digits in the fractional seconds (0, 3, 6, 9 or 12. The default value is 0.)

Note

For calculating CURRENT_USER, assume the data type is VARCHAR and the defined length is 100.

(d) Determining the variable RTHD_ROLLBKSZ

The variable *RTHD_ROLLBKSZ* is required when rollback processing is executed or when the HADB server is restarted. Substitute the following value.

Value (kilobytes)

RTHD_ROLLBKSZ = 67

(e) Determining the variable RTHD_DBEXTSZ

The variable *RTHD_DBEXTSZ* is required when a DB area is extended. Substitute the following value.

Value (kilobytes)

RTHD_DBEXTSZ = 32,768

(f) Determining the variable RTHD_WORKBUF

The variable *RTHD_WORKBUF* is required when a work table is used. Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_WORKBUF =
↑(clt_wrk_page_num × (wrk_page_size + 304) +8
× (clt_wrk_page_num + WIO_NUM) + wrk_page_size) ÷ 1,024↑ × 1.05
```

Explanation of variables

- *clt_wrk_page_num*
  Value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition (Value specified for the export option `adb_export_wrktbl_blk_num` when the `adbexport` command is to be executed)
- *wrk_page_size*
  Determine this value in bytes based on the explanation of the page size of the work table DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.
- *WIO_NUM*
  Substitute the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` client definition operand (or the value specified for the export option `adb_export_wrktbl_blk_num` when the `adbexport` command is to be executed), or the value of the variable *UTHNUM*, whichever is smaller. For details about the variable *UTHNUM*, see (a) Determining the variable SCI in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(g) Determining the variable RTHD_EXESQLDICSZ

The variable *RTHD_EXESQLDICSZ* is required when a data manipulation SQL statement is preprocessed. Use the following formula to determine its value.

Note that if the target table is a viewed table, add 39 kilobytes to the value determined by the formula.

Formula (kilobytes)

RTHD_EXESQLDICSZ = 104 + ↑（（*LIKE_REG_PRD_SIZE* + *CONTAINS_PRD_SIZE*）÷ 1,024）↑

Explanation of variables

LIKE_REG_PRD_SIZE
Execution management area for the `LIKE_REGEX` predicate (bytes)
Use the formula shown below to determine its value.
Note that if the `LIKE_REGEX` predicate is not specified, substitute 0 for the variable *LIKE_REG_PRD_SIZE*.

```
LIKE_REG_PRD_SIZE=
  ↑（scan_str_num÷50）↑×2,400
  +MAX（（5+normal_factor_size+paren_nest_num)×8 ,
         (1+vertical_line_num+quantifier_nest_num+repetition_factor_sum)×16)
  +BYTE_CODE_SIZE
```

scan_str_num
Number of characters in the character string specified as the pattern character string

normal_factor_size
Number of regular factors in the regular expression specified as the pattern character string

paren_nest_num

Number of parentheses in the regular expression specified as the pattern character string

vertical_line_num

Number of vertical bars in the regular expression specified as the pattern character string

quantifier_nest_num

Number of quantifiers in the regular expression specified as the pattern character string

repetition_factor_sum

Total of the upper limits specified for the repetition factors in the regular expression specified as the pattern character string

If no upper limits are specified for repetition factors, substitute the lower limit value specified for the repetition factors.

Regular expression specification example

```
((a{2}){5}){8,10}
```

Formula (count)

repetition_factor_sum= 2+5+10 = 17

The following shows examples of regular expressions and formulas for determining the variable *LIKE_REG_PRD_SIZE*.

Regular expression specification example 1

```
KFAA11[0-9]{3}-E
```

Formula example (bytes)

LIKE_REG_PRD_SIZE =
↑(16 / 50)↑ x 2,400 + MAX((5 + 9 + 0) x 8, (1 + 0 + 0 + 3) x 16) + 170 = 2,682

Explanation
The underlined value is determined from the formula for the variable *BYTE_CODE_SIZE*.

Regular expression specification example 2

```
KFAA6000[2-9]-E.*uid=12345.*
```

Formula example (bytes)

LIKE_REG_PRD_SIZE =
↑(28 / 50)↑ x 2,400 + MAX((5 + 22 + 0) x 8, (1 + 0 + 0 + 0) x 16) + 165 = 2,781

Explanation
The underlined value is determined from the formula for the variable *BYTE_CODE_SIZE*.

BYTE_CODE_SIZE

Area that is used to store the bytecodes resulting from converting (for evaluation) the regular expression specified as the pattern character string in the execution management area for the `LIKE_REGEX` predicate (bytes)

Use the formula shown below to determine its value.

```
BYTE_CODE_SIZE=
  char_specifier_num×5+quantifier_num×18
  +char_class_num×16+normal_char_set_num×16
  +char_list_num×34+repetition_factor_sum×9+1
```

char_specifier_num

Total number of character specifiers that are used in the regular expression specified as the pattern character string

If repetition factors are specified for the character specifiers, add the number of repetitions to the value of this variable. If there is a repetition factor in a nested form (such as `(a{10}){10}`), multiply each number of repetitions. Then, add the result of multiplication to the variable *char_specifier_num*.

For details about the number of repetitions specified for repetition factors, see Table 6-7: Determining the number of repetitions specified for a repetition factor in (c) Determining the variable RTHD_EXESQLSZ.

quantifier_num

Number of quantifiers excluding the repetition factors in the regular expression specified as the pattern character string

char_class_num

Total number of character classes in the regular expression specified as the pattern character string

If repetition factors are specified for the character classes, add the number of repetitions to the value of this variable. If there is a repetition factor in a nested form (such as `(\d{10}){10}`), multiply each number of repetitions. Then, add the result of multiplication to the variable *char_class_num*.

For details about the number of repetitions specified for repetition factors, see Table 6-7: Determining the number of repetitions specified for a repetition factor in (c) Determining the variable RTHD_EXESQLSZ.

normal_char_set_num

Total number of regular character set identifiers in the regular expression specified as the pattern character string

If repetition factors are specified for the regular character set identifiers, add the number of repetitions to the value of this variable. If there is a repetition factor in a nested form (such as `([[:alpha:]]{10}){10}`), multiply each number of repetitions. Then, add the result of multiplication to the variable *normal_char_set_num*.

For details about the number of repetitions specified for repetition factors, see Table 6-7: Determining the number of repetitions specified for a repetition factor in (c) Determining the variable RTHD_EXESQLSZ.

Note

In the case of `([[:alpha:]]{10}){10}`, a regular character set is also repeated. Therefore, you must also add 100 to the variable *char_list_num*.

char_list_num

Total number of regular character sets in the regular expression specified as the pattern character string

If repetition factors are specified for the regular character sets, add the number of repetitions to the value of this variable. If there is a repetition factor in a nested form (such as `([abc]{10}){10}`), multiply each number of repetitions. Then, add the result of multiplication to the variable *char_list_num*.

For details about the number of repetitions specified for repetition factors, see Table 6-7: Determining the number of repetitions specified for a repetition factor in (c) Determining the variable RTHD_EXESQLSZ.

repetition_factor_sum

Total of the upper limits specified for the repetition factors in the regular expression specified as the pattern character string

If no upper limits are specified for repetition factors, substitute the lower limit value specified for the repetition factors.

Regular expression specification example

```
((a{2}){5}){8,10}
```

Formula (count)

repetition_factor_sum= 2+5+10 = 17

The following shows examples of regular expressions and formulas for determining the variable *BYTE_CODE_SIZE*.

Regular expression specification example 1

```
KFAA11[0-9]{3}-E
```

Formula example (bytes)

BYTE_CODE_SIZE = 8 x 5 + 1 x 34 x 3 + 3 x 9 + 1 = 170

Explanation
- 8 is substituted for *char_specifier_num*.
- 3 is substituted for *char_list_num* (because the number of regular character sets is 1 and the number of repetitions is 3).
- 3 is substituted for *repetition_factor_sum*.

Regular expression specification example 2

```
KFAA6000[2-9]-E.*uid=12345.*
```

Formula example (bytes)

BYTE_CODE_SIZE = 19 x 5 + 2 x 18 + 34 x 1 = 165

Explanation
- 19 is substituted for *char_specifier_num*.
- 2 is substituted for *quantifier_num*.
- 1 is substituted for *char_list_num*.

CONTAINS_PRD_SIZE

Execution management area for the scalar function `CONTAINS` (bytes)

Use the following formula to determine this value.

Note that if the scalar function `CONTAINS` is not specified, substitute 0 for the variable *CONTAINS_PRD_SIZE*.

CONTAINS_PRD_SIZE=*scan_str_num*×8+*SYN_LIST_SIZE*

scan_str_num

Number of characters in the character string specified as the pattern character string

SYN_LIST_SIZE

Area that is used to store synonyms (bytes)

Use the following formula to determine this value.

Note that if no synonym-search specification is included in the scalar function `CONTAINS`, substitute 0 for the variable *SYN_LIST_SIZE*.

SYN_LIST_SIZE=*syn_str_num*×*syn_group_size*×8

syn_str_num

Average number of synonyms

syn_group_size

Average number of synonyms specified in one synonym group

(h) Determining the variable RTHD_COMMUSZ

Add the variable *RTHD_COMMUSZ* when processing communication. Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_COMMUSZ = RCVBUF + SNDBUF
```

Explanation of variables

RCVBUF: Receive buffer setting

Use the following formula to determine its value.

Formula (kilobytes)

```
RCVBUF = ↑(352 + CMSG) ÷ 1,024↑
```

CMSG: Size of data sent from HADB clients

See the variable *CMSG* under *Memory required for communication between an HADB client and the HADB server* in *Estimating the Memory Requirements for an HADB Client* in the *HADB Application Development Guide*.

SNDBUF: Send buffer setting

For the send buffer size during the initial allocation, substitute the following value.

Value (kilobytes)

```
SNDBUF = 4
```

If send data exceeding the initially allocated send buffer size is created, the HADB server reallocates the value determined from the following formula.

Formula (kilobytes)

```
SNDBUF = ↑SMSG ÷ 4,096↑ × 4
```

SMSG: Size of data sent by the HADB server

The size of send data depends on the processing content. The following table shows the size of the send data allocated in various processes.

Table 6-10: Size of send data allocated in various processes

No.	Processing	Formula for determining the send data size (bytes)
1	SQL statement is executed.	*SMSG* = *srv_base_info* + *srv_execute* + *ARRAY_RESULT*
2	SQL statement execution is NOROW.	*SMSG* = *srv_base_info* + *srv_execute* + *ARRAY_RESULT* + *srv_norow*
3	FETCH	*SMSG* = *srv_base_info* + *srv_fetch* + *FETCH_DATA*

No.	Processing	Formula for determining the send data size (bytes)
4	`FETCH` is `NOROW` (one or more rows were fetched in a batch transfer).	*SMSG* = *srv_base_info* + *srv_fetch* + *srv_norow*
5	SQL statement is preprocessed.	*SMSG* = *srv_base_info* + *srv_prepare* + *PREPARE_INFO_SIZE* + *APATHVIEW*
6	The `adbdbstatus` command is executed	*SMSG* = *srv_base_info* + *srv_status_info*

Legend:

srv_base_info: Send data base information

Substitute 152 bytes.

srv_execute: Execution-specific information

Substitute 24 bytes.

srv_fetch: `FETCH`-specific information

Substitute 16 bytes.

srv_norow: `NOROW`-specific information

Substitute 520 bytes.

srv_prepare: Preprocessing-specific information

Substitute 56 bytes.

srv_status_info: Information specific to DB status analysis

Substitute 1,048,576 bytes.

ARRAY_RESULT: Batch updating result

Use the following formula to determine its value.

Formula (bytes)

```
ARRAY_RESULT = (array_num - 1) × 8
```

array_num: Number of dynamic parameter pairs that were batch-updated

This value depends on the application installation method.

- Application that uses CLI functions

  The value of this variable is the number of `ArrayCount` arguments specified for `a_rdb_SQLBindArrayParams()`. If `a_rdb_SQLBindParams()` is used to join parameters, the value if this variable is fixed to 1.

- Application that uses the JDBC driver

  The value of this variable is the number of parameter lists that are registered by using the `addBatch` method. If the `executeBatch` or `executeLargeBatch` method is not used, the value of this variable is fixed to 1.

FETCH_DATA: Batch transfer row data

Use the following formula to determine its value.

Formula (bytes)

$$FETCH\_DATA = fetch\_size \times \sum_{i=1}^{col\_num} col\_size_{(i)}$$

col_num

Number of selection expressions specified in the SQL statement

col_size(i)

Data length of a selection expression

fetch_size: Number of rows sent in a batch during FETCH processing

Substitute the value specified for the `adb_clt_fetch_size` operand in the client definition. For details about the `adb_clt_fetch_size` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

PREPARE_INFO_SIZE: SQL statement preprocessing information

See the description of the variable *PREPARE_INFO_SIZE* in (c) Determining the variable RTHD_EXESQLSZ.

APATHVIEW: Access path information control area that is created for displaying an access path

See the description of the variable *APATHVIEW* in (b) Determining the variable PROC_TOTALSQLSSZ under (1) Determining the process common memory requirement (during normal operation).

(i) Determining the variable RTHD_EXPSQLTRCSZ

Add the `RTHD_EXPSQLTRCSZ` variable when the SQL trace buffer is extended (the SQL trace buffer is extended when the value of the `RTHD_EXPSQLTRCSZ` variable exceeds 32,768 kilobytes). Use the following formula to determine its value.

Formula (kilobytes)

$$RTHD\_EXPSQLTRCSZ = \uparrow(PARAM\_INFO\_SIZE \times array\_num) \div 1{,}024\uparrow$$

Explanation of variables

PARAM_INFO_SIZE

See the description of the variable *PARAM_INFO_SIZE* in (c) Determining the variable RTHD_EXESQLSZ.

array_num

Number of dynamic parameter sets specified during batch update processing

(j) Determining the variable RTHD_AUDINFSZ

Add the variable *RTHD_AUDINFSZ* when the audit trail facility is enabled. Substitute the following value.

Value (kilobytes)

```
RTHD_AUDINFSZ = 3
```

(k) Determining the variable RTHD_COLUMNIZESZ

Add the variable *RTHD_COLUMNIZESZ* when the updated-row columnizing facility is enabled. Use the following formula to determine this value.

Formula (kilobytes)

$$RTHD\_COLUMNIZESZ = TABLELIST + \max_{1 \leq i \leq table\_num} (TABLEINFO(i) + RELOCATE\_DAT(i) + RTHD\_DATACOMPRESS(i) \times 2)$$

Explanation of variables

TABLELIST: Table name list information

Substitute the following value.

Value (kilobytes)

```
TABLELIST = 833
```

table_num: Total number of column store tables to which the updated-row columnizing facility is applied

For the column store tables to which the updated-row columnizing facility is applied, see (1) Checking the column store tables to which the facility is applied in 11.18.2 Preparation tasks.

TABLEINFO(i): Information about the i-th column store table to which the updated-row columnizing facility is applied

Use the following formula to determine this value.

Formula (kilobytes)

$$TABLEINFO(i) = \uparrow(232 \times column\_num + 1{,}092 \times index\_num + 2{,}504) \div 1{,}024\uparrow + \max_{1 \leq k \leq chunk\_num} SEGMENTLIST(k)$$

column_num

Number of columns in the i-th column store table to which the updated-row columnizing facility is applied

index_num

Number of range indexes defined for the i-th column store table to which the updated-row columnizing facility is applied

chunk_num

Number of chunks in the i-th column store table to which the updated-row columnizing facilityis applied

For a single-chunk table, substitute 1.

SEGMENTLIST(k): Segment list information for the k-th chunk

Use the following formula to determine this value.

Formula (kilobytes)

```
SEGMENTLIST(k) = ↑24 × segment_num ÷ 1,024↑
```

segment_num

Number of row-data segments for the k-th chunk

Determine the value by referring to the explanation of the following variable in (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

■ **For a multi-chunk table**

Variable *ROWDATASEGNUM(i,k)* in (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table)

■ **For a single-chunk table**

Variable *ROWDATASEGNUM(i)* in (b) Determining the variable SGCOLUMNTBL (for a single-chunk table)

RELOCATE_DAT(i): Work area for re-allocating the i-th column store table to which the updated-row columnizing facility is applied

Use the following formula to determine this value.

Formula (kilobytes)

```
RELOCATE_DAT(i) =
  ↑(max_rowsz × 2 + 1,768 + 16 × col_num
       + ↓(96 + 216# + 544 + (652 + page_size) × col_num
             + (80 + page_size) × SEGSIZE + 7) ÷ 8↓ × 8
     + 432 + 128 × col_num
     + ↓(2 × col_num + 7) ÷ 8↓ × 8
     + (1,736 + 16 × col_num) × col_num + max_rowsz
    ) ÷ 1,024↑
  + 16,384 + 1,024 × col_num
```

#

Add this value if range indexes are defined for the i-th column store table to which the updated-row columnizing facility is applied.

max_rowsz

Maximum row length (bytes)

Determine the maximum row length based on the formula for the row length *ROWSZ* in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

page_size

Page size of the data DB area that stores the i-th column store table to which the updated-row columnizing facility is applied (bytes)

Determine this value by referring to Table 6-3: DB area page size in (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

col_num

Number of columns in the i-th column store table to which the updated-row columnizing facility is applied

SEGSIZE

Segment size of the data DB area that stores the i-th column store table to which the updated-row columnizing facility is applied (pages)

Use the following formula to determine this value.

```
SEGSIZE = 4,194,304 ÷ page_size
```

RTHD_DATACOMPRESS(i): Work area for compressing the data of the i-th column store table to which the updated-row columnizing facility is applied

Determine the value by referring to the explanation of the variable *RTHD_DATACOMPRESS* in (d) Determining the variable RTHD_DATALOAD in (2) Determining the real thread private memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

(3) Determining the heap memory requirement (during normal operation)

Use the following formula to determine the heap memory requirements (*HEAP_EXECSZ*) during normal operation.

Formula (kilobytes)

HEAP_EXECSZ = *max_sql*(*UTHD_WORKSZ*) x *rthd_num*

Explanation of variables

rthd_num

Value specified for the `adb_sys_rthd_num` operand in the server definition

max_sql( )

Calculate the result for the variable in the parentheses for each SQL statement to be executed. Then, substitute the largest among the determined values.

UTHD_WORKSZ: Pseudo-thread work area

Determine this value from the following formula. Note, however, that if the following condition is met, substitute 0:

- The SQL statement to be executed is not a `SELECT` statement

Formula (kilobytes)

```
UTHD_WORKSZ = ↑QE x uthd_num ÷ 2↑
```

uthd_num

Value specified for the `adb_sys_uthd_num` operand in the server definition

QE

Value determined by applying the rules for the variable *QB* to the query expression body of the query expression

QB

How you determine the value of this variable depends on whether the query expression body includes set operators.

■ If the query expression body does not include set operators

Value determined by applying the rules for the variable *QS* to the query specification of the query expression body

■ If the query expression body includes set operators

Use the following formula to determine the value:

```
QB = MA(ST1, ... , STi) + setop_num
```

ST1, ... , STi

Value determined by applying the rules for the variable *QS* to each query specification in a set operand

setop_num

Number of set operations specified in the query expression body

QS

How you determine the value of this variable depends on whether the query expression body includes subqueries.

■ If the query specification does not include subqueries

Use the following formula to determine the value:

```
QS = scan_num + 1
```

■ If the query specification includes subqueries

Use the following formula to determine the value:

```
QS = MAX(SQ1, ... , SQi) + scan_num + 1
```

SQ1, ... , SQi

Value determined by applying the rules for the variable *QB* to each subquery specified in the query specification

scan_num

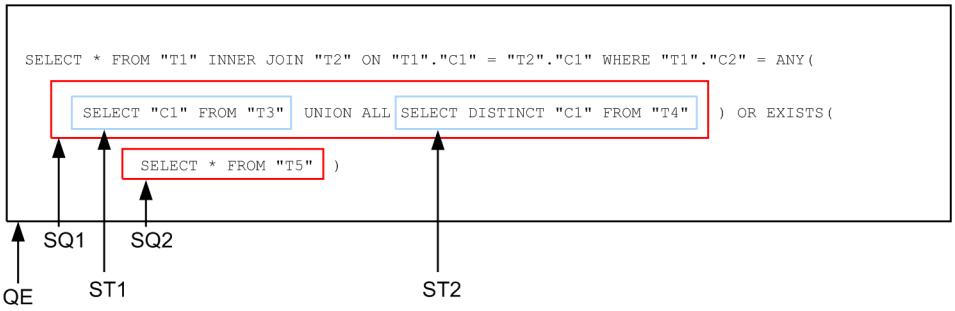
Total number of the following elements specified in the query specification

- `GROUP BY` clauses
- `SELECT DISTINCT`
- `DISTINCT` set functions
- Inverse distribution functions
- Window functions
- Number of derived tables
- Number of joined tables (count each table as 3 in the case of `FULL OUTER JOIN`)
- *number-of-tables-that-are-not-joined-tables* - 1
- *number-of-archivable-multi-chunk-tables* x 3

The following shows an example of determining the *QE* variable:

■ **Example of determining the variable *QE***

■ SQL statement to be executed



■ Formula

Determine QE from the following formula. Determine QB by following steps 1 to 4.

QE = QB

1. Determine QB from the following formula:
(two subqueries, scan_num = 1)

QB = MAX(SQ1 ,SQ2) + 1 + 1

2. Determine SQ1 from the following formula:
(two set operators, setop_num = 1)

SQ1 = MAX(ST1 ,ST2) + 1

ST1 = 1 (no subqueries, scan_num = 0)
ST2 = 2 (no subqueries, scan_num = 1)

Based on the results of the preceding calculations, the value of SQ1 is 3.

SQ1 = MAX(1 ,2) + 1 = 3

3. Determine SQ2 from the following formula:
(no subqueries, scan_num = 0)

SQ2 = 1

4. From the results of the formulas in steps 1 to 3, the value of QB is 5.

QB = MAX(3 ,1) + 2 = 5

Based on the results of the preceding calculations, the value of QE is 5.

Note the following when specifying viewed tables and query names:

■ Notes about specifying viewed tables and query names

When a viewed table or query name is specified in the SQL statement to be executed, estimate the memory requirements as if an internal derived table corresponding to each viewed table or query name were applied. An example follows.

Example: View definition

```
CREATE VIEW "V1" AS SELECT * FROM "T1","T2" WHERE "T1"."C1"="T2"."C2"
```

Example: SQL statement to be executed

```
SELECT * FROM "V1","T3" WHERE "V1"."C1"="T3"."C3"
```

With this particular combination of view definition and SQL statement to be executed, the SQL statement for which memory requirements are to be estimated is as follows:

SQL statement for which memory requirements are to be estimated

```
SELECT * FROM (SELECT * FROM "T1","T2" WHERE "T1"."C1"="T2"."C2"),"T3"
         WHERE "V1"."C1"="T3"."C3"
```

6.3.5 Determining the memory requirement for executing the adbinit command

When executing the `adbinit` command, the HADB server uses the types of memory listed below. Determine the requirement for each type of memory.

■ Shared memory

- Shared memory management area (*INIT_CMN*)

■ Process memory

- Heap memory (*INIT_MEM*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the shared memory management area requirement (for executing the adbinit command)

The amount of shared memory management area required for executing the `adbinit` command (*INIT_CMN*) is the same as the value of the variable *SHMMAN*. For details about the variable *SHMMAN*, see (1) Determining the shared memory management area requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(2) Determining the heap memory requirement (for executing the adbinit command)

Use the following formula to determine the heap memory (*INIT_MEM*) required when executing the `adbinit` command.

Formula (megabytes)

```
INIT_MEM = ↑(16,384 + DBAINF + IOCTL) ÷ 1,024↑ + 661
```

Explanation of variables

DBAINF: DB area setting

Use the following formula to determine its value.

Formula (kilobytes)

```
DBAINF = 1,024 × dbarea_defnum
```

dbarea_defnum: Number specified in the initialization option `adbinitdbarea`

IOCTL: DB area file output control information

Use the following formula to determine its value.

Formula (kilobytes)

```
IOCTL = 8,201 × multi_max
```

multi_max: Value specified for the initialization option `adb_init_multi_max`

6.3.6 Determining the memory requirement for executing the adbimport command

When executing the `adbimport` command, the HADB server uses the types of memory listed below. Determine the requirement for each type of memory.

■ **Shared memory**

- Process common memory (*PROC_IMPORTSZ*)
- Real thread private memory (*RTHD_IMPORTSZ*)

■ **Process memory**

- Heap memory (*HEAP_IMPORTSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbimport command)

Use the following formula to determine the process common memory (*PROC_IMPORTSZ*) required for executing the `adbimport` command.

Formula (kilobytes)

$$PROC\_IMPORTSZ = \Sigma(PROC\_IMPT + PROC\_IDX^{\#1} + PROC\_IDXBUILD^{\#1} + PROC\_RNGIDX^{\#2} + PROC\_INFILE^{\#3} + PROC\_DBUPDINF + PROC\_RTHDUPDINF)^{\#4}$$

Explanation of variables

- *PROC_IMPT*
  Process common memory to be used for importing data
  Determine the value as explained in (a) Determining the variable PROC_IMPT.
- *PROC_IDX*
  Process common memory to be used for creating a B-tree index
  Determine the value as explained in (b) Determining the variable PROC_IDX.
- *PROC_IDXBUILD*
  Process common memory to be used for creating a B-tree index
  Determine the value as explained in (c) Determining the variable PROC_IDXBUILD.
- *PROC_RNGIDX*
  Process common memory to be used for creating a range index
  Determine the value as explained in (d) Determining the variable PROC_RNGIDX.
- *PROC_INFILE*
  Process common memory to be used when the number of input data files exceeds 1,024
  Determine the value as explained in (e) Determining the variable PROC_INFILE.

- *PROC_DBUPDINF*
  Process common memory required for storing DB area, table, index, and chunk update information
  Determine the value as explained in (f) Determining the variable PROC_DBUPDINF.
- *PROC_RTHDUPDINF*
  Processing real thread update information
  Determine the value as explained in (g) Determining the variable PROC_RTHDUPDINF.

#1

Add this value when importing data into a table for which a B-tree index will be defined.

#2

Add this value when importing data into a table for which a range index will be defined.

#3

Add this value when the number of input data files specified in the input data path file name exceeds 1,024.

#4

If you execute multiple `adbimport` commands concurrently, determine the memory requirement for each `adbimport` command. Then, add up the total memory requirements.

(a) Determining the variable PROC_IMPT

The variable *PROC_IMPT* is used to determine the amount of process common memory to be used when importing data. Use the following formula to determine its value.

Formula (kilobytes)

$$PROC\_IMPT = 3 + 135 + DIC + IOA + LOD + IMPORTBUF\_CTL + STS + PAGEALLOC + PROC\_AUDINFSZ^{\#1} + PROC\_IMPT\_COLUMN^{\#2}$$

#1

Add this value when the audit trail facility is enabled.

#2

Add this value when importing data into a column store table.

Explanation of variables

DIC: Memory for acquiring the definition information of the table to be imported

Use the following formula to determine its value.

Formula (kilobytes)

$$DIC = \uparrow(1{,}012 + (512 + 32{,}008^{\#1}) \times col\_num + max\_rowsz + 1{,}156 \times idx\_num + 15{,}864^{\#2}) \div 1{,}024\uparrow$$

#1

Add this value if cost information has been collected in the table.

For details about how to check whether cost information has been collected in the table, see (1) Determining the names of all base tables from which cost information was collected and the collection dates and times in C.9 Searching system tables.

#2

Add this value when the target table is an archivable multi-chunk table.

col_num

Number of columns in the table to be imported

max_rowsz

Maximum row length (bytes)

Determine the maximum row length based on the formula for the row length *ROWSZ* in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

idx_num

Number of indexes defined for the table to be imported

IOA: Memory for creating the data image to be stored in the table

Use the following formula to determine its value.

Formula (kilobytes)

```
IOA =
  ↑(max_rowsz + 436 + 448 × col_num + 124 × idx_num + 4 × dba_num)
    ÷ 1,024↑ × load_rthd
```

- *max_rowsz*

  Maximum row length (bytes)

  Determine the maximum row length based on the formula for the row length *ROWSZ* in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

- *col_num*

  Number of columns in the table to be imported

- *idx_num*

  Number of indexes defined for the table to be imported

- *dba_num*

  Number of DB area files in the data DB areas used for the table to be imported

- *load_rthd*

  Use the following formula to calculate the value:

  ```
  value-specified-for-import-option-adb_import_rthd_num - 1
  ```

LOD: Memory required for storing data

Use the following formula to determine its value.

Formula (kilobytes)

$$LOD=$$
$$\uparrow((read\_size\times read\_buff\_num+decomp\_buff\_size^{\#1}+1)\times 1{,}024\times load\_rthd)$$
$$+((5{,}200+1{,}024{,}000^{\#2}+input\_edit\_area)\times load\_rthd)$$
$$+(511\times input\_file\_num))\div 1{,}024\uparrow$$

#1

Add this value if you specify a compressed (GZIP-format) file as the input data file.

#2

Add if you specify a column structure information file.

read_size

Value specified for the import option `adb_import_read_size`

read_buff_num

Substitute 3.

decomp_buff_size

Value specified for the import option `adb_import_decompress_buff_size`

load_rthd

Use the following formula to calculate the value:

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

input_file_num

Substitute 1,024.

input_edit_area

Length of the input data editing area

Use the following formula to determine its value.

Formula

$$input\_edit\_area = MIN(MAX(\uparrow input\_recsize \div 1{,}024\uparrow \times 1{,}024,\ 32{,}768),\ 536{,}870{,}912)$$

input_recsize

input record length

- Fixed-length data format

  Substitute the value specified for the import option `adb_import_input_record_size`.

- CSV format

  Use the following formula to determine its value.

  Formula

  $$input\_recsize = \Sigma(DATASIZE) + number\ of\ field\ data\ items + 1$$

number-of-field-data-items

Obtain the data field data number of each column specified in the column structure information file or the default value if the specification was omitted. Assign the largest of all such values.

Σ(*DATASIZE*)

Sum of the largest data lengths (in character format) for the data types of all columns in the processing-target table

Determine the largest data length of each data type in character format from the following table.

Table 6-11: Largest data length in character format for each data type

No.	Classification	Data type	Largest data length in character format
1	Numeric data	`INTEGER`	23
2		`SMALLINT`	13
3		`DECIMAL`(*m*,*n*)	*m* + 5
4		`DOUBLE PRECISION`	511
5	Character string data	`CHAR`(*n*)	*n* × 2 + 2
6		`VARCHAR`(*n*)	*n* × 2 + 2

No.	Classification	Data type	Largest data length in character format
7	Datetime data	DATE	12
8		TIME(*p*)	8 + *p* + 3
9		TIMESTAMP(*p*)	19 + *p* + 3
10	Binary data	BINARY(*n*)	*n* × 8 + 2
11		VARBINARY(*n*)	*n* × 8 + 2

Legend:

m, *n*, *p*: See the topic *List of data types* in the manual *HADB SQL Reference*.

IMPORTBUF_CTL: Memory used for controlling the buffer for importing data and creating indexes

Substitute the following value.

Value (kilobytes)

```
IMPORTBUF_CTL = ↑(248 + BUFBLK) ÷ 1,024↑ × load_rthd
```

- *BUFBLK*

  For details, see the description of the variable *BUFBLK* in (f) Determining the variable PROC_UPDSZ under (1) Determining the process common memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

- *load_rthd*

  Use the following formula to calculate the value:

  ```
  value-specified-for-import-option-adb_import_rthd_num - 1
  ```

STS: Memory used for saving status information

Use the following formula to determine its value.

Formula (kilobytes)

```
STS=
 ↑（256＋（1,185×idx_num）＋（2,067×（b_tree_idx_num＋range_idx_num）×load_rthd
  ＋2,067×text_idx_num×16）＋1,025）÷1,024↑
```

idx_num

Number of indexes defined for the table to be imported

b_tree_idx_num

Number of B-tree indexes defined for the table to be imported

range_idx_num

Number of range indexes defined for the table to be imported

load_rthd

Use the following formula to calculate the value:

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

text_idx_num

Number of text indexes defined for the table to be imported

PAGEALLOC: Information for controlling the allocation of pages for importing data

Use the following formula to determine its value.

Formula (kilobytes)

```
PAGEALLOC = import_rthd × 310
```

- *import_rthd*

  Use the following formula to calculate the value:

  ```
  value-specified-for-import-option-adb_import_rthd_num - 1
  ```

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

PROC_IMPT_COLUMN: Memory used to import data into a column store table

Use the following formula to calculate the value:

Formula (kilobytes)

```
PROC_IMPT_COLUMN =
            ↑((16,128 + 8 × col_num) × rthd_num
            + 512 × (infile_num + workfile_num)) ÷ 1,024↑
```

col_num

Number of columns in the table to be imported

rthd_num

Use the following formula to determine this value.

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

infile_num

Number of input data files

workfile_num

Number of storage directories for temporary work files

(b) Determining the variable PROC_IDX

The variable *PROC_IDX* is used to determine the amount of process common memory to be used when creating indexes. Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_IDX =
  ↑(idxf_buff_size × 1,024 + (13,151 + (33,928 + (13 × KEYSZ))
    × idx_num × load_rthd) × cmd_d_opt
    + (152 + rd_buff_size × load_rthd × 2 × 1,024 + 2 × ld_buff_size × 1,024
    + (KEYSZ + CTRL) × text_idx_num) × dividx_rthd) ÷ 1,024↑
```

Explanation of variables

idxf_buff_size

Substitute 1,024.

KEYSZ

Key length of the B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length. If multiple B-tree indexes have been defined for the processing-target table, use the largest key length of all B-tree indexes.

idx_num

Number of B-tree indexes defined for the table to be processed

load_rthd

Use the following formula to calculate the value:

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

cmd_d_opt

The value of the variable *cmd_d_opt* differs depending on whether an option is specified for the `adbimport` command, and on the status of the table to be imported. Determine the value of the variable *cmd_d_opt* according to the following table.

No.	Whether an option is specified for the adbimport command, and the status of the table to be imported	cmd_d_opt value
1	• When the `adbimport` command is executed with the `-d` option specified, but not the `-b` option • When the `adbimport` command is executed without the `-d` or `-b` option specified on an import-target table for which row storage segments are not allocated • When the `adbimport` command is executed with the `-b` option specified, but not the `-d` option, for a multi-chunk table	1
2	When the `adbimport` command is executed without the `-d` or `-b` option specified on an import-target table for which row storage segments are allocated	3

rd_buff_size

Value specified for the import option `adb_import_dividx_rd_buff_size`

ld_buff_size

Value specified for the import option `adb_import_dividx_wt_buff_size`

CTRL

- When the indexed columns are fixed-length columns
  Substitute 10 bytes.
- When the indexed columns include variable-length columns
  Substitute 12 bytes.

text_idx_num

Number of text indexes defined for the table to be imported

dividx_rthd

Use the following formula to determine the value.

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

(c) Determining the variable PROC_IDXBUILD

The variable *PROC_IDXBUILD* is used to determine the amount of process common memory to be used when creating B-tree indexes.

The variable *PROC_IDXBUILD* takes one of the values described in the following, depending on whether the `-d` and `-b` options of the `adbimport` command are specified and on the status of the import-target table.

PROC_IDXBUILD value (kilobytes)

```
PROC_IDXBUILD = PROC_IDXCREATE or PROC_IDXCREATE + PROC_IDXMAINT
```

■ **Cases in which *PROC_IDXCREATE* is used as the value of *PROC_IDXBUILD***

PROC_IDXCREATE is used in any of the following cases:

- The `adbimport` command is executed with the `-d` option specified and the `-b` option not specified.
- When the `adbimport` command is executed without the `-d` or `-b` option specified on an import-target table for which row storage segments are not allocated
- When the `adbimport` command is executed with the `-b` option specified, but not the `-d` option, for a multi-chunk table

■ **Cases in which *PROC_IDXCREATE* + *PROC_IDXMAINT* is used as the value of *PROC_IDXBUILD***

PROC_IDXCREATE + *PROC_IDXMAINT* is used as the value of *PROC_IDXBUILD* in the following case:

- When the `adbimport` command is executed without the `-d` or `-b` option specified on an import-target table for which row storage segments are allocated

Use the following formulas to determine the values of *PROC_IDXCREATE* and *PROC_IDXMAINT*.

Explanation of variables

PROC_IDXCREATE: Use the following formula to determine its value.

Formula (kilobytes)

$$PROC\_IDXCREATE = \max_{1 \le k \le idx\_num} PROC\_IDXCREATE\_MEM_{(k)}$$

idx_num

Number of B-tree indexes defined for the table to be imported

PROC_IDXCREATE_MEM(k)

Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_IDXCREATE_MEM(k) =
  ↑(456 + 256 × idx_col_num + (24 + 144 × ↑idx_lv ÷ 16↑)
   × (dividx_rthd + 1) + (KEYSZ + (idx_col_num + 1) × 2 + 12)
   × 2 + page_size × 2 + 268) ÷ 1,024↑
```

idx_col_num

Number of B-tree indexed columns

idx_lv: Number of levels in the B-tree index

Calculate the recursion formula *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment and substitute the value of *n* when *PIDX(n)* results in 1.

Note that in the case of *PIDX(1)* = 1, the number of levels in the B-tree index is 2.

dividx_rthd

Use the following formula to determine the value.

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

KEYSZ: Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

page_size

Page size of the data DB area in which the B-tree index is defined (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

PROC_IDXMAINT: Use the following formula to determine its value.

Formula (kilobytes)

$$PROC\_IDXMAINT = \max_{1 \le k \le idx\_num} PROC\_IDXMAINT\_MEM_{(k)}$$

idx_num

Number of B-tree indexes defined for the table to be imported

PROC_IDXMAINT_MEM(k)

Use the following formula to determine this value.

Formula (kilobytes)

```
PROC_IDXMAINT_MEM(k) =
   ↑(16 + (8 + KEYSZ + idx_col_num + 1) × (dividx_rthd - 1)
    + 528 + 144 × idx_col_num
    + ↑(KEYSZ + (idx_col_num + 1) × 2 + 12) ÷ 8↑ × 8
    + (176 + 48 × (idx_lv - 1) + 32 × (idx_lv + 1)
    + page_size × 2
    + ↑(KEYSZ + (idx_col_num + 1) × 2 + 3,068) ÷ 8↑ × 8
    + ↑(KEYSZ + (idx_col_num + 1) × 2 + 4) ÷ 8↑× 16
    + ↑(KEYSZ + (idx_col_num + 1) × 2 + 12) ÷ 8↑ × 8)
    × dividx_rthd) ÷ 1,024↑
```

KEYSZ: Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

idx_col_num

Number of B-tree indexed columns

dividx_rthd

Use the following formula to determine the value.

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

idx_lv: Number of levels in the B-tree index

Calculate the recursion formula *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment and substitute the value of *n* when *PIDX(n)* results in 1.

Note that in the case of *PIDX(1)* = 1, the number of levels in the B-tree index is 2.

page_size

Page size of the data DB area in which the B-tree index is defined (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(d) Determining the variable PROC_RNGIDX

The variable *PROC_RNGIDX* is used to determine the amount of process common memory to be used when creating range indexes. Use the following formula to determine its value.

Formula (kilobytes)

PROC_RNGIDX = ↑64 × *rngidx_num* ÷ 1,024↑

Explanation of variable

rngidx_num: Number of range indexes defined for the table to be imported

(e) Determining the variable PROC_INFILE

You need to add the variable *PROC_INFILE* when the number of input data files specified in the input data path file name exceeds 1,024. Use the following formula to determine its value.

Formula (kilobytes)

PROC_INFILE = ↑((16 + 511) × (*INFMAX* - 1,024)) ÷ 1,024↑

Explanation of variable

INFMAX: Number of input data files specified in the input data path file name

(f) Determining the variable PROC_DBUPDINF

Use the following formula to determine the value of variable *PROC_DBUPDINF*.

Formula (kilobytes)

PROC_DBUPDINF=
↑（（40＋↑（*DBUPDINF_ENT_NUM*÷1,024）↑×131,136）÷1,024）↑

Explanation of variable

DBUPDINF_ENT_NUM

Number of entries in the DB area, table, index, and chunk update information

Determine this value from the formula described below. Note that the formula to be used differs depending on whether the multi-node function is used.

Formula to be used when the multi-node function is not used (count)

DBUPDINF_ENT_NUM = 1 + *idx_num* + *rngidx_num* + *txtidx_num*

Formula to be used when the multi-node function is used (count)

DBUPDINF_ENT_NUM =
(1 + *idx_num* + *rngidx_num* + *txtidx_num*) × 2 + *idx_num* × 6 + *rngidx_num* × 5 + *txtidx_num* × 2 + 48

idx_num

Number of B-tree indexes defined for the table to be processed

rngidx_num

Number of range indexes defined for the table to be processed

txtidx_num

Number of text indexes defined for the table to be processed

(g) Determining the variable PROC_RTHDUPDINF

Use the following formula to determine the value of variable *PROC_RTHDUPDINF*.

Formula (kilobytes)

$$PROC\_RTHDUPDINF = \uparrow(648 + (168 \times rthd\_num)) \div 1{,}024\uparrow$$

Explanation of variable

rthd_num

Number of processing real threads

Use the following formula to determine its value.

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

(2) Determining the real thread private memory requirement (for executing the adbimport command)

Use the following formula to determine the real thread private memory (*RTHD_IMPORTSZ*) required for executing the `adbimport` command.

Formula (kilobytes)

$$RTHD\_IMPORTSZ = IMPORTBUF + SORTIOBUF^{\#1} + SORTBUF^{\#1} + RTHD\_DATALOAD + RTHD\_IDXREC^{\#1} + RTHD\_IDXBUILD^{\#1} + RTHD\_RNGIDX^{\#2} + RTHD\_TXTIDX^{\#3}$$

Explanation of variables

- *IMPORTBUF*
  Buffer to be used for importing data and for creating indexes
  Determine the value as explained in (a) Determining the variable IMPORTBUF.
- *SORTIOBUF*
  I/O buffer for sort processing
  Determine the value as explained in (b) Determining the variable SORTIOBUF.
- *SORTBUF*
  Sort buffer during the initialization process
  Determine the value as explained in (c) Determining the variable SORTBUF.
- *RTHD_DATALOAD*
  Buffer used when importing data
  Determine the value as explained in (d) Determining the variable RTHD_DATALOAD.
- *RTHD_IDXREC*
  Buffer used when creating B-tree indexes
  Determine the value as explained in (e) Determining the variable RTHD_IDXREC.
- *RTHD_IDXBUILD*
  Buffer used when creating B-tree indexes

Determine the value as explained in (f) Determining the variable RTHD_IDXBUILD.

- *RTHD_RNGIDX*
  Buffer used when creating range indexes
  Determine the value as explained in (g) Determining the variable RTHD_RNGIDX.
- *RTHD_TXTIDX*
  Buffer used when creating text indexes
  Determine the value as explained in (h) Determining the variable RTHD_TXTIDX.

#1

Add this value when importing data into a table for which a B-tree index is defined.

#2

Add this value when importing data into a table for which a range index is defined.

#3

Add this value when importing data into a table for which a text index is defined.

(a) Determining the variable IMPORTBUF

The variable *IMPORTBUF* is used to determine the amount of real thread private memory to be used when importing data and when creating indexes. Use the following formula to determine its value.

Formula (kilobytes)

```
IMPORTBUF =
     ↑(IMP_SQBLK + IMP_SQIO + IMP_SQPGE + IMP_SQHS) ÷ 1,024↑
```

Explanation of variables

IMP_SQBLK: Use the following formula to determine its value.

Formula (bytes)

```
IMP_SQBLK =
   imp_blknum × (112 + ↑IMP_BLKSZ ÷ imp_pagesize ÷ 64↑ × 24)
```

imp_blknum:

Value specified for the import option `adb_import_buff_blk_num`

IMP_BLKSZ: Use the following formula to determine the value:

```
IMP_BLKSZ = 4,096 × 1,024
```

imp_pagesize:

The smallest of the page sizes of the following DB areas (bytes):

- DB area storing the processing-target table
- DB area storing the B-tree index or text index that is defined for the processing-target table

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

IMP_SQIO: Use the following formula to determine its value.

Formula (bytes)

```
IMP_SQIO = 568 × imp_blknum
```

imp_blknum:

Value specified for the import option `adb_import_buff_blk_num`

IMP_SQPGE: Use the following formula to determine its value.

Formula (bytes)

```
IMP_SQPGE =
   (176 + imp_pagesize + BUFLOG) × (IMP_BLKSZ ÷ imp_pagesize)
     × imp_blknum
```

imp_pagesize:

The smallest of the page sizes of the following DB areas (bytes):

- DB area storing the processing-target table
- DB area storing the B-tree index or text index that is defined for the processing-target table

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

BUFLOG:

For details, see the description of the variable *BUFLOG* in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

IMP_BLKSZ: Use the following formula to determine its value:

```
IMP_BLKSZ = 4,096 × 1,024
```

imp_blknum:

Value specified for the import option `adb_import_buff_blk_num`

IMP_SQHS: Use the following formula to determine its value.

Formula (bytes)

```
IMP_SQHS = (8 × imp_blknum) + (40 × imp_blknum)
```

imp_blknum:

Value specified for the import option `adb_import_buff_blk_num`

(b) Determining the variable SORTIOBUF

The variable *SORTIOBUF* is used to determine the amount of real thread private memory to be used for sort processing. Use the following formula to determine its value.

Formula (kilobytes)

```
SORTIOBUF = sort_io_buff_size
```

Explanation of variables

sort_io_buff_size

Substitute 2,048.

(c) Determining the variable SORTBUF

The variable *SORTBUF* is used to determine the amount of real thread private memory to be used for creating indexes during sort processing. Use the following formula to determine its value.

Formula (kilobytes)

```
SORTBUF = 48 + sort_buff_size × 1,024
```

Explanation of variables

sort_buff_size

Value specified for the import option `adb_import_sort_buff_size`#

#

A work file is created during sort processing. The formula for the variable *sort_buff_size*, which minimizes the size of this file, is shown below. Allocate at least the amount of space indicated by the formula. However, if you are running out of memory or if most data items are already arranged in index key order, do not specify more space than is necessary.

Formula

$$sort\_buff\_size \geq \left\uparrow \left(\frac{REC\_SIZE + 7}{2} + \sqrt{(BLK + 8) \times row\_num \times KEY\_INF + \frac{REC\_SIZE + 7}{4}} + COL\_INFO \right) \div 1{,}024 \right\uparrow$$

Explanation of variables

REC_SIZE: Use the following formula to determine its value.

Formula

```
REC_SIZE = (KEYSZ + SYS1) + SYS2
```

KEYSZ: Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length. If multiple B-tree indexes have been defined for the processing-target table, use the largest key length among all the B-tree indexes.

SYS1: Assume one of the following values:

- If the indexed columns of the indexes defined for the table to be imported are fixed-length: 10
- If any of the indexed columns of the indexes defined for the table to be imported are variable-length: 12

SYS2: Assume one of the following values:

- If any of the indexes defined for the table to be imported are multiple-column indexes: (*number of indexed columns* × 4)
- For all other indexes: 0

BLK: Use the following formula to determine its value.

Formula

```
BLK = REC_SIZE + (KEYSZ + 8) + 56
```

row_num

Number of rows to be stored in the table by the `adbimport` command

KEY_INF: Use the following formula to determine its value.

Formula

```
KEY_INF = REC_SIZE + (KEYSZ + 8) + 28
```

COL_INFO: Use the following formula to determine its value.

Formula

```
COL_INFO = 2,112 + (MULTI_KEY_INFO × 32) + (KEYSZ + 8)
```

MULTI_KEY_INFO: Assume one of the following values:

- If any of the B-tree indexes defined for the table to be imported are variable-length multiple-column indexes: (*number of indexed columns* × 2) + 2
- In all other cases: 5

(d) Determining the variable RTHD_DATALOAD

The variable *RTHD_DATALOAD* is used to determine the amount of real thread private memory to be used for importing data. Use the following formula to determine its value.

The formula differs depending on whether the table to be imported is a row store table or a column store table.

■ For a row store table

Formula (kilobytes)

$$RTHD\_DATALOAD = \uparrow(\downarrow(80+192^{\#1}+8\times(col\_num+var\_col\_num)+32\times var\_col\_num^{\#2}+7)\div 8\downarrow\times 8+513^{\#3})\div 1,024\uparrow$$

#1

Add this value when a range index is defined for the table that is to be imported.

#2

Add this value when the table to be imported is not a FIX table.

#3

Add this value when executing the `adbimport` command with the `-d` option specified for an archivable multi-chunk table.

Explanation of variables

col_num

Number of columns in the table to be imported

var_col_num

Number of `VARCHAR` and `VARBINARY` columns in the table to be imported

■ For a column store table

Formula (kilobytes)

$$RTHD\_DATALOAD = \uparrow\frac{\downarrow\frac{80+192^{\#}+544+(612+page\_size)\times col\_num+(80+page\_size)\times SEGSIZE+7}{8}\downarrow\times 8}{1,024}\uparrow + IMP\_COLUMN + RTHD\_DATACOMPRESS$$

#

Add this value when a range index is defined for the table that is to be imported.

Explanation of variables

page_size

Page size of the DB area that stores the table to be imported (bytes)

See Table 6-3: DB area page size in (2) Determining the global buffer page requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

col_num

Number of columns in the table to be imported

SEGSIZE

Segment size in the DB area in which DB area files are defined

Use the following formula to determine the value:

Formula (pages)

```
SEGSIZE = 4,194,304 ÷ page_size
```

IMP_COLUMN

Work area for importing data into a column store table

Use the following formula to determine the value:

Formula (kilobytes)

```
IMP_COLUMN =
   16,384 + 1,024 × col_num + ↑((1,736 + 16 × col_num) × col_num
   + input_recsize + 8 × col_num + MAX(input_recsize ,  2,097,152) + IMP_ERRD
ATA#)
   ÷ 1,024↑
```

#

Add this value when the `adb_import_errdata_file_name` import option is specified and the data to be imported is in CSV format.

col_num

Number of columns in the table to be imported

input_recsize

Input record length

For details, see the description of the variable *LOD* in (a) Determining the variable PROC_IMPT under (1) Determining the process common memory requirement (for executing the adbimport command).

IMP_ERRDATA

Error record output buffer

Use the following formula to determine the value:

Formula (bytes)

```
IMP_ERRDATA = input_recsize × 1.2
```

input_recsize

Input record length

For details, see the description of the variable *LOD* in (a) Determining the variable PROC_IMPT under (1) Determining the process common memory requirement (for executing the adbimport command).

RTHD_DATACOMPRESS

Work area for data compression

Use the following formula to determine the value:

Formula (kilobytes)

$$RTHD\_DATACOMPRESS = \left\uparrow \frac{\sum_{i=1}^{col\_num\_in\_table} col\_compress\_work\_size(i)}{1,024} \right\uparrow$$

col_num_in_table

Number of columns in the table to be imported (tables)

col_compress_work_size(i)

Work area required to compress each column (bytes)

When defining the table to be imported in the `CREATE TABLE` statement, the value to be substituted differs depending on the column-data compression type of each column. Substitute the value in the following table that corresponds to the column-data compression type. For details about specifying the column-data compression type, see *CREATE TABLE (define a table)* in *Definition SQL* in the manual *HADB SQL Reference*.

Table 6-12: Work area required to compress each column (value to substitute for col_compress_work_size variable)

No.	Column-data compression type	Work area required to compress each column (value to substitute for *col_compress_work_size* variable) (bytes)
1	`NONE`	65,552
2	`RUNLENGTH`	65,552
3	`DICTIONARY`	23,270,960
4	`DELTA`	65,552
5	`DELTA_RUNLENGTH`	65,552
6	`AUTO`	23,270,960

(e) Determining the variable RTHD_IDXREC

The variable *RTHD_IDXREC* is used to determine the amount of real thread private memory to be used when creating B-tree indexes. Use the following formula to determine its value.

Formula (kilobytes)

RTHD_IDXREC =
↑(386 + 328 × (*idx_num* - 1) + 32 × *max_idx_col_num*
+ (64 + *buf_size* × 1,024) × 2 × *idx_num*) ÷ 1,024↑

Explanation of variables

- *idx_num*
  Number of B-tree indexes defined for the table that is to be imported
- *max_idx_col_num*
  Maximum number of indexed columns in the B-tree index defined for the table that is to be imported
- *buf_size*
  Substitute 1,024.

(f) Determining the variable RTHD_IDXBUILD

The variable *RTHD_IDXBUILD* is used to determine the amount of real thread private memory to be used when creating B-tree indexes. Use the following formula to determine the value of variable *RTHD_IDXBUILD*.

Formula (kilobytes)

$$RTHD\_IDXBUILD = \max_{1 \leq k \leq idx\_num} RRTHD\_IDXBUILD\_MEM_{(k)}$$

Explanation of variables

idx_num

Number of B-tree indexes defined for the table that is to be imported

RTHD_IDXBUILD_MEM$_{(k)}$

Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_IDXBUILD_MEM (k) =
   ↑(61,474 + ↑MAX (255,KEYSZ) ÷ 2↑ × 2 + 3,070
   + ↑KEYSZ + CTRL ÷ 8↑ × 8 + MAX (255,KEYSZ)
   + 14 + ↓8 + page_size × 95 ÷ 100↓ × 2) ÷ 1,024↑
```

page_size

Page size of the data DB area in which the B-tree index is defined (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

KEYSZ

Key length of the B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL

- When all key lengths in the indexed columns are fixed

  Substitute 10 bytes.
- When some of the key lengths in the indexed columns are variable

  Substitute 12 bytes.

(g) Determining the variable RTHD_RNGIDX

The variable *RTHD_RNGIDX* is used to determine the amount of real thread private memory to be used when creating range indexes.

Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_RNGIDX = ↑(16 + 128 × rngidx_num) × load_rthd ÷ 1,024↑
```

Explanation of variables

rngidx_num

Number of range indexes defined for the table that is to be imported

load_rthd

Use the formula below to determine the value:

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

(h) Determining the variable RTHD_TXTIDX

The variable *RTHD_TXTIDX* is used to determine the amount of real thread private memory used to build a text index.

Use the following formula to determine its value.

Formula (kilobytes)

$$RTHD\_TXTIDX = \max_{1 \le k \le idx\_num} RTHD\_TXTIDXBUILD\_MEM_{(k)}$$

Explanation of variables

idx_num

Number of text indexes defined for the table to be processed

RTHD_TXTIDXBUILD_MEM$_{(k)}$

Use the following formula to determine its value.

Formula (kilobytes)

$$RTHD\_TXTIDXBUILD\_MEM_{(k)} = buff\_blk\_size \times buff\_blk\_num + 3{,}072 + \uparrow(idx\_div\_num \times 24) \div 1{,}024\uparrow + txt\_sort\_buff\_size \times 1{,}024$$

buff_blk_size

Substitute 4,096.

buff_blk_num

Value specified for the import option `adb_import_buff_blk_num`

idx_div_num

Length defined for the indexed column for text index *k*

txt_sort_buff_size

Value specified for the import option `adb_import_txt_buff_size`

(3) Determining the heap memory requirement (for executing the adbimport command)

Use the following formula to determine the heap memory (*HEAP_IMPORTSZ*) required to execute the `adbimport` command.

Formula (kilobytes)

$$HEAP\_IMPORTSZ = \sum(HEAP\_ZLIB + HEAP\_SRTHMEM)^{\#}$$

#

If you execute multiple `adbimport` commands concurrently, determine the memory requirement for each `adbimport` command. Then, add up the total memory requirements.

Explanation of variables

HEAP_ZLIB

See (a) Determining the variable HEAP_ZLIB.

HEAP_SRTHMEM

See (b) Determining the variable HEAP_SRTHMEM.

(a) Determining the variable HEAP_ZLIB

Use the following formula to determine the value of variable *HEAP_ZLIB*.

Formula (kilobytes)

$$HEAP\_ZLIB = 42 \times load\_rthd$$

Explanation of variables

load_rthd

Use the formula below to determine the value:

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

(b) Determining the variable HEAP_SRTHMEM

Use the following formula to determine the value of variable *HEAP_SRTHMEM*.

Formula (kilobytes)

$$HEAP\_SRTHMEM = 505 \times sort\_rthd$$

Explanation of variables

sort_rthd

Use the following formula to determine the value.

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

6.3.7 Determining the memory requirement for executing the adbidxrebuild command

When the `adbidxrebuild` command is executed, the HADB server uses the types of memory described below. You need to determine the memory requirement for each type.

■ Shared memory

- Process common memory (*PROC_IDXRBLDSZ*)

- Real thread private memory (*RTHD_IDXRBLDSZ*)

■ **Process memory**

- Heap memory (*HEAP_IDXRBLDSZ*)

The following subsections describe the formulas for determining these memory requirements.

(1) Determining the process common memory requirement (for executing the adbidxrebuild command)

Use the following formula to determine the amount of process common memory (*PROC_IDXRBLDSZ*) needed to execute the `adbidxrebuild` command.

Formula (kilobytes)

$$PROC\_IDXRBLDSZ =$$
$$\Sigma(PROC\_MNG + PROC\_IDX^{\#1} + PROC\_IDXREBUILD^{\#1} + PROC\_RNGIDX^{\#2}$$
$$+ PROC\_DBUPDINF^{\#3} + PROC\_RTHDUPDINF)^{\#4}$$

Explanation of variables

- *PROC_MNG*
  Process common memory used to manage index rebuilding
  Determine the value as explained in (a) Determining the variable PROC_MNG.
- *PROC_IDX*
  Process common memory used for batch-creating indexes
  Determine the value as explained in (b) Determining the variable PROC_IDX.
- *PROC_IDXREBUILD*
  Process common memory used for rebuilding B-tree indexes
  Determine the value as explained in (c) Determining the variable PROC_IDXREBUILD.
- *PROC_RNGIDX*
  Process common memory used for batch-updating range indexes
  Determine the value as explained in (d) Determining the variable PROC_RNGIDX.
- *PROC_DBUPDINF*
  Process common memory used for storing DB area, table, index, and chunk update information
  Determine the value as explained in (e) Determining the variable PROC_DBUPDINF.
- *PROC_RTHDUPDINF*
  Process common memory required for managing processing real thread update information
  Determine the value as explained in (f) Determining the variable PROC_RTHDUPDINF.

#1

Add this value when the index to be processed contains a B-tree index.

#2

Add this value when the index to be processed contains a range index.

#3

Add this value when you use the multi-node function.

#4

If you execute multiple `adbidxrebuild` commands concurrently, determine the memory requirement for each `adbidxrebuild` command. Then, add up the total memory requirements.

(a) Determining the variable PROC_MNG

The variable *PROC_MNG* is used to determine the amount of process common memory used for managing index rebuilding. Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_MNG =
    3 + DIC + IOA + SCAN + IDXRBLDBUF_CTL + STS + PAGEALLOC + PROC_AUDINFSZ#
```

#

Add this value when the audit trail facility is enabled.

Explanation of variables

DIC: Memory used to acquire the definition information of the table to be processed

See the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

IOA: Memory used to access the database

Use the following formula to determine its value.

Formula (kilobytes)

```
IOA = 1,056 + (scan_buff_size + offset_area × 2) × scan_rthd
```

- *scan_buff_size*

  Use the following formula to determine its value:

  value-specified-for-index-rebuild-option-adb_idxrebuild_scan_buff_size × 1,024

- *offset_area*

  Use the following formula to determine its value:

  (8 × *number-of-columns-comprising-processing-target-index* × (*scan_buff_size* × 1,024
  ÷ (*sum-total-of-lengths-of-columns-comprising-processing-target-index* + 8))) ÷ 1,024

- *scan_rthd*

  Use the following formula to determine the value.

  ↓(*value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num* - 1) ÷ 2↓

SCAN: Memory needed to output the index record file

Use the following formula to determine its value.

Formula (kilobytes)

```
SCAN =
   ↑(write_size × scan_rthd × b-tree_index_num)↑
```

- *write_size*

  Value specified for the index rebuild option `adb_idxrebuild_dvix_wtbuff_size`

- *scan_rthd*

Use the following formula to determine the value.

```
↓(value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1) ÷ 2↓
```

- *b-tree_index_num*

  Number of B-tree indexes to be processed

IDXRBLDBUF_CTL: Buffer control information for rebuilding indexes

Use the following formula to determine its value.

Formula (kilobytes)

```
IDXRBLDBUF_CTL =
   ↑(248 + BUFBLK) ÷ 1,024↑ × scan_rthd
```

- *BUFBLK*

  For details, see the description of the variable *BUFBLK* in (f) Determining the variable PROC_UPDSZ under (1) Determining the process common memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

- *scan_rthd*

  Use the following formula to determine the value.

  ```
  ↓(value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1) ÷ 2↓
  ```

STS: Memory for saving status information

Use the following formula to determine its value.

Formula (kilobytes)

```
STS=
 ↑（256＋（1,185×idx_num）＋（2,067×（b_tree_idx_num＋range_idx_num）×scan_rthd
  ＋2,067×text_idx_num×16）＋1,025）÷1,024↑＋imp_sts
```

idx_num

Number of indexes to be processed

b_tree_idx_num

Number of B-tree indexes defined for the table to be processed

range_idx_num

Number of range indexes defined for the table to be processed

scan_rthd

Use the following formula to determine the value.

```
↓(value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1) ÷ 2↓
```

text_idx_num

Number of text indexes defined for the table to be processed

imp_sts

Memory for saving status information of `adbimport` command

If you will be executing the `adbidxrebuild` command with the `--create-temp-file` option specified after the `adbimport` command is interrupted, add the value determined for the variable *STS* as explained in (a) Determining the variable PROC_IMPT under (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

PAGEALLOC: Page allocation control information for rebuilding indexes

Use the following formula to determine its value.

Formula (kilobytes)

```
PAGEALLOC = dividx_rthd × 310
```

- *dividx_rthd*

  Use the following formula to determine the value.

  ```
  value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1
  ```

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(b) Determining the variable PROC_IDX

The variable *PROC_IDX* is used to determine the amount of process common memory used to batch-create indexes. Use the following formula to determine its value.

Formula (kilobytes)

$$
\begin{aligned}
&PROC\_IDX = \\
&\quad \uparrow((13{,}151 + (33{,}928 + (13 \times KEYSZ)) \times scan\_rthd \\
&\qquad + (152 + rd\_buff\_size \times scan\_rthd \times 2 \times 1{,}024 + 2 \times ld\_buff\_size \times 1{,}024 \\
&\qquad + (KEYSZ + CTRL) \times text\_idx\_num) \times dividx\_rthd) \times (b\_tree\_idx\_num + text\_idx\_num)) \div 1{,}024 \uparrow
\end{aligned}
$$

Explanation of variables

KEYSZ

Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length. If multiple B-tree indexes have been defined for the processing-target table, use the largest key length among all the B-tree indexes.

scan_rthd

Use the following formula to determine the value:

```
↓(value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1) ÷ 2↓
```

rd_buff_size

Value specified for the index rebuild option `adb_idxrebuild_dvix_rdbuff_size`

ld_buff_size

Value specified for the index rebuild option `adb_idxrebuild_dvix_wtbuff_size`

dividx_rthd

Use the following formula to determine the value:

```
value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1
```

CTRL

- If the indexed columns are all fixed-length columns

  Substitute 10 bytes.

- If the indexed columns include variable-length columns

  Substitute 12 bytes.

b_tree_idx_num

Number of B-tree indexes to be processed

text_idx_num

Number of text indexes defined for the table to be processed

(c) Determining the variable PROC_IDXREBUILD

The variable *PROC_IDXREBUILD* is used to determine the amount of process common memory used to rebuild B-tree indexes. Use the following formula to determine its value.

Formula (kilobytes)

$$PROC\_IDXREBUILD = \max_{1 \leq k \leq idx\_num} PROC\_IDXREBUILD\_MEM_{(k)}$$

idx_num

Number of B-tree indexes to be rebuilt

PROC_IDXREBUILD_MEM(k)

Use the following formula to determine this value.

Formula (kilobytes)

```
PROC_IDXREBUILD_MEM(k) =
  ↑(184 + 256 × idx_col_num + (18 + 256 × ↑idx_lv ÷ 16↑) × dividx_rthd
    + KEYSZ + (idx_col_num + 1) × 2 + 4) ÷ 1,024↑
```

- *idx_col_num*

  Number of indexed columns in a B-tree index

- *idx_lv*

  Number of levels in the B-tree index

  Calculate the recursion formula *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment and substitute the value of *n* when *PIDX(n)* results in 1.

  Note that in the case of *PIDX(1)* = 1, the number of levels in the B-tree index is 2.

- *dividx_rthd*

  Use the following formula to determine the value:

  ```
  value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1
  ```

- *KEYSZ*: Key length of a B-tree index (bytes)

  Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

(d) Determining the variable PROC_RNGIDX

The variable *PROC_RNGIDX* is used to determine the amount of process common memory used to batch-update range indexes. Use the following formula to determine its value.

Formula (kilobytes)

PROC_RNGIDX = ↑64 × *rngidx_num* ÷ 1,024↑

Explanation of variables

rngidx_num: Number of range indexes to be rebuilt

(e) Determining the variable PROC_DBUPDINF

Use the following formula to determine the value of variable *PROC_DBUPDINF*.

Formula (kilobytes)

```
PROC_DBUPDINF=
    ↑（（40+↑（DBUPDINF_ENT_NUM÷1,024)↑×131,136）÷1,024）↑
```

Explanation of variables

DBUPDINF_ENT_NUM

Number of entries in the DB area, table, index, and chunk update information

Use the following formula to determine its value.

Formula (entries)

$$DBUPDINF\_ENT\_NUM = (idx\_num + (idx\_num \times 2 + rngidx\_num + txtidx\_num \times 2) \times chunk\_num) + idx\_num \times 3 + (rngidx\_num \times 3) + 15$$

idx_num

Number of B-tree indexes to be rebuilt

rngidx_num

Number of range indexes to be rebuilt

txtidx_num

Number of text indexes to be rebuilt

chunk_num

Number of chunks in the table in which the indexes to be rebuilt are defined

(f) Determining the variable PROC_RTHDUPDINF

Use the following formula to determine the value of variable *PROC_RTHDUPDINF*.

Formula (kilobytes)

$$PROC\_RTHDUPDINF = \uparrow(648 + (168 \times rthd\_num)) \div 1{,}024\uparrow$$

Explanation of variables

rthd_num

Number of processing real threads

Use the following formula to determine its value.

```
value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1
```

(2) Determining the real thread private memory requirement (for executing the adbidxrebuild command)

Use the following formula to determine the real thread private memory requirement (*RTHD_IDXRBLDSZ*) for executing the `adbidxrebuild` command.

Formula (kilobytes)

RTHD_IDXRBLDSZ =
IDXRBLDBUF#1 + *RTHD_SCAN* + *SORTIOBUF*#2 + *SORTBUF*#2 + *RTHD_IDXREC*#2
+ *RTHD_IDXREBUILD*#2 + *RTHD_RNGIDX*#3 + *RTHD_TXTIDX*#4

Explanation of variables

- *IDXRBLDBUF*
  Buffer used for index rebuilding
  Determine the value as explained in (a) Determining the variable IDXRBLDBUF.
- *RTHD_SCAN*
  Real thread private memory used for table searches when creating index record files
  Determine the value as explained in (b) Determining the variable RTHD_SCAN.
- *SORTIOBUF*
  Sorting I/O buffer
  Determine the value as explained in (c) Determining the variable SORTIOBUF.
- *SORTBUF*
  Sort buffer for index creation
  Determine the value as explained in (d) Determining the variable SORTBUF.
- *RTHD_IDXREC*
  Real thread private memory for index rebuilding
  Determine the value as explained in (e) Determining the variable RTHD_IDXREC.
- *RTHD_IDXREBUILD*
  Real thread private memory used for B-tree index rebuilding
  Determine the value as explained in (f) Determining the variable RTHD_IDXREBUILD.
- *RTHD_RNGIDX*
  Real thread private memory used for batch- updating range indexes
  Determine the value as explained in (g) Determining the variable RTHD_RNGIDX.
- *RTHD_TXTIDX*
  Real thread private memory used for batch- updating text indexes
  Determine the value as explained in (h) Determining the variable RTHD_TXTIDX.

#1

If all the processing-target indexes are range indexes, substitute 0.

#2

Add this value when you are rebuilding a B-tree index.

#3

Add this value when a range index is included among the indexes to be rebuilt.

#4

Add this value when a text index is included among the indexes to be rebuilt.

(a) Determining the variable IDXRBLDBUF

The variable *IDXRBLDBUF* is used to determine the amount of real thread private memory used to rebuild an index. Use the following formula to determine its value.

Formula (kilobytes)

```
IDXRBLDBUF =
  ↑(RBLD_SQBLK + RBLD_SQIO + RBLD_SQPGE + RBLD_SQHS) ÷ 1,024↑
```

Explanation of variables

RBLD_SQBLK: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQBLK =
  rbld_blknum × (112 + ↑RBLD_BLKSZ ÷ rbld_pagesize ÷ 64↑ × 24)
```

- *rbld_blknum*

  Value specified for the index rebuild option `adb_idxrebuild_buff_blk_num`

- *RBLD_BLKSZ*

  Substitute 4,194,304.

- *rbld_pagesize*

  The smallest of the page sizes of the DB areas storing the B-tree indexes or text indexes that are targeted for batch index creation (bytes)

  Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

RBLD_SQIO: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQIO = 568 × rbld_blknum
```

- *rbld_blknum*

  Value specified for the index rebuild option `adb_idxrebuild_buff_blk_num`

RBLD_SQPGE: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQPGE =
  (176 + rbld_pagesize + BUFLOG) × (RBLD_BLKSZ ÷ rbld_pagesize)
  × rbld_blknum
```

- *rbld_pagesize*

  The smallest of the page sizes of the DB areas storing the B-tree indexes or text indexes that are targeted for batch index creation (bytes)

  Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *BUFLOG*

For details, see the description of the variable *BUFLOG* in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *RBLD_BLKSZ*
  Substitute 4,194,304.
- *rbld_blknum*
  Value specified for the index rebuild option `adb_idxrebuild_buff_blk_num`

RBLD_SQHS: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQHS = 8 × rbld_blknum + 40 × rbld_blknum
```

- *rbld_blknum*
  Value specified for the index rebuild option `adb_idxrebuild_buff_blk_num`

(b) Determining the variable RTHD_SCAN

The variable *RTHD_SCAN* is used to determine the amount of real thread private memory to be used for table searches when creating index record files. Use the following formula to determine the value:

Formula (kilobytes)

```
RTHD_SCAN = MAX( RTHD_EXESQLSZ , RTHD_EXESQLDICSZ )
```

Note

Determine the values of the variables *RTHD_EXESQLSZ* and *RTHD_EXESQLDICSZ* on the assumption that the following SQL statement were executed:

```
SELECT selection-expression# FROM "name-of-table-to-be-processed"
```

#

Determine the value on the basis of `CHAR(16)` columns and as if duplicates were removed from all indexed columns of the index to be processed.

Explanation of the variables

RTHD_EXESQLSZ

See (c) Determining the variable RTHD_EXESQLSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

RTHD_EXESQLDICSZ

See (g) Determining the variable RTHD_EXESQLDICSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

(c) Determining the variable SORTIOBUF

The variable *SORTIOBUF* is used to determine the amount of real thread private memory used for sort processing. Use the following formula to determine its value.

Formula (kilobytes)

```
SORTIOBUF = sort_io_buff_size × sort_rthd
```

Explanation of variables

- *sort_io_buff_size*
  Substitute 16.
- *sort_rthd*
  Use the following formula to determine the value:

```
value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1
```

(d) Determining the variable SORTBUF

The variable *SORTBUF* is used to determine the amount of real thread private memory used for sort processing during index rebuilding. Use the following formula to determine its value.

Formula (kilobytes)

```
SORTBUF = (48 + sort_buff_size × 1,024) × sort_rthd
```

Explanation of variables

sort_rthd
Use the following formula to determine the value:

```
value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1
```

sort_buff_size
Value specified for the index rebuild option `adb_idxrebuild_sort_buff_size`#

#
A work file is created during sort processing. The formula for the variable *sort_buff_size*, which minimizes the size of this file, is shown below. Allocate at least the amount of space indicated by the formula. However, if you are running out of memory or if most data items are already arranged in index key order, do not specify more space than is necessary.

Formula

$$sort\_buff\_size \geq \left\uparrow \left(\frac{REC\_SIZE + 7}{2} + \sqrt{(BLK + 8) \times row\_num \times KEY\_INF + \frac{REC\_SIZE + 7}{4}} + COL\_INFO \right) \div 1{,}024 \right\uparrow$$

Explanation of variables

REC_SIZE: Use the following formula to determine its value.

Formula

```
REC_SIZE = (KEYSZ + SYS1) + SYS2
```

KEYSZ: Index key length of the B-tree indexes defined for the table to be processed

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length. If multiple B-tree indexes have been defined for the processing-target table, use the largest key length among all the B-tree indexes.

SYS1: Assume one of the following values:

- If the indexed columns of the indexes defined for the table to be processed are fixed-length: 10
- If any of the indexed columns of the indexes defined for the table to be imported are variable-length: 12

SYS2: Assume one of the following values:

- If any of the indexes defined for the table to be processed are variable-length multiple-column indexes: (*number of indexed columns* × 4)
- For all other indexes: 0

BLK: Use the following formula to determine its value.

Formula

```
BLK = REC_SIZE + (KEYSZ + 8) + 56
```

row_num

Number of rows stored in the table to be processed

KEY_INF: Use the following formula to determine its value.

Formula

```
KEY_INF = REC_SIZE + (KEYSZ + 8) + 28
```

COL_INFO: Use the following formula to determine its value.

Formula

```
COL_INFO = 2,112 + (MULTI_KEY_INFO × 32) + (KEYSZ + 8)
```

MULTI_KEY_INFO: Assume one of the following values:

- If any of the B-tree indexes defined for the table to be processed are variable-length multiple-column indexes

  (*number of indexed columns* × 2) + 2

- In all other cases

  5

(e) Determining the variable RTHD_IDXREC

The variable *RTHD_IDXREC* is used to determine the amount of real thread private memory used for index rebuilding. Use the following formula to determine its value.

Formula (kilobytes)

RTHD_IDXREC =
↑(386 + 328 × (*idx_num* - 1) + 32 × *max_idx_col_num*
+ (64 + *buf_size* × 1,024) × 2 × *idx_num*) ÷ 1,024↑

Explanation of variables

- *idx_num*
  Number of B-tree indexes to be rebuilt
- *max_idx_col_num*
  Maximum number of indexed columns in the B-tree index to be rebuilt
- *buf_size*
  Substitute 1,024.

(f) Determining the variable RTHD_IDXREBUILD

The variable *RTHD_IDXREBUILD* is used to determine the amount of real thread private memory used to rebuild a B-tree index. Use the following formula to determine its value.

Formula (kilobytes)

$$RTHD\_IDXREBUILD = \max_{1 \leq k \leq idx\_num} RTHD\_IDXREBUILD\_MEM_{(k)}$$

Explanation of variables

idx_num

Number of B-tree indexes to be rebuilt

RTHD_IDXREBUILD_MEM(k)

Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_IDXREBUILD_MEM (k) =
  ↑ (61,458 + ↑MAX (255, KEYSZ) ÷ 2↑ × 2 + 3,070
   + ↑KEYSZ + CTRL ÷ 8↑ × 8 + MAX (255, KEYSZ) + 14
   + ↓8 + page_size × 95 ÷ 100↓ × 2) ÷ 1,024↑
```

page_size

Page size of the data DB area in which the B-tree index is defined (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

KEYSZ

Key length of the B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL

- When all key lengths in the indexed columns are fixed: 10 bytes
- When some of the key lengths in the indexed columns are variable: 12 bytes

(g) Determining the variable RTHD_RNGIDX

The variable *RTHD_RNGIDX* is used to determine the amount of real thread private memory used for batch-updating range indexes. Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_RNGIDX = ↑(16 + 128 × rngidx_num) × scan_rthd ÷ 1,024↑
```

Explanation of variables

- *rngidx_num*

  Number of range indexes to be rebuilt

- *scan_rthd*

  Use the following formula to determine the value:

  ```
  ↓(value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1) ÷ 2↓
  ```

(h) Determining the variable RTHD_TXTIDX

The variable *RTHD_TXTIDX* is used to determine the amount of real thread private memory to be used for batch updating of text indexes. Use the following formula to determine its value.

Formula (kilobytes)

$$RTHD\_TXTIDX = \max_{1 \le k \le idx\_num} RTHD\_TXTIDXBUILD\_MEM_{(k)}$$

Explanation of variables

idx_num

Number of text indexes defined for the table to be processed

$RTHD\_TXTIDXBUILD\_MEM_{(k)}$

Use the following formula to determine its value.

Formula (kilobytes)

$$RTHD\_TXTIDXBUILD\_MEM_{(k)} = buff\_blk\_size \times buff\_blk\_num + 3{,}072 + \uparrow(idx\_div\_num \times 24) \div 1{,}024\uparrow + txt\_sort\_buff\_size \times 1{,}024$$

buff_blk_size

Substitute 4,096.

buff_blk_num

Value specified for the index rebuild option `adb_idxrebuild_buff_blk_num`

idx_div_num

Length defined for the indexed column for text index *k*

txt_sort_buff_size

Value specified for the index rebuild option `adb_idxrebuild_txt_buff_size`

(3) Determining the heap memory requirement (for executing the adbidxrebuild command)

Use the following formula to determine the heap memory (*HEAP_IDXRBLDSZ*) required when executing the `adbidxrebuild` command.

Formula (kilobytes)

$$HEAP\_IDXRBLDSZ = \Sigma\ (505 \times sort\_rthd)^{\#}$$

Explanation of variables

sort_rthd

Use the following formula to determine the value:

value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1

#

If you execute multiple `adbidxrebuild` commands concurrently, determine the amount of heap memory used by each `adbidxrebuild` command. Then, add up the total heap memory requirements.

6.3.8 Determining the memory requirement for executing the adbgetcst command

When executing the `adbgetcst` command, the HADB server uses the types of memory listed below. Determine the requirement for each type of memory.

■ Shared memory

- Process common memory (*PROC_GETCOSTSZ*)
- Real thread private memory (*RTHD_GETCOSTSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbgetcst command)

Use the following formula to determine the process common memory (*PROC_GETCOSTSZ*) required for executing the `adbgetcst` command.

Formula (kilobytes)

```
PROC_GETCOSTSZ = Σ(PROC_MNG + PROC_CST)#
```

Explanation of variables

- *PROC_MNG*
  Process common memory to be used for managing the collection of cost information
  Determine the value as explained in (a) Determining the variable PROC_MNG.
- *PROC_CST*
  Process common memory to be used as a buffer when collecting cost information
  Determine the value as explained in (b) Determining the variable PROC_CST.

#

If you execute multiple `adbgetcst` commands concurrently, determine the memory requirement for each `adbgetcst` command. Then, add up the total memory requirements.

(a) Determining the variable PROC_MNG

The variable *PROC_MNG* is used to determine the amount of process common memory to be used for managing the collection of cost information. Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_MNG=
    3+DIC+IOA+TBL+DBSTA+PROC_AUDINFSZ#
```

#

Add this value when the audit trail facility is enabled.

Explanation of variables

DIC

Memory for acquiring the definition information of the table to be processed

Determine the value by referring to the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

IOA

Memory to be used for database access

Formula (kilobytes)

$$IOA = 32 + 1{,}024$$

TBL

Memory to be used for managing the tables to be processed

Formula (kilobytes)

$$TBL = 64 \times tbl\_num$$

- *tbl_num*
  Number of tables to be processed

DBSTA

Memory to be used for DB status analysis

Formula (kilobytes)

$$DBSTA = \uparrow \frac{256{,}088 + 88 \times \text{number of chunks created in the table to be processed}}{1{,}024} \uparrow$$

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(b) Determining the variable PROC_CST

The variable *PROC_CST* is used to determine the amount of process common memory to be used as a buffer when collecting cost information.

If the `adbgetcst` command is executed with the `-t` option omitted, determine the variable *PROC_CST* for all tables in the schema of the authorization identifier specified in the `-u` option. Then use the largest such value as the value of the variable *PROC_CST*.

Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_CST = DBHTBLINF + DICTBLINF
```

Explanation of variables

DBHTBLINF

Memory to be used for collecting the cost information of a table to be processed

Formula (kilobytes)

$$DBHTBLINF = \left\lceil \left\{ 56 + 8 \times col\_num + sql\_rthd\_num \times (128 + 8 \times col\_num) + \left(\sum_{i=1}^{col\_num} \left(40 + \left(\left\lceil \frac{coldata_{(i)}}{2} \right\rceil \times 2 \right) \times 2 + 80 + 8 \times hash\_size + \left(\left\lceil \frac{coldata_{(i)}}{2} \right\rceil \times 2 \right) \times cardinality\_num \right) \times sql\_rthd\_num \right) \right\} \div 1{,}024 \right\rceil + 1$$

col_num

Number of columns in the table to be processed

sql_rthd_num

Use the following formula to determine the value:

```
value-specified-for-cost-information-collection-option-adb_getcst_rthd_num - 1
```

coldata$_{(i)}$

Data length of each column

Determine the value according to Table 6-9: Data length of each data type in (c) Determining the variable RTHD_EXESQLSZ under (2) Determining the real thread private memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

hash_size

Substitute 65,536.

cardinality_num

Substitute 20,000.

DICTBLINF

Memory used for storing the cost information of the table to be processed in the system table

Formula (kilobytes)

$$DICTBLINF = \mathrm{MAX}\left\{ 64, \left\lceil \frac{(128 + 128 \times col\_num + 128 \times (b\_tree\_idx\_num + text\_idx\_num) + row\_size \times 2)}{1{,}024} \right\rceil + 1 \right\}$$

col_num

Number of columns in the table to be processed

b_tree_idx_num

Number of B-tree indexes defined for the table to be processed

text_idx_num

Number of text indexes defined for the table to be processed

row_size

Row size in the table to be processed

Formula

$$row\_size = \sum_{i=1}^{col\_num} \left(\left\uparrow \frac{(coldata_{(i)} + 7)}{8} \right\uparrow \times 8 \right)$$

- $coldata_{(i)}$: Data length of each column

  Determine the value according to Table 6-9: Data length of each data type in (c) Determining the variable RTHD_EXESQLSZ under (2) Determining the real thread private memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

(2) Determining the real thread private memory requirement (for executing the adbgetcst command)

Use the following formula to determine the amount of real thread private memory (*RTHD_GETCOSTSZ*) required for executing the `adbgetcst` command.

Formula (kilobytes)

```
RTHD_GETCOSTSZ = COSTSQL
```

Explanation of variables

COSTSQL

Memory to be used by a retrieval SQL statement executed when collecting cost information

When the `adbgetcst` command is executed, the following SQL statement is executed in order to collect cost information from the table to be processed:

```
SELECT COUNT(*) FROM "target-table-name"
SELECT MIN("column-name-1"),MAX("column-name-1") FROM "target-table-name"
    :
SELECT MIN("column-name-n"),MAX("column-name-n") FROM "target-table-name"
SELECT * FROM "target-table-name"
```

Note: column-name-1 to column-name-*n* are the names of the columns defined for the target table.

Determine for each processing-target table the amount of memory required to execute a retrieval SQL statement by referring to (c) Determining the variable RTHD_EXESQLSZ and (g) Determining the variable RTHD_EXESQLDICSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation. Then substitute the largest among the determined values.

6.3.9 Determining the memory requirement for executing the adbdbstatus command

When executing the `adbdbstatus` command, the HADB server uses the types of memory listed below. Determine the requirement for each type of memory.

■ Shared memory

- Process common memory (*PROC_DBSTATUSSZ*)
- Real thread private memory (*RTHD_DBSTATUSSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbdbstatus command)

Use the following formula to determine the process common memory (*PROC_DBSTATUSSZ*) required for executing the `adbdbstatus` command.

Formula (kilobytes)

```
PROC_DBSTATUSSZ = Σ(PROC_MNG)#
```

#

If you execute multiple `adbdbstatus` commands concurrently, determine the memory requirement for each `adbdbstatus` command. Then, add up the total memory requirements.

Explanation of variables

PROC_MNG

Process common memory used for managing DB status analysis

Formula (kilobytes)

```
PROC_MNG = 1,353 + DICINF + DBINF + ARCINF#1 + PROC_AUDINFSZ#2
```

#1

Add this value if you specify `archivechunk` for the `-c` option of the `adbdbstatus` command.

#2

Add this value when the audit trail facility is enabled.

DICINF

Memory for managing definition information

Formula (kilobytes)

DICINF ＝ MAX(64 ， *A*)

A : The value is determined by the following formula:
↑(274,880＋120 × *dbarea_num*＋224 × *table_num*＋224 × *arctable_num*＋232 × *index_num*
＋232 × *arctable_num* × 3)÷1,024↑

- *dbarea_num*
  Number of data DB areas
- *table_num*
  Number of user-defined base tables
- *arctable_num*
  Number of user-defined base tables that are archivable multi-chunk tables
- *index_num*
  Number of user-defined indexes

DBINF

Memory for managing database information

Formula (kilobytes)

$$DBINF = \uparrow \frac{(264{,}600 + 56 \times file\_num + 88 \times chunk\_num + 24 \times uthd\_num)}{1{,}024} \uparrow$$

- *file_num*

  The largest number of DB area files among the DB areas to be analyzed and the DB areas storing the base tables and indexes to be analyzed

- *chunk_num*

  The largest number of chunks created in the base tables and indexes to be analyzed

  Determine the value as explained in (10) Determining the number of chunks created under C.9 Searching system tables.

- *uthd_num*

  Value specified for the `adb_sys_uthd_num` operand in the server definition

  If you do not specify the `-d reorginfo` option in the `adbdbstatus` command, substitute 0.

ARCINF

Memory for managing archivable multi-chunk tables

Formula (kilobytes)

$$ARCINF = \uparrow(1{,}012 + 580 \times col\_num + max\_rowsz + 860 \times idx\_num + 5{,}200) \div 1{,}024 \uparrow$$

col_num

Number of columns in the table to be processed

max_rowsz

Maximum row length (bytes)

Determine the maximum row length based on the formula for the row length *ROWSZ* in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

idx_num

Number of indexes defined for the table to be processed

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(2) Determining the real thread private memory requirement (for executing adbdbstatus command)

The amount of real thread private memory (*RTHD_DBSTATUSSZ*) required for executing the `adbdbstatus` command is as follows.

Value (kilobytes)

$$RTHD\_DBSTATUSSZ = 3$$

6.3.10 Determining the memory requirement for executing the adbexport command

When the `adbexport` command is executed, the HADB server uses the type of memory described in this subsection. Determine the requirement for each type of memory.

■ **Shared memory**

- Process common memory (*PROC_EXPORTSZ*)
- Real thread private memory (*RTHD_EXPORTSZ*)

■ **Process memory**

- Heap memory (*HEAP_EXPORTSZ*)

(1) Determining the process common memory requirement (for executing the adbexport command)

Use the following formula to determine the amount of process common memory (*PROC_EXPORTSZ*) required to execute the `adbexport` command.

Formula (kilobytes)

$$PROC\_EXPORTSZ = \sum (PROC\_MNG + PROC\_OUTFILE^{\#1})^{\#2}$$

#1

Add this value when the number of output data files specified for the output data path file name exceeds 1,024.

#2

If you execute multiple `adbexport` commands concurrently, determine the memory requirement for each `adbexport` command. Then, add up the total memory requirements.

Explanation of variables

PROC_MNG

Use the following formula to determine the value.

Formula (kilobytes)

$$PROC\_MNG = 3 + DIC^{\#1} + IOA + EXPF + PROC\_EXPSQLS + PROC\_AUDINFSZ^{\#2}$$

#1

Add this value if you specify the `-n` option for the `adbexport` command.

#2

Add this value when the audit trail facility is enabled.

DIC

Determine the value by referring to the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

IOA

Use the following formula to determine this value.

Formula (kilobytes)

$IOA = sql\_size + (scan\_buff\_size + offset\_area \times 2) \times scan\_rthd \times 2$

sql_size

Use the following formula to determine this value.

Formula

```
sql_size = (SQL-statement-file-size) x 2 + 1 + 1,024
```

scan_buff_size

Use the following formula to determine this value.

Formula

```
value-specified-for-export-option-adb_export_scan_buff_size x 1,024
```

offset_area

Use the following formula to determine this value.

Formula

```
(8 x number-of-columns-in-search-result x (scan_buff_size x 1,024 ÷ data-length-
in-search-result)) ÷ 1,024
```

scan_rthd

Use the following formula to determine the value:

Formula

```
↓(value-specified-for-export-option-adb_export_rthd_num - 1) ÷ 2↓
```

EXPF

Use the following formula to determine this value.

Formula (kilobytes)

$EXPF = 552 + (512 + 512^{\#}) \times exp\_rthd$

#

Add this value when you specify the `--compress GZIP` option for the `adbexport` command.

exp_rthd

Use the following formula to determine this value.

Formula

```
↓(value-specified-for-export-option-adb_export_rthd_num - 1) ÷ 2↓
```

PROC_EXPSQLS

When the `adbexport` command is executed, the following SQL statement is executed once to export data from the processing-target table.

If the -n option is specified

```
SELECT * FROM target-table-name;
```

If the -q option is specified

```
SELECT-statement-specified-in-the-SQL-statement-file
```

Determine for each processing-target table the amount of memory required to execute a retrieval SQL statement by referring to (c) Determining the variable PROC_EXECSQLSZ in (1) Determining the process common memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation. Then, substitute the largest of the determined values.

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

PROC_OUTFILE

This value must be added when the number of output data files specified for the output data path file name exceeds 1,024. Use the following formula to determine this value.

Formula (kilobytes)

PROC_OUTFILE＝↑（（120＋512）×（*OUTF*MAX－1,024））÷1,024↑

OUTFMAX

Number of output data files specified for the output data path file

(2) Determining the real thread private memory requirement (during execution of the adbexport command)

Use the following formula to determine the amount of real thread private memory (*RTHD_EXPORTSZ*) required to execute the `adbexport` command.

Formula (kilobytes)

RTHD_EXPORTSZ＝ *EXPORTSQL*

Explanation of variables

EXPORTSQL

When the `adbexport` command is executed, the following SQL statement is executed once to export data from the processing-target table.

If the -n option is specified

```
SELECT * FROM target-table-name;
```

If the -q option is specified

SELECT-statement-specified-in-the-SQL-statement-file

Determine for each processing-target table the amount of memory required to execute a retrieval SQL statement by referring to (c) Determining the variable RTHD_EXESQLSZ and (g) Determining the variable RTHD_EXESQLDICSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation. Then, substitute the largest of the determined values.

(3) Determining the heap memory requirement (for executing the adbexport command)

Use the following formula to determine the amount of heap memory (*HEAP_EXPORTSZ*) used to execute the `adbexport` command.

Formula (kilobytes)

$$HEAP\_EXPORTSZ = \sum(HEAP\_ZLIB)^{\#}$$

#

If you execute multiple `adbexport` commands concurrently, determine the memory requirement for each `adbexport` command. Then, add up the total memory requirements.

Explanation of variables

HEAP_ZLIB

If you specify the `--compress GZIP` option, use the following formula to determine this value.

Formula (kilobytes)

$$HEAP\_ZLIB = 262 \times file\_num$$

file_num

Number of output data files

6.3.11 Determining the memory requirement for executing the adbstat command

When executing the `adbstat` command, the HADB server uses the memory described in the following:

■ Process memory

- Heap memory (*STATMEMSZ*)

Use the following formula to determine the heap memory (*STATMEMSZ*) used to execute the `adbstat` command.

Formula (kilobytes)

$$STATMEMSZ = 1,273 \times exec\_num$$

Explanation of variables

exec_num: Number of `adbstat` commands to be executed concurrently

6.3.12 Determining the memory requirement for executing the adbmodarea command

When the `adbmodarea` command is executed, the HADB server uses the type of memory described in this subsection. Determine the requirement for each type of memory.

■ **Shared memory**

- Process common memory (*PROC_MODASZ*)
- Real thread private memory (*RTHD_MODASZ*)

■ **Process memory**

- Heap memory (*MODA_MEM*)

(1) Determining the process common memory requirement (for executing the adbmodarea command)

Use the following formula to determine the process common memory (*PROC_MODASZ*) required for executing the `adbmodarea` command.

Formula (kilobytes)

```
PROC_MODASZ = MODA_ROWS + MODA_CTL + MODA_DBH + PROC_AUDINFSZ# + 8
```

#

Add this value when the audit trail facility is enabled.

Explanation of variables

- *MODA_ROWS*
  Determine the value as explained in (a) Determining the variable MODA_ROWS.
- *MODA_CTL*
  Determine the value as explained in (b) Determining the variable MODA_CTL.
- *MODA_DBH*
  Determine the value as explained in (c) Determining the variable MODA_DBH.
- *PROC_AUDINFSZ*
  Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(a) Determining the variable MODA_ROWS

Use the following formula to determine the value of variable *MODA_ROWS*.

Formula (kilobytes)

```
MODA_ROWS = ↑ (256 x row_cnt) / 1,024 ↑
```

Explanation of variables

row_cnt

Number of rows in the dictionary table `SQL_DBAREAS`

(b) Determining the variable MODA_CTL

Use the following formula to determine the value of variable *MODA_CTL*.

Formula (kilobytes)

```
MODA_CTL=
    8,192 + MODA_REQ + ↑ ( (8 x row_cnt)
    + (64 x (bdev_num_new + bdev_num_alloc))
    + (8 x bdev_num_area) + (256 x bdev_num_file)) / 1,024 ↑
```

Explanation of variables

MODA_REQ

See the description of the variable *MODA_REQ* in (2) Determining the real thread private memory requirement (for executing the adbmodarea command).

row_cnt

Number of rows in the dictionary table `SQL_DBAREAS`

bdev_num_new

Number of new block special files to be allocated

bdev_num_alloc

Number of already allocated block special files

bdev_num_area

Number of DB areas in which block special files are allocated

bdev_num_file

Total number of DB area files in DB areas for which block special files are allocated

(c) Determining the variable MODA_DBH

Use the following formula to determine the value of variable *MODA_DBH*.

Formula (kilobytes)

■ When a DB area is to be added

$$MODA\_DBH = \sum (239 + (add\_dbarea\_file\_num - 1) \times 143 + add\_dbarea\_file\_num \times 329)^{\#}$$

■ When a DB area is to be expanded (DB area files are to be added)

$$MODA\_DBH = \sum (115 + (add\_dbarea\_file\_num\_all - 1) \times 107 + add\_dbarea\_file\_num \times 365)^{\#}$$

#

If you add or expand multiple DB areas, determine the value for each DB area. Then, add up the determined values.

Explanation of variables

add_dbarea_file_num

Number of DB area files added when DB areas are added or expanded by using the `adbmodarea` command

add_dbarea_file_num_all

Total combined number of existing DB area files and DB area files added by expansion of DB areas

(2) Determining the real thread private memory requirement (for executing the adbmodarea command)

Use the following formula to determine the amount of real thread private memory (*RTHD_MODASZ*) required to execute the `adbmodarea` command.

Formula (kilobytes)

$$RTHD\_MODASZ = 20{,}480 + MODA\_REQ$$

Explanation of variables

MODA_REQ

Use the following formula to determine this value.

$$MODA\_REQ = 3 + \uparrow (256 \times bdev\_num\_new) \div 1{,}024 \uparrow$$

bdev_num_new

Number of new block special files to be allocated

(3) Determining the heap memory requirement (for executing the adbmodarea command)

Use the following formula to determine the heap memory (*MODA_MEM*) required to execute the `adbmodarea` command.

Formula (megabytes)

$$MODA\_MEM = \uparrow \frac{\{17 + bdev\_num\_new\}}{1{,}024} \uparrow$$

Explanation of variables

bdev_num_new

Number of new block special files to be allocated

Important

When you use the `adbmodarea` command to add or delete a DB area, the memory requirements listed below change. Therefore, estimate them again.

- **Global buffer page requirement (variable SHM_BUFGLOBAL)**

  See (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- **Process common memory requirement (variable BUFGLOBAL)**

  See (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

6.3.13 Determining the memory requirement for executing the adbmergechunk command

When the `adbmergechunk` command is executed, the HADB server uses the types of memory described in this subsection. You need to determine the memory requirement for each type.

■ **Shared memory**

- Process common memory (*PROC_MERGCSZ*)
- Real thread private memory (*RTHD_MERCHKSZ*)

■ **Process memory**

- Heap memory (*HEAP_MERCHKSZ*)

The following subsections describe the formulas for determining these memory requirements.

(1) Determining the process common memory requirement (for executing the adbmergechunk command)

Use the following formula to determine the amount of process common memory (*PROC_MERGCSZ*) needed to execute the `adbmergechunk` command.

Formula (kilobytes)

$$PROC\_MERGCSZ =$$
$$\Sigma(PROC\_MNG + PROC\_IDX^{\#1} + PROC\_IDXREBUILD^{\#1} + PROC\_RNGIDX^{\#2}$$
$$+ PROC\_CHUNK\_INFO + PROC\_DBUPDINF + PROC\_RTHDUPDINF)^{\#3}$$

Explanation of variables

- *PROC_MNG*
  Determine the value as explained in (a) Determining the variable PROC_MNG.
- *PROC_IDX*
  Determine the value as explained in (b) Determining the variable PROC_IDX.
- *PROC_IDXREBUILD*
  Determine the value as explained in (c) Determining the variable PROC_IDXREBUILD.
- *PROC_RNGIDX*
  Determine the value as explained in (d) Determining the variable PROC_RNGIDX.
- *PROC_CHUNK_INFO*
  Determine the value as explained in (e) Determining the variable PROC_CHUNK_INFO.
- *PROC_DBUPDINF*
  Determine the value as explained in (f) Determining the variable PROC_DBUPDINF.
- *PROC_RTHDUPDINF*
  Determine the value as explained in (g) Determining the variable PROC_RTHDUPDINF.

#1

Add this value when the index to be processed contains a B-tree index.

#2

Add this value when the index to be processed contains a range index.

#3

If you execute multiple `adbmergechunk` commands concurrently, determine the memory requirement for each `adbmergechunk` command. Then, add up the total memory requirements.

(a) Determining the variable PROC_MNG

Use the following formula to determine the value of variable *PROC_MNG*.

Formula (kilobytes)

```
PROC_MNG =
    5 + DIC + IOA + SCAN + IDXRBLDBUF_CTL + STS + PAGEALLOC + CHUNK + PROC_AUDINFSZ#
```

#

Add this value when the audit trail facility is enabled.

Explanation of variables

DIC: Memory for acquiring the definition information of the table to be processed

See the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

IOA: Memory used to access the database

Use the following formula to determine its value.

Formula (kilobytes)

```
IOA = sql_size + (scan_buff_size + offset_area × 2) × scan_rthd
```

- *sql_size*

  Use the following formula to determine its value:

  ```
  32 + 1,024
  ```

- *scan_buff_size*

  Use the following formula to determine its value:

  value-specified-for-merge-chunk-option-adb_mergechunk_scan_buff_size × 1,024

- *offset_area*

  Use the following formula to determine its value.

  (8 × *number-of-columns-comprising-processing-target-index* × (*scan_buff_size* × 1,024
  ÷ (*sum-total-of-lengths-of-columns-comprising-processing-target-index* + 8))) ÷ 1,024

- *scan_rthd*

  Use the following formula to determine the value:

  ```
  ↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
  ```

SCAN: Memory needed to output the index record file

Use the following formula to determine its value.

Formula (kilobytes)

```
SCAN =
    ↑(write_size × scan_rthd × b-tree_index_num)↑
```

- *write_size*
  Value specified for the merge chunk option adb_mergechunk_dvix_wtbuff_size
- *scan_rthd*
  Use the following formula to determine the value:

  ```
  ↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
  ```

- *b-tree_index_num*
  Number of B-tree indexes to be processed

IDXRBLDBUF_CTL: Buffer control information for merging chunks

Use the following formula to determine its value.

Formula (kilobytes)

```
IDXRBLDBUF_CTL =
    ↑(248 + BUFBLK) ÷ 1,024↑ × scan_rthd
```

- *BUFBLK*
  For details, see the description of the variable *BUFBLK* in (f) Determining the variable PROC_UPDSZ under (1) Determining the process common memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.
- *scan_rthd*
  Use the following formula to determine the value:

  ```
  ↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
  ```

STS: Memory for saving status information

Use the following formula to determine its value.

Formula (kilobytes)

$$STS = \uparrow(256 + (1{,}185 \times idx\_num) + (2{,}067 \times (b\_tree\_idx\_num + range\_idx\_num) \times scan\_rthd + 2{,}067 \times text\_idx\_num \times 16)) \div 1{,}024\uparrow$$

idx_num
Number of indexes to be processed

b_tree_idx_num
Number of B-tree indexes defined for the table to be processed

range_idx_num
Number of range indexes defined for the table to be processed

scan_rthd
Use the following formula to determine the value:

```
↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
```

text_idx_num
Number of text indexes defined for the table to be processed

PAGEALLOC: Page allocation control information for merging chunks

Use the following formula to determine its value.

Formula (kilobytes)

```
PAGEALLOC = dividx_rthd × 310
```

- *dividx_rthd*

  Use the following formula to determine the value:

  ```
  value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
  ```

CHUNK: Chunk management block

Use the following formula to determine its value.

Formula

```
CHUNK = ↑(8 × (chunk_num × 2) + 30,000) ÷ 1,024↑
```

- *chunk_num*

  Number of chunks created in the table to be processed

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(b) Determining the variable PROC_IDX

Use the following formula to determine the value of variable *PROC_IDX*.

Formula (kilobytes)

$$
\begin{aligned}
&PROC\_IDX = \\
&\quad \uparrow((13{,}151 + (33{,}928 + (13 \times KEYSZ)) \times scan\_rthd \\
&\qquad + (152 + rd\_buff\_size \times scan\_rthd \times 2 \times 1{,}024 + 2 \times ld\_buff\_size \times 1{,}024 \\
&\qquad + (KEYSZ + CTRL) \times text\_idx\_num) \times dividx\_rthd) \\
&\qquad \times (b\_tree\_idx\_num + text\_idx\_num)) \div 1{,}024\uparrow
\end{aligned}
$$

Explanation of variables

KEYSZ

Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length. If multiple B-tree indexes have been defined for the processing-target table, use the largest key length among all the B-tree indexes.

scan_rthd

Use the following formula to determine the value:

```
↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
```

rd_buff_size

Value specified for the merge chunk option `adb_mergechunk_dvix_rdbuff_size`

ld_buff_size

Value specified for the merge chunk option `adb_mergechunk_dvix_wtbuff_size`

CTRL

- If the indexed columns are fixed-length columns
  Substitute 10 bytes.
- If the indexed columns include variable-length columns
  Substitute 12 bytes.

text_idx_num

Number of text indexes defined for the table to be processed

dividx_rthd

Use the following formula to determine the value:

```
value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
```

b_tree_idx_num

Number of B-tree indexes to be processed

(c) Determining the variable PROC_IDXREBUILD

Use the following formula to determine the variable *PROC_IDXREBUILD*.

Formula (kilobytes)

$$PROC\_IDXREBUILD = \max_{1 \leq k \leq idx\_num} PROC\_IDXREBUILD\_MEM_{(k)}$$

idx_num

Number of B-tree indexes to be rebuilt

PROC_IDXREBUILD_MEM(k)

Use the following formula to determine its value.

Formula (kilobytes)

```
PROC_IDXREBUILD_MEM(k) =
  ↑(184 + 256 × idx_col_num + (18 + 256 × ↑idx_lv ÷ 16↑) × dividx_rthd
    + KEYSZ + (idx_col_num + 1) × 2 + 4) ÷ 1,024↑
```

- *idx_col_num*
  Number of indexed columns in a B-tree index
- *idx_lv*
  Number of levels in the B-tree index
  Calculate the recursion formula *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment and substitute the value of *n* when *PIDX(n)* results in 1.
  Note that in the case of *PIDX(1)* = 1, the number of levels in the B-tree index is 2.
- *dividx_rthd*
  Use the following formula to determine the value:

  ```
  value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
  ```
- *KEYSZ*: Key length of a B-tree index (bytes)
  Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

(d) Determining the variable PROC_RNGIDX

Use the following formula to determine the value of variable *PROC_RNGIDX*.

Formula (kilobytes)

PROC_RNGIDX = ↑64 × *rngidx_num* ÷ 1,024↑

Explanation of variables

rngidx_num: Number of range indexes to be rebuilt

(e) Determining the variable PROC_CHUNK_INFO

Use the following formula to determine the value of variable *PROC_CHUNK_INFO*.

Formula (kilobytes)

```
PROC_CHUNK_INFO = ↑(1 + index_num) × 120 × (chunk_num + 1) ÷ 1,024↑
```

Explanation of variables

index_num: Number of indexes defined for the table to be processed by the merge chunk command

chunk_num: Maximum number of chunks in the table to be processed by the merge chunk command

When the `adbmergechunk` command is executed, the amount of process common memory determined by the variable *PROC_CHUNK_INFO* is allocated. When the `adbmergechunk` command finishes, the allocated process common memory is released.

However, if *deletion-pending chunks* remain for any of the following reasons, the allocated process common memory is not released. It is released when all delete-pending chunks are deleted.

- The `adbmergechunk` command is interrupted after the merge-target chunk is ready but before the merge-source chunks are deleted

  For example, merge-source chunks might remain undeleted if the HADB server has insufficient memory or when the `adbcancel` command is executed.
- A merge-chunk target table is already being referenced when the `adbmergechunk` command is executed with `NOWAIT` specified in the `--purge-chunk` option

(f) Determining the variable PROC_DBUPDINF

Use one of the formulas below to determine the value of variable *PROC_DBUPDINF*. Note that the formula to use differs depending on whether the multi-node function is being used.

Formula when the multi-node function is not used (kilobytes)

PROC_DBUPDINF=
↑（（40＋↑（*DBUPDINF_ENT_NUM*÷1,024）↑×131,136）÷1,024）↑

Formula when the multi-node function is used (kilobytes)

PROC_DBUPDINF=
↑（（80＋（↑（*DBUPDINF_ENT_NUM*÷1,024）↑＋1）×131,136）÷1,024）↑

Explanation of variables

DBUPDINF_ENT_NUM

Number of entries in the DB area, table, index, and chunk update information

Use one of the formulas below to determine its value. Note that the formula to use differs depending on whether the multi-node function is used.

Formula to be used when the multi-node function is not used (entries)

```
DBUPDINF_ENT_NUM = 1 + 1 + index_num
```

Formula to be used when the multi-node function is used (entries)

```
DBUPDINF_ENT_NUM= 3+ (1+index_num) × chunk_num+ (index_num × 6)
```

index_num

Number of indexes defined for the table to be processed by the merge chunk command

chunk_num

Number of chunks to be merged

(g) Determining the variable PROC_RTHDUPDINF

Use the following formula to determine the value of variable *PROC_RTHDUPDINF*.

Formula (kilobytes)

```
PROC_RTHDUPDINF =
    ↑(648 + (168 × rthd_num)) ÷ 1,024↑
```

Explanation of variables

rthd_num

Use the following formula to determine its value.

```
value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
```

(2) Determining the real thread private memory requirement (for executing the adbmergechunk command)

Use the following formula to determine the real thread private memory requirement (*RTHD_MERCHKSZ*) for executing the `adbmergechunk` command.

Formula (kilobytes)

```
RTHD_MERCHKSZ=
    IDXRBLDBUF+RTHD_SCAN+SORTIOBUF#1+SORTBUF#1+RTHD_IDXREC#1
    +RTHD_IDXREBUILD#1+RTHD_RNGIDX#2+RTHD_TXTIDX#3
```

Explanation of variables

- *IDXRBLDBUF*

  Determine the value as explained in (a) Determining the variable IDXRBLDBUF.
- *RTHD_SCAN*

  Determine the value as explained in (b) Determining the variable RTHD_SCAN.

- *SORTIOBUF*
  Determine the value as explained in (c) Determining the variable SORTIOBUF.
- *SORTBUF*
  Determine the value as explained in (d) Determining the variable SORTBUF.
- *RTHD_IDXREC*
  Determine the value as explained in (e) Determining the variable RTHD_IDXREC.
- *RTHD_IDXREBUILD*
  Determine the value as explained in (f) Determining the variable RTHD_IDXREBUILD.
- *RTHD_RNGIDX*
  Determine the value as explained in (g) Determining the variable RTHD_RNGIDX.
- *RTHD_TXTIDX*
  Determine the value as explained in (h) Determining the variable RTHD_TXTIDX.

#1

Add this value when a B-tree index is to be processed.

#2

Add this value when a range index is included among the indexes to be processed.

#3

Add this value when a text index is included among the indexes to be processed.

(a) Determining the variable IDXRBLDBUF

Use the following formula to determine the value of variable *IDXRBLDBUF*.

Formula (kilobytes)

```
IDXRBLDBUF =
  ↑(RBLD_SQBLK + RBLD_SQIO + RBLD_SQPGE + RBLD_SQHS) ÷ 1,024↑
```

Explanation of variables

RBLD_SQBLK: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQBLK =
  rbld_blknum × (112 + ↑RBLD_BLKSZ ÷ rbld_pagesize ÷ 64↑ × 24)
```

- *rbld_blknum*
  Value specified for the merge chunk option `adb_mergechunk_buff_blk_num`
- *RBLD_BLKSZ*
  Use the following formula to determine its value:

  ```
  4,096 × 1,024
  ```

- *rbld_pagesize*
  The smallest of the page sizes of the DB areas storing the B-tree indexes or text indexes that are defined for the processing-target table (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

RBLD_SQIO: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQIO = 568 × rbld_blknum
```

- *rbld_blknum*

  Value specified for the merge chunk option `adb_mergechunk_buff_blk_num`

RBLD_SQPGE: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQPGE =
  (176 + rbld_pagesize + BUFLOG) × (RBLD_BLKSZ ÷ rbld_pagesize) × rbld_blknum
```

- *rbld_pagesize*

  The smallest of the page sizes of the DB areas storing the B-tree indexes or text indexes that are defined for the processing-target table (bytes)

  Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *BUFLOG*

  For details, see the description of the variable *BUFLOG* in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *RBLD_BLKSZ*

  Use the following formula to determine its value:

  ```
  4,096 × 1,024
  ```

- *rbld_blknum*

  Value specified for the merge chunk option `adb_mergechunk_buff_blk_num`

RBLD_SQHS: Use the following formula to determine its value.

Formula (bytes)

```
RBLD_SQHS = 8 × rbld_blknum + 40 × rbld_blknum
```

- *rbld_blknum*

  Value specified for the merge chunk option `adb_mergechunk_buff_blk_num`

(b) Determining the variable RTHD_SCAN

Use the following formula to determine the value of the variable *RTHD_SCAN*.

Formula (kilobytes)

```
RTHD_SCAN = MAX( RTHD_EXESQLSZ , RTHD_EXESQLDICSZ )
```

Note

Determine the values of the variables *RTHD_EXESQLSZ* and *RTHD_EXESQLDICSZ* on the assumption that the following SQL statement were executed:

```
SELECT selection-expression# FROM "name-of-table-to-be-processed"
```

#

Determine the value on the basis of CHAR(16) columns and as if duplicates were removed from all indexed columns of the indexes defined for the table to be processed.

Explanation of variables

RTHD_EXESQLSZ

See (c) Determining the variable RTHD_EXESQLSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

RTHD_EXESQLDICSZ

See (g) Determining the variable RTHD_EXESQLDICSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation.

(c) Determining the variable SORTIOBUF

Use the following formula to determine the value of variable *SORTIOBUF*.

Formula (kilobytes)

```
SORTIOBUF = sort_io_buff_size × sort_rthd
```

Explanation of variables

- *sort_io_buff_size*
  Substitute 16.
- *sort_rthd*
  Use the following formula to determine the value:
  ```
  value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
  ```

(d) Determining the variable SORTBUF

Use the following formula to determine the value of variable *SORTBUF*.

Formula (kilobytes)

```
SORTBUF = (48 + sort_buff_size × 1,024) × sort_rthd
```

Explanation of variables

sort_rthd

Use the following formula to determine the value:

```
value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
```

sort_buff_size

Value specified for the merge chunk option adb_mergechunk_sort_buff_size#

#

A work file is created during sort processing. The formula for the variable *sort_buff_size*, which minimizes the size of this file, is shown below. Allocate at least the amount of space indicated by the formula. However, if you are

running out of memory or if most data items are already arranged in index key order, do not specify more space than is necessary.

Formula

$$sort\_buff\_size \geq \left\lceil \left(\frac{REC\_SIZE + 7}{2} + \sqrt{(BLK + 8) \times row\_num \times KEY\_INF + \frac{REC\_SIZE + 7}{4}} + COL\_INFO \right) \div 1{,}024 \right\rceil$$

Explanation of variables

REC_SIZE: Use the following formula to determine its value.

Formula

```
REC_SIZE = (KEYSZ + SYS1) + SYS2
```

KEYSZ: Index key length of the B-tree index defined for the table to be processed

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. If multiple B-tree indexes have been defined for the processing-target table, use the largest key length among all the B-tree indexes.

SYS1: Assume one of the following values:

- If the indexed columns of the indexes defined for the table to be processed are fixed-length: 10
- If any of the indexed columns of the indexes defined for the table to be processed are variable-length: 12

SYS2: Assume one of the following values:

- If the indexes defined for the table to be processed are multiple-column indexes

  Use the following formula to determine its value:

  (*number of indexed columns* × 4)

- For all other indexes

  0

BLK: Use the following formula to determine its value.

Formula

```
BLK = REC_SIZE + (KEYSZ + 8) + 56
```

row_num

Number of rows stored in the table to be processed

KEY_INF: Use the following formula to determine its value.

Formula

```
KEY_INF = REC_SIZE + (KEYSZ + 8) + 28
```

COL_INFO: Use the following formula to determine its value.

Formula

```
COL_INFO = 2,112 + (MULTI_KEY_INFO × 32) + (KEYSZ + 8)
```

MULTI_KEY_INFO: Assume one of the following values:

- If the B-tree indexes defined for the table to be processed are variable-length multiple-column indexes

  Use the following formula to determine its value:

(*number of indexed columns* × 2) + 2

- For all other B-tree indexes

5

(e) Determining the variable RTHD_IDXREC

Use the following formula to determine the value of variable *RTHD_IDXREC*.

Formula (kilobytes)

RTHD_IDXREC =
↑(386 + 328 × (*idx_num* - 1) + 32 × *max_idx_col_num*
+ (64 + *buf_size* × 1,024) × 2 × *idx_num*) ÷ 1,024↑

Explanation of variables

- *idx_num*
  Number of B-tree indexes to be rebuilt
- *max_idx_col_num*
  Maximum number of indexed columns in the B-tree index to be rebuilt
- *buf_size*
  Substitute 1,024.

(f) Determining the variable RTHD_IDXREBUILD

Use the following formula to determine the value of variable *RTHD_IDXREBUILD*.

Formula (kilobytes)

$$RTHD\_IDXREBUILD = \max_{1 \le k \le idx\_num} RTHD\_IDXREBUILD\_MEM_{(k)}$$

Explanation of variables

idx_num
Number of B-tree indexes to be rebuilt

RTHD_IDXREBUILD_MEM(*k*)
Use the following formula to determine its value.

Formula (kilobytes)

```
RTHD_IDXREBUILD_MEM (k) =
  ↑ (61,458 + ↑MAX (255, KEYSZ) ÷ 2↑ × 2 + 3,070
   + ↑KEYSZ + CTRL ÷ 8↑ × 8 + MAX (255, KEYSZ) + 14
   + ↓8 + page_size × 95 ÷ 100↓ × 2) ÷ 1,024↑
```

KEYSZ
Key length of the B-tree index (bytes)
Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL

- When all key lengths in the indexed columns are fixed

Substitute 10 bytes.

- When some of the key lengths in the indexed columns are variable
  Substitute 12 bytes.

page_size

Page size of the data DB area in which the B-tree index is defined (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(g) Determining the variable RTHD_RNGIDX

Use the following formula to determine the value of variable *RTHD_RNGIDX*.

Formula (kilobytes)

```
RTHD_RNGIDX = ↑(16 + 128 × rngidx_num) × scan_rthd ÷ 1,024↑
```

Explanation of variables

- *rngidx_num*
  Number of range indexes to be rebuilt
- *scan_rthd*
  Use the following formula to determine the value:

```
↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
```

(h) Determining the variable RTHD_TXTIDX

Use the following formula to determine the value of variable *RTHD_TXTIDX*.

Formula (kilobytes)

$$RTHD\_TXTIDX = \max_{1 \le k \le idx\_num} RTHD\_TXTIDXBUILD\_MEM_{(k)}$$

Explanation of variables

idx_num

Number of text indexes defined for the table to be processed

$RTHD\_TXTIDXBUILD\_MEM_{(k)}$

Use the following formula to determine its value.

Formula (kilobytes)

$$RTHD\_TXTIDXBUILD\_MEM_{(k)} = buff\_blk\_size \times buff\_blk\_num + 3{,}072 + \uparrow(idx\_div\_num \times 24) \div 1{,}024\uparrow + txt\_sort\_buff\_size \times 1{,}024$$

buff_blk_size

Substitute 4,096.

buff_blk_num

Value specified for the `adb_mergechunk_buff_blk_num` merge chunk option

idx_div_num

Length defined for the indexed column for text index *k*

txt_sort_buff_size

Value specified for the `adb_mergechunk_txt_buff_size` merge chunk option

(3) Determining the heap memory requirement (for executing the adbmergechunk command)

Use the following formula to determine the heap memory (*HEAP_MERCHKSZ*) required when executing the `adbmergechunk` command.

Formula (kilobytes)

```
HEAP_MERCHKSZ = Σ (505 x sort_rthd)#
```

#

If you execute multiple `adbmergechunk` commands concurrently, determine the amount of heap memory used by each `adbmergechunk` command. Then, add up the total heap memory requirements.

Explanation of variables

sort_rthd

Use the following formula to determine the value:

```
value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1
```

6.3.14 Determining the memory requirement for executing the adbchgchunkcomment command

When you execute the `adbchgchunkcomment` command, the HADB server uses the following type of memory:

■ Shared memory

- Process common memory (*PROC_CHGCCMSZ*)

Use the following formula to determine the process common memory (*PROC_CHGCCMSZ*) used to execute the `adbchgchunkcomment` command.

Formula (kilobytes)

```
PROC_CHGCCMSZ= Σ(5 + DIC + PROC_AUDINFSZ#1)#2
```

#1

Add this value when the audit trail facility is enabled.

#2

If you execute multiple `adbchgchunkcomment` commands concurrently, determine the memory requirement for each `adbchgchunkcomment` command. Then, add up the total memory requirements.

Explanation of variables

DIC

Memory for acquiring the definition information of the table to be processed

Determine the value by referring to the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

6.3.15 Determining the memory requirement for executing the adbchgchunkstatus command

When you execute the `adbchgchunkstatus` command, the HADB server uses the following type of memory:

■ Shared memory

- Process common memory (*PROC_CHGCSTSZ*)

Use the following formula to determine the process common memory (*PROC_CHGCSTSZ*) used to execute the `adbchgchunkstatus` command.

Formula (kilobytes)

$$PROC\_CHGCSTSZ = PROC\_MCS + PROC\_DBUPDINF^{\#}$$

#

Add this value when you use the multi-node function.

Explanation of variables

PROC_MCS

Process common memory to be used for controlling chunk status changes

Formula (kilobytes)

$$PROC\_MCS = 5 + DIC + \uparrow (30 \times chunk\_num) \div 1{,}024 \uparrow + PROC\_AUDINFSZ^{\#}$$

#

Add this value when the audit trail facility is enabled.

DIC

Memory for acquiring the definition information of the table to be processed

Determine the value by referring to the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

chunk_num

Number of chunks created in the table to be processed

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

PROC_DBUPDINF

Process common memory required for storing DB area, table, index, and chunk update information

Value (kilobytes)

PROC_DBUPDINF= 129

6.3.16 Determining the memory requirement for executing the adbarchivechunk command

When the `adbarchivechunk` command is executed, the HADB server uses the following types of memory. Determine the requirement for each type of memory.

- **Shared memory**
  - Process common memory (*PROC_ARCCKSZ*)
  - Real thread private memory (*RTHD_ARCCKSZ*)
- **Process memory**
  - Heap memory (*HEAP_ARCCKSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbarchivechunk command)

Use the following formula to determine the process common memory (*PROC_ARCCKSZ*) required for executing the `adbarchivechunk` command.

Formula (kilobytes)

$$PROC\_ARCCKSZ = \sum(PROC\_ARCCON + (PROC\_MAC + PROC\_OUTFILE + PROC\_DBUPDINF)^{\#1})^{\#2}$$

Explanation of variables

PROC_ARCCON

Determine the value as explained in (a) Determining the variable PROC_ARCCON.

PROC_MAC

Determine the value as explained in (b) Determining the variable PROC_MAC.

PROC_OUTFILE

Determine the value as explained in (c) Determining the variable PROC_OUTFILE.

PROC_DBUPDINF

Determine the value as explained in (d) Determining the variable PROC_DBUPDINF.

#1

If you execute the `adbarchivechunk` command for multiple chunks, determine the memory requirement for each chunk. Then, add the largest of the determined values to this variable.

#2

If you execute multiple `adbarchivechunk` commands concurrently, determine the memory requirement for each `adbarchivechunk` command. Then, add up the total memory requirements.

(a) Determining the variable PROC_ARCCON

Use the following formula to determine the value of variable *PROC_ARCCON*.

Formula (kilobytes)

PROC_ARCCON＝↑（250,000＋800）÷1,024↑＋*PROC_AUDINFSZ*#

#

Add this value when the audit trail facility is enabled.

Explanation of variables

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(b) Determining the variable PROC_MAC

Use the following formula to determine the value of variable *PROC_MAC*.

Formula (kilobytes)

PROC_MAC＝ 3＋*DIC*＋*IOA*＋*EXPF*＋*PROC_EXPSQLS*＋*SGMTGRP*

Explanation of variables

DIC

See the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

IOA

Use the following formula to determine this value.

Formula (kilobytes)

IOA＝ *sql_size*＋（*scan_buff_size*＋*offset_area*×2）×*scan_rthd*×2

sql_size

Use the following formula to determine this value.

Formula

sql_size＝32＋1,024

scan_buff_size

Use the following formula to determine this value.

Formula

scan_buff_size = *value-specified-for-archive-chunk-option-adb_arcv_scan_buff_size* x 1,024

offset_area

Use the following formula to determine this value.

Formula

offset_area = (8 x *number-of-columns-in-target-table* x (*scan_buff_size* x 1,024
÷ *data-length-of-target-table*)) ÷ 1,024

scan_rthd

Use the following formula to determine the value:

↓(*value-specified-for-archive-chunk-option-adb_arcv_rthd_num* - 1) ÷ 2↓

EXPF

Use the following formula to determine this value.

Formula (kilobytes)

EXPF= 552＋（512＋512）×*exp_rthd*

exp_rthd

Use the following formula to determine this value.

↓(*value-specified-for-archive-chunk-option-adb_arcv_rthd_num* - 1) ÷ 2↓

PROC_EXPSQLS

When the `adbarchivechunk` command is executed, the following SQL statement is executed once to archive the data in a chunk.

```
SELECT * FROM target-table-name;
```

Determine for each processing-target table the amount of memory required to execute a retrieval SQL statement by referring to (c) Determining the variable PROC_EXECSQLSZ in (1) Determining the process common memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation. Then, substitute the largest of the determined values.

SGMTGRP

Use the following formula to determine this value.

Formula (kilobytes)

SGMTGRP= *SGMTIDLIST*＋*SGMTGRPLIST*＋*BINTREE*

SGMTIDLIST

Use the following formula to determine this value.

Formula (kilobytes)

SGMTIDLIST= ↑(*max_sgmtnum_in_chunks*×16)÷1,024↑

max_sgmtnum_in_chunks

Specify the number of segments in the largest table among the chunks to be archived.

Note that the number of index segments must be excluded.

To check the number of table segments for each chunk, use either of the following methods:

- Execute the `adbdbstatus` command with the `-d used` and `-c table` options specified to output the information about the usage of DB areas, tables, and indexes. Then, obtain the number of table segments for each chunk from the value of `Used_segments` in the output results.
- Use the following formula to determine the value of the variable *chunk_sgmtnum* for each chunk to be archived. Then, substitute the largest of the determined *chunk_sgmtnum* values.

  Formula

$$chunk\_sgmtnum = \left\uparrow \frac{CHBP}{SEGSIZE} \right\uparrow + \left\uparrow \frac{CHVP}{SEGSIZE} \right\uparrow$$

chunk_sgmtnum

Specify the number of table segments stored in the chunk.

Note that the number of index segments must be excluded.

CHBP

See *CHBP(i,k)* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

CHVP

See *CHVP(i,k)* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

SEGSIZE

See *SEGSIZE* in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area.

page_size

See Table 6-3: DB area page size in (2) Determining the global buffer page requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

SGMTGRPLIST

Use the following formula to determine this value.

Formula (kilobytes)

$$SGMTGRPLIST = \uparrow(SGMTGRPNUM \times 312) \div 1{,}024\uparrow$$

SGMTGRPNUM

The value of the variable *SGMTGRPNUM* changes according to the value of the variable *max_sgmtnum_in_chunks*. The following table shows the relationship between the variable *max_sgmtnum_in_chunks* and the variable *SGMTGRPNUM*.

Table 6-13: Relationship between the variables max_sgmtnum_in_chunks and SGMTGRPNUM

No.	Value of the variable max_sgmtnum_in_chunks	Value of the variable SGMTGRPNUM
1	`max_sgmtnum_in_chunks` < 16	For *SGMTGRPNUM*, substitute 1.
2	16 ≤ `max_sgmtnum_in_chunks` and `max_sgmtnum_in_chunks` < 32	For *SGMTGRPNUM*, substitute 4.
3	32 ≤ `max_sgmtnum_in_chunks`	For *SGMTGRPNUM*, substitute 8.

No.	Value of the variable max_sgmtnum_in_chunks	Value of the variable SGMTGRPNUM
	and max_sgmtnum_in_chunks < 64	
4	64 ≤ max_sgmtnum_in_chunks and max_sgmtnum_in_chunks < 256	For *SGMTGRPNUM*, substitute 16.
5	256 ≤ max_sgmtnum_in_chunks	Use the following formula to determine the value of *SGMTGRPNUM*: *SGMTGRPNUM*= MAX(↓ *max_sgmtnum_in_chunks*÷256 ↓ , 32)

BINTREE

Use the following formula to determine this value.

Formula (kilobytes)

BINTREE= ↑(*SGMTGRPNUM*×240)÷1,024↑

Use the formula in Table 6-13: Relationship between the variables max_sgmtnum_in_chunks and SGMTGRPNUM to calculate the variable *SGMTGRPNUM*.

(c) Determining the variable PROC_OUTFILE

Use the following formula to determine the value of variable *PROC_OUTFILE*.

Formula (kilobytes)

PROC_OUTFILE= ↑((120+512)×(*OUTF*MAX−1,024))÷1,024↑

Explanation of variables

OUTFMAX

Use the following formula to determine this value.

↓(*value-specified-for-archive-chunk-option-adb_arcv_rthd_num* - 1) ÷ 2↓

(d) Determining the variable PROC_DBUPDINF

Use the following formula to determine the value of variable *PROC_DBUPDINF*. Note that the formula to be used differs depending on whether the multi-node function is used.

Formula to be used when the multi-node function is not used (kilobytes)

PROC_DBUPDINF=
↑（（40+↑（*DBUPDINF_ENT_NUM*÷1,024）↑×122,888）÷1,024）↑

Formula to be used when the multi-node function is used (kilobytes)

PROC_DBUPDINF=
129+↑（（40+↑（*DBUPDINF_ENT_NUM*÷1,024）↑×122,888）÷1,024）↑

Explanation of variables

DBUPDINF_ENT_NUM

Use the following formula to determine this value.

Formula (count)

```
DBUPDINF_ENT_NUM= SGMTGRPNUM× scan_rthd_num
```

SGMTGRPNUM

See Table 6-13: Relationship between the variables max_sgmtnum_in_chunks and SGMTGRPNUM in (b) Determining the variable PROC_MAC.

scan_rthd_num

Use the following formula to determine the value:

```
↓(value-specified-for-archive-chunk-option-adb_arcv_rthd_num - 1) ÷ 2↓
```

(2) Determining the real thread private memory requirement (for executing the adbarchivechunk command)

Use the following formula to determine the amount of real thread private memory (*RTHD_ARCCKSZ*) required to execute the `adbarchivechunk` command.

Formula (kilobytes)

```
RTHD_ARCCKSZ= RTHD_ARCSQL+RTHD_ARCDIRPATH
```

Explanation of variables

RTHD_ARCSQL

Determine the value as explained in (a) Determining the variable RTHD_ARCSQL.

RTHD_ARCDIRPATH

Determine the value as explained in (b) Determining the variable RTHD_ARCDIRPATH.

(a) Determining the variable RTHD_ARCSQL

The following describes how to determine the value of the variable *RTHD_ARCSQL*.

When the `adbarchivechunk` command is executed, the following SQL statement is executed once to archive the data in a chunk.

```
SELECT * FROM target-table-name;
```

Determine for each processing-target table the amount of memory required to execute a retrieval SQL statement by referring to (c) Determining the variable RTHD_EXESQLSZ and (g) Determining the variable RTHD_EXESQLDICSZ in (2) Determining the real thread private memory requirement (during normal operation) under 6.3.4 Determining the memory requirement during normal operation. Then, substitute the largest of the determined values.

(b) Determining the variable RTHD_ARCDIRPATH

Use the following formula to determine the value of variable *RTHD_ARCDIRPATH*.

Formula (kilobytes)

$$RTHD\_ARCDIRPATH = \uparrow SGMTGRPNUM \times scan\_rthd\_num \times 640 \uparrow \div 1{,}024$$

Explanation of variables

SGMTGRPNUM

See Table 6-13: Relationship between the variables max_sgmtnum_in_chunks and SGMTGRPNUM in (b) Determining the variable PROC_MAC under (1) Determining the process common memory requirement (for executing the adbarchivechunk command).

scan_rthd_num

Use the following formula to determine the value:

$$\downarrow (value\text{-}specified\text{-}for\text{-}archive\text{-}chunk\text{-}option\text{-}adb\_arcv\_rthd\_num - 1) \div 2 \downarrow$$

(3) Determining the heap memory requirement (for executing the adbarchivechunk command)

Use the following formula to determine the heap memory (*HEAP_ARCCKSZ*) required to execute the `adbarchivechunk` command.

Formula (kilobytes)

$$HEAP\_ARCCKSZ = \sum (HEAP\_ZLIB)^{\#}$$

#

If you execute multiple `adbarchivechunk` commands concurrently, determine the heap memory requirement for each `adbarchivechunk` command. Then, add up the total heap memory requirements.

Explanation of variables

HEAP_ZLIB

Use the following formula to determine this value.

Formula (kilobytes)

$$HEAP\_ZLIB = 262 \times archive\_file\_num$$

archive_file_num

Specify the number of archive files for the archived chunk.

If you execute the `adbarchivechunk` command for multiple chunks, check the number of archive files for each chunk, and then substitute for this variable the largest of the numbers.

Use the `adbdbstatus` command to output the summary information of archived chunks, and then check the value of `Archive_file_num` for each chunk. For details about `Archive_file_num`, see the following section in the manual *HADB Command Reference*: *List of items that are output in the summary information of archived chunks* in *Items that are output in the summary information of archived chunks* in *adbdbstatus (Analyze the Database Status)*.

6.3.17 Determining the memory requirement for executing the adbunarchivechunk command

When the `adbunarchivechunk` command is executed, the HADB server uses the following types of memory. Determine the requirement for each type of memory.

- **Shared memory**
  - Process common memory (*PROC_UNARCSZ*)
  - Real thread private memory (*RTHD_UNARCCKSZ*)
- **Process memory**
  - Heap memory (*HEAP_UNARCCKSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbunarchivechunk command)

Use the following formula to determine the process common memory (*PROC_UNARCSZ*) required for executing the `adbunarchivechunk` command.

Formula (kilobytes)

$$PROC\_UNARCSZ = \sum(PROC\_UNARCCON + (PROC\_UMAC + PROC\_IDX^{\#1} + PROC\_IDXBUILD^{\#1} + PROC\_RNGIDX^{\#2} + PROC\_INFILE^{\#3} + PROC\_DBUPDINF + PROC\_RTHDUPDINF)^{\#4})^{\#5}$$

#1

Add this value if you execute the `adbunarchivechunk` command for a table for which a B-tree index has been defined.

#2

Add this value if you execute the `adbunarchivechunk` command for a table for which a range index has been defined.

#3

Add this value if the number of archive files for the target chunk exceeds 1,024.

#4

If you execute the `adbunarchivechunk` command for multiple chunks, determine the memory requirement for each chunk. Then, add the largest of the determined values to this variable.

#5

If you execute multiple `adbunarchivechunk` commands concurrently, determine the memory requirement for each `adbunarchivechunk` command. Then, add up the total memory requirements.

Explanation of variables

PROC_UNARCCON

Determine the value as explained in (a) Determining the variable PROC_UNARCCON.

PROC_UMAC

Determine the value as explained in (b) Determining the variable PROC_UMAC.

PROC_IDX

Determine the value as explained in (c) Determining the variable PROC_IDX.

PROC_IDXBUILD

Determine the value as explained in (d) Determining the variable PROC_IDXBUILD.

PROC_RNGIDX

Determine the value as explained in (e) Determining the variable PROC_RNGIDX.

PROC_INFILE

Determine the value as explained in (f) Determining the variable PROC_INFILE.

PROC_DBUPDINF

Determine the value as explained in (g) Determining the variable PROC_DBUPDINF.

PROC_RTHDUPDINF

Determine the value as explained in (h) Determining the variable PROC_RTHDUPDINF.

(a) Determining the variable PROC_UNARCCON

Use the following formula to determine the value of variable *PROC_UNARCCON*.

Formula (kilobytes)

$$PROC\_UNARCCON = \uparrow(250{,}000 + 800) \div 1{,}024 \uparrow + PROC\_AUDINFSZ^{\#}$$

#

Add this value when the audit trail facility is enabled.

Explanation of variables

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(b) Determining the variable PROC_UMAC

Use the following formula to determine the value of variable *PROC_UMAC*.

Formula (kilobytes)

$$PROC\_UMAC = 3 + 135 + DIC + IOA + LOD + IMPORTBUF\_CTL + STS + PAGEALLOC$$

Explanation of variables

DIC

See the explanation of the variable *DIC* in (a) Determining the variable PROC_IMPT in (1) Determining the process common memory requirement (for executing the adbimport command) in 6.3.6 Determining the memory requirement for executing the adbimport command.

IOA

Use the following formula to determine this value.

Formula (kilobytes)

```
IOA＝↑（max_rowsz＋436＋448×col_num＋124×idx_num＋4×dba_num）÷1,024↑×load_rthd
```

max_rowsz

Maximum row length (bytes)

Determine the maximum row length based on the formula for the row length *ROWSZ* in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

col_num

Number of columns in the table to be processed

idx_num

Number of indexes defined for the table to be processed

dba_num

Number of DB area files for the DB areas that store the table to be processed

load_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

LOD

Use the following formula to determine this value.

Formula (kilobytes)

```
LOD＝
 ↑（（（read_size×read_buff_num＋decomp_buff_size＋1）×1,024×load_rthd）
 ＋（（5,200＋1,024,000#＋input_edit_area）×load_rthd）
 ＋（511×input_file_num））÷1,024↑
```

#

If you have added columns after the chunk has been archived, add the information about the added columns.

read_size

Value specified for the unarchive chunk option `adb_unarcv_read_size`

read_buff_num

Substitute 3.

decomp_buff_size

Value specified for the unarchive chunk option `adb_unarcv_decompress_buff_size`

load_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

input_file_num

Substitute 1,024.

input_edit_area

Use the following formula to determine this value.

Formula

```
input_edit_area＝
    MIN（MAX(↑input_recsize÷1,024↑×1,024 , 32,768) , 536,870,912）
```

input_recsize

Use the following formula to determine this value.

Formula

input_recsize = Σ(*DATASIZE*) + *number-of-columns-in-table-definition* + 1

Σ(*DATASIZE*)

Sum of the largest data lengths (in character format) for the data types of all columns in the processing-target table

Determine the largest data length of each data type in character format from the following table.

Table 6-14: Largest data length in character format for each data type

No.	Classification	Data type	Largest data length in character format
1	Numeric data	INTEGER	23
2		SMALLINT	13
3		DECIMAL(*m*,*n*)	*m* + 5
4		DOUBLE PRECISION	511
5	Character string data	CHAR(*n*)	*n* × 2 + 2
6		VARCHAR(*n*)	*n* × 2 + 2
7	Datetime data	DATE	12
8		TIME(*p*)	8 + *p* + 3
9		TIMESTAMP(*p*)	19 + *p* + 3
10	Binary data	BINARY(*n*)	*n* × 8 + 2
11		VARBINARY(*n*)	*n* × 8 + 2

Legend:

m, *n*, *p*: See the topic *List of data types* in the manual *HADB SQL Reference*.

IMPORTBUF_CTL

Use the following formula to determine this value.

Formula (kilobytes)

IMPORTBUF_CTL = ↑(248 + *BUFBLK*) ÷ 1,024↑ × *load_rthd*

BUFBLK

For details, see the description of the variable *BUFBLK* in (f) Determining the variable PROC_UPDSZ under (1) Determining the process common memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

load_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

STS

Use the following formula to determine this value.

Formula (kilobytes)

```
STS=
 ↑（256＋（1,185×idx_num）＋（2,067×（b_tree_idx_num＋range_idx_num）×load_rthd
  ＋2,067×text_idx_num×16）＋1,025）÷1,024↑
```

idx_num

Number of indexes defined for the table to be processed

b_tree_idx_num

Number of B-tree indexes defined for the table to be processed

range_idx_num

Number of range indexes defined for the table to be processed

load_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

text_idx_num

Number of text indexes defined for the table to be processed

PAGEALLOC

Use the following formula to determine this value.

Formula (kilobytes)

```
PAGEALLOC = unarcv_rthd × 310
```

unarcv_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

(c) Determining the variable PROC_IDX

Use the following formula to determine the value of variable *PROC_IDX*.

Formula (kilobytes)

```
PROC_IDX=
 ↑（idxf_buff_size×1,024＋（13,151＋（33,928＋（13×KEYSZ））
  ×idx_num×load_rthd）×cmd_d_opt＋（152＋rd_buff_size×load_rthd
  ×2×1,024＋2×ld_buff_size×1,024＋（KEYSZ＋CTRL）×text_idx_num）
  ×dividx_rthd）÷1,024↑
```

Explanation of variables

idxf_buff_size

Substitute 1,024.

KEYSZ

Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length. If multiple B-tree indexes have been defined for the processing-target table, use the largest key length among all the B-tree indexes.

idx_num

Number of B-tree indexes defined for the table to be processed

load_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

cmd_d_opt

Substitute 1.

rd_buff_size

Value specified for the unarchive chunk option `adb_unarcv_dividx_rd_buff_size`

ld_buff_size

Value specified for the unarchive chunk option `adb_unarcv_dividx_wt_buff_size`

CTRL

- If the indexed columns are fixed-length columns
  Substitute 10 bytes.
- If the indexed columns include variable-length columns
  Substitute 12 bytes.

text_idx_num

Number of text indexes defined for the table to be processed

dividx_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

(d) Determining the variable PROC_IDXBUILD

Use the following formula to determine the variable *PROC_IDXBUILD*.

Formula (kilobytes)

$$PROC\_IDXBUILD = PROC\_IDXCREATE$$

Explanation of variables

PROC_IDXCREATE: Use the following formula to determine its value.

Formula (kilobytes)

$$PROC\_IDXCREATE = \max_{1 \leqq k \leqq idx\_num} PROC\_IDXCREATE\_MEM(k)$$

idx_num

Number of B-tree indexes defined for the table to be processed

PROC_IDXCREATE_MEM(k)

Use the following formula to determine this value.

Formula (kilobytes)

```
PROC_IDXCREATE_MEM(k)=
 ↑（456＋256×idx_col_num＋（24＋144×↑idx_lv÷16↑）
  ×（dividx_rthd＋1）＋（KEYSZ＋（idx_col_num＋1）×2＋12）
  ×2＋page_size×2＋268）÷1,024↑
```

idx_col_num

Number of B-tree indexed columns

idx_lv

Number of levels in the B-tree index

Calculate the recursion formula *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment and substitute the value of *n* when *PIDX(n)* results in 1.

Note that in the case of *PIDX(1)* = 1, the number of levels in the B-tree index is 2.

dividx_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

KEYSZ

Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

page_size

Page size of the data DB area in which the B-tree index is defined (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(e) Determining the variable PROC_RNGIDX

Use the following formula to determine the variable *PROC_RNGIDX*.

Formula (kilobytes)

```
PROC_RNGIDX= ↑64×rngidx_num÷1,024↑
```

Explanation of variables

rngidx_num

Number of range indexes defined for the table to be processed

(f) Determining the variable PROC_INFILE

If the number of archive files for one chunk exceeds 1,024, you must add the variable *PROC_INFILE*. Use the following formula to determine this value.

Formula (kilobytes)

```
PROC_INFILE= ↑((16＋511)×(INFMAX－1,024))÷1,024↑
```

Explanation of variables

INFMAX

Number of archive files for the chunk

(g) Determining the variable PROC_DBUPDINF

Use the following formula to determine the value of variable *PROC_DBUPDINF*. Note that the formula to be used differs depending on whether the multi-node function is used.

Formula to be used when the multi-node function is not used (kilobytes)

```
PROC_DBUPDINF=
 ↑（（40＋↑（DBUPDINF_ENT_NUM÷1,024）↑×122,888）÷1,024）↑
```

Formula to be used when the multi-node function is used (kilobytes)

```
PROC_DBUPDINF=
 129＋↑（（40＋↑（DBUPDINF_ENT_NUM÷1,024）↑×122,888）÷1,024）↑
```

Explanation of variables

DBUPDINF_ENT_NUM

Use the following formula to determine this value.

Formula (count)

```
DBUPDINF_ENT_NUM= archive_file_num
```

archive_file_num

Specify the number of archive files for the archived chunk.

Use the `adbdbstatus` command to output the summary information of archived chunks, and then check the value of `Archive_file_num` for each chunk. For details about `Archive_file_num`, see the following section in the manual *HADB Command Reference*: *List of items that are output in the summary information of archived chunks* in *Items that are output in the summary information of archived chunks* in *adbdbstatus (Analyze the Database Status)*.

(h) Determining the variable PROC_RTHDUPDINF

Use the following formula to determine the variable *PROC_RTHDUPDINF*.

Formula (kilobytes)

```
PROC_RTHDUPDINF=↑（648＋（168×rthd_num））÷1,024↑
```

Explanation of variables

rthd_num

Use the following formula to determine this value.

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

(2) Determining the real thread private memory requirement (for executing the adbunarchivechunk command)

Use the following formula to determine the amount of real thread private memory (*RTHD_UNARCCKSZ*) required to execute the `adbunarchivechunk` command.

Formula (kilobytes)

```
RTHD_UNARCCKSZ=
    IMPORTBUF+SORTIOBUF #1+SORTBUF #1
    +RTHD_DATALOAD+RTHD_IDXREC #1+RTHD_IDXBUILD #1
    +RTHD_RNGIDX #2+RTHD_TXTIDX #3+RTHD_ARCDIRPATH
```

#1

Add this value if you execute the `adbunarchivechunk` command for a table for which a B-tree index has been defined.

#2

Add this value if you execute the `adbunarchivechunk` command for a table for which a range index has been defined.

#3

Add this value if you execute the `adbunarchivechunk` command for a table for which a text index has been defined.

Explanation of variables

IMPORTBUF

Determine the value as explained in (a) Determining the variable IMPORTBUF.

SORTIOBUF

Determine the value as explained in (b) Determining the variable SORTIOBUF.

SORTBUF

Determine the value as explained in (c) Determining the variable SORTBUF.

RTHD_DATALOAD

Determine the value as explained in (d) Determining the variable RTHD_DATALOAD.

RTHD_IDXREC

Determine the value as explained in (e) Determining the variable RTHD_IDXREC.

RTHD_IDXBUILD

Determine the value as explained in (f) Determining the variable RTHD_IDXBUILD.

RTHD_RNGIDX

Determine the value as explained in (g) Determining the variable RTHD_RNGIDX.

RTHD_TXTIDX

Determine the value as explained in (h) Determining the variable RTHD_TXTIDX.

RTHD_ARCDIRPATH

Determine the value as explained in (i) Determining the variable RTHD_ARCDIRPATH.

(a) Determining the variable IMPORTBUF

Use the following formula to determine the variable *IMPORTBUF*.

Formula (kilobytes)

```
IMPORTBUF=
  ↑(IMP_SQBLK+IMP_SQIO+IMP_SQPGE+IMP_SQHS)÷1,024↑
```

Explanation of variables

IMP_SQBLK

Use the following formula to determine this value.

Formula (bytes)

```
IMP_SQBLK=
  imp_blknum×(112+↑IMP_BLKSZ÷imp_pagesize÷64↑×24)
```

imp_blknum

Value specified for the unarchive chunk option `adb_unarcv_buff_blk_num`

IMP_BLKSZ

Use the following formula to determine this value.

Formula

```
IMP_BLKSZ= 4,096×1,024
```

imp_pagesize

The smallest of the page sizes of the following DB areas (bytes):

- DB area storing the processing-target table
- DB area storing the B-tree index or text index that is defined for the processing-target table

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

IMP_SQIO

Use the following formula to determine this value.

Formula (bytes)

```
IMP_SQIO = 568×imp_blknum
```

imp_blknum

Value specified for the unarchive chunk option `adb_unarcv_buff_blk_num`

IMP_SQPGE

Use the following formula to determine this value.

Formula (bytes)

```
IMP_SQPGE=
  (176+imp_pagesize+BUFLOG)×(IMP_BLKSZ÷imp_pagesize)×imp_blknum
```

imp_pagesize

The smallest of the page sizes of the following DB areas (bytes):

- DB area storing the processing-target table
- DB area storing the B-tree index or text index that is defined for the processing-target table

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

BUFLOG

For details, see the description of the variable *BUFLOG* in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

IMP_BLKSZ

Use the following formula to determine this value.

Formula

IMP_BLKSZ= 4,096 × 1,024

imp_blknum

Value specified for the unarchive chunk option `adb_unarcv_buff_blk_num`

IMP_SQHS

Use the following formula to determine this value.

Formula (bytes)

IMP_SQHS= (8 × *imp_blknum*)+(40 × *imp_blknum*)

imp_blknum

Value specified for the unarchive chunk option `adb_unarcv_buff_blk_num`

(b) Determining the variable SORTIOBUF

Use the following formula to determine the variable *SORTIOBUF*.

Formula (kilobytes)

SORTIOBUF= *sort_io_buff_size* × *sort_rthd*

Explanation of variables

sort_io_buff_size

Substitute 16.

sort_rthd

Use the following formula to determine the value:

value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1

(c) Determining the variable SORTBUF

Use the following formula to determine the variable *SORTBUF*.

Formula (kilobytes)

SORTBUF= （48+*sort_buff_size* × 1,024） × *sort_rthd*

Explanation of variables

sort_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

sort_buff_size

Value specified for the unarchive chunk option `adb_unarcv_sort_buff_size`#

#

A work file is created during sort processing. The formula for the variable *sort_buff_size*, which minimizes the size of this file, is shown later. Make sure that the variable *sort_buff_size* meets the following formula. However, if you are running out of memory or if most data items are already arranged in index key order, do not specify more space than is necessary.

Formula

$$sort\_buff\_size \geqq \left\lceil \left(\frac{REC\_SIZE+7}{2} + \sqrt{(BLK+8) \times row\_num \times KEY\_INF + \frac{REC\_SIZE+7}{4}} + COL\_INFO \right) \div 1{,}024 \right\rceil$$

REC_SIZE

Use the following formula to determine this value.

Formula

$$REC\_SIZE = (KEYSZ + SYS1) + SYS2$$

KEYSZ

Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length. If multiple B-tree indexes have been defined for the table to be processed, use the largest key length among all the B-tree indexes.

SYS1

- If the indexed columns of the indexes defined for the table to be processed are fixed-length:
  Substitute 10.
- If any of the indexed columns of the indexes defined for the table to be processed are variable-length:
  Substitute 12.

SYS2

- If any of the indexes defined for the table to be processed are variable-length multiple-column indexes:
  Use the following formula to determine this value.
  Formula

  $$SYS2 = (number\ of\ indexed\ columns \times 4)$$

- For all other indexes:
  Substitute 0.

BLK

Use the following formula to determine this value.

Formula

$$BLK = REC\_SIZE + (KEYSZ + 8) + 56$$

row_num

Number of rows to be stored in the table by the `adbunarchivechunk` command

KEY_INF

Use the following formula to determine this value.

Formula

$$KEY\_INF = REC\_SIZE + (KEYSZ + 8) + 28$$

COL_INFO

Use the following formula to determine this value.

Formula

$$COL\_INFO = 2{,}112 + (MULTI\_KEY\_INFO \times 32) + (KEYSZ + 8)$$

MULTI_KEY_INFO

- If any of the B-tree indexes defined for the table to be processed are variable-length multiple-column indexes:
  Use the following formula to determine this value.
  Formula

  $$MULTI\_KEY\_INFO = (number\ of\ indexed\ columns \times 2) + 2$$

- For all other indexes:
  Substitute 5.

(d) Determining the variable RTHD_DATALOAD

Use the following formula to determine the variable *RTHD_DATALOAD*.

Formula (kilobytes)

$$RTHD\_DATALOAD = \uparrow\downarrow (80 + 192^{\#1} + 8 \times (col\_num + var\_col\_num) + 32 \times var\_col\_num^{\#2} + 7) \div 8\downarrow \times 8 \div 1{,}024\uparrow$$

#1

Add this value if a range index is defined for the table to be processed.

#2

Add this value when the table to be processed is not a FIX table.

Explanation of variables

col_num

Number of columns in the table to be processed

var_col_num

Number of `VARCHAR`-type and `VARBINARY`-type columns in the table to be processed

(e) Determining the variable RTHD_IDXREC

Use the following formula to determine the variable *RTHD_IDXREC*.

Formula (kilobytes)

$$RTHD\_IDXREC=$$
$$\uparrow(386+328\times(idx\_num-1)+32\times max\_idx\_col\_num$$
$$+(64+buf\_size\times1{,}024)\times2\times idx\_num)\div1{,}024\uparrow$$

Explanation of variables

idx_num

Number of B-tree indexes defined for the table to be processed

max_idx_col_num

Maximum number of indexed columns in the B-tree index defined for the table to be processed

buf_size

Substitute 1,024.

(f) Determining the variable RTHD_IDXBUILD

Use the following formula to determine the variable *RTHD_IDXBUILD*.

Formula (kilobytes)

$$RTHD\_IDXBUILD=\max_{1\leqq k\leqq idx\_num} RTHD\_IDXBUILD\_MEM(k)$$

Explanation of variables

idx_num

Number of B-tree indexes defined for the table to be processed

RTHD_IDXBUILD_MEM(k)

Use the following formula to determine this value.

Formula (kilobytes)

$$RTHD\_IDXBUILD\_MEM(k)=$$
$$\uparrow(61{,}474+\uparrow MAX(255,\ KEYSZ)\div2\uparrow\times2+3{,}070$$
$$+\uparrow KEYSZ+CTRL\div8\uparrow\times8+MAX(255,\ KEYSZ)$$
$$+14+\downarrow8+page\_size\times95\div100\downarrow\times2)\div1{,}024\uparrow$$

page_size

Page size of the data DB area in which the B-tree index is defined (bytes)

Determine this value based on the explanation of the page size of the data DB area in Table 6-3: DB area page size under (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

KEYSZ

Key length of a B-tree index (bytes)

Determine the key length of the index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL

- If all key lengths in the indexed columns are fixed:

  Substitute 10 bytes.

- If some of the key lengths in the indexed columns are variable:
  Substitute 12 bytes.

(g) Determining the variable RTHD_RNGIDX

Use the following formula to determine the variable *RTHD_RNGIDX*.

Formula (kilobytes)

$$RTHD\_RNGIDX = \uparrow (16 + 128 \times rngidx\_num) \times load\_rthd \div 1{,}024 \uparrow$$

Explanation of variables

rngidx_num
Number of range indexes defined for the table to be processed

load_rthd
Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

(h) Determining the variable RTHD_TXTIDX

Use the following formula to determine the variable *RTHD_TXTIDX*.

Formula (kilobytes)

$$RTHD\_TXTIDX = \max_{1 \leqq k \leqq idx\_num} RTHD\_TXTIDXBUILD\_MEM(k)$$

Explanation of variables

idx_num
Number of text indexes defined for the table to be processed

RTHD_TXTIDXBUILD_MEM(k)
Use the following formula to determine this value.

Formula (kilobytes)

$$RTHD\_TXTIDXBUILD\_MEM(k) = buff\_blk\_size \times buff\_blk\_num + 3{,}072 + \uparrow (idx\_div\_num \times 24) \div 1{,}024 \uparrow + txt\_sort\_buff\_size \times 1{,}024$$

buff_blk_size
Substitute 4,096.

buff_blk_num
Value specified for the unarchive chunk option `adb_unarcv_buff_blk_num`

idx_div_num
Length defined for the indexed column for the k-th text index

txt_sort_buff_size
Value specified for the unarchive chunk option `adb_unarcv_txt_buff_size`

(i) Determining the variable RTHD_ARCDIRPATH

Substitute the following value for the variable *RTHD_ARCDIRPATH*.

Value (kilobytes)

RTHD_ARCDIRPATH= 1

(3) Determining the heap memory requirement (for executing the adbunarchivechunk command)

Use the following formula to determine the heap memory (*HEAP_UNARCCKSZ*) required to execute the `adbunarchivechunk` command.

Formula (kilobytes)

$$HEAP\_UNARCCKSZ = \sum (HEAP\_ZLIB + HEAP\_SRTHMEM)^{\#}$$

#

If you execute multiple `adbunarchivechunk` commands concurrently, determine the heap memory requirement for each `adbunarchivechunk` command. Then, add up the total heap memory requirements.

Explanation of variables

HEAP_ZLIB

Determine the value as explained in (a) Determining the variable HEAP_ZLIB.

HEAP_SRTHMEM

Determine the value as explained in (b) Determining the variable HEAP_SRTHMEM.

(a) Determining the variable HEAP_ZLIB

Use the following formula to determine the value of variable *HEAP_ZLIB*.

Formula (kilobytes)

$$HEAP\_ZLIB = 42 \times load\_rthd$$

Explanation of variables

load_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

(b) Determining the variable HEAP_SRTHMEM

Use the following formula to determine the value of variable *HEAP_SRTHMEM*.

Formula (kilobytes)

```
HEAP_SRTHMEM= 505 × sort_rthd
```

Explanation of variables

sort_rthd

Use the following formula to determine the value:

```
value-specified-for-unarchive-chunk-option-adb_unarcv_rthd_num - 1
```

6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command

When the `adbreorgsystemdata` command is executed, the HADB server uses the following types of memory. Determine the requirement for each type of memory.

Note that the table to be processed here means the system table (base table).

▪ **Shared memory**

- Process common memory (*PROC_REORGSZ*)
- Real thread private memory (*RTHD_REORGSZ*)

▪ **Process memory**

- Heap memory (*HEAP_REORGSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbreorgsystemdata command)

Use the following formula to determine the process common memory (*PROC_REORGSZ*) required for executing the `adbreorgsystemdata` command.

Formula (kilobytes)

```
PROC_REORGSZ=
   PROC_MNG+PROC_IDX+PROC_IDXBUILD
   +PROC_DBUPDINF+PROC_RTHDUPDINF+PROC_UNLOAD_BUFF
```

Explanation of variables

PROC_MNG

Process common memory used for managing reorganization of system tables

Determine the value as explained in (a) Determining the variable PROC_MNG.

PROC_IDX

Process common memory used for creating B-tree indexes

Determine the value as explained in (b) Determining the variable PROC_IDX.

PROC_IDXBUILD

Process common memory used for creating B-tree indexes

Determine the value as explained in (c) Determining the variable PROC_IDXBUILD.

PROC_DBUPDINF

Process common memory required for storing DB area, table, index, and chunk update information

Determine the value as explained in (d) Determining the variable PROC_DBUPDINF.

PROC_RTHDUPDINF

Processing real thread update information

Determine the value as explained in (e) Determining the variable PROC_RTHDUPDINF.

PROC_UNLOAD_BUFF

Buffer used for reorganizing system tables

Determine the value as explained in (f) Determining the variable PROC_UNLOAD_BUFF.

(a) Determining the variable PROC_MNG

Use the following formula to determine the value of variable *PROC_MNG*.

Formula (kilobytes)

$$PROC\_MNG = 3 + 135 + DIC + IOA + LOD + IMPORTBUF\_CTL + STS + PAGEALLOC + EXPF + PROC\_AUDINFSZ^{\#}$$

#

Add this value when the audit trail facility is enabled.

Explanation of variables

DIC

Memory for acquiring the definition information of the table to be processed

Use the following formula to determine this value.

Formula (kilobytes)

$$DIC = \uparrow(1{,}012 + 512 \times col\_num + max\_rowsz + 1{,}156 \times idx\_num) \div 1{,}024\uparrow$$

col_num

Number of columns in the table to be processed

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

max_rowsz

Maximum row length

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

idx_num

Number of indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

Table 6-15: Variable value corresponding to system table

No.	Name of table to be processed	Value of col_num variable	Value of var_col_num variable	Value of max_rowsz variable	Value of Σ(DATASIZE) variable	Value of idx_num variable	Value of max_idx_col_num variable	Value of max_keysz variable
1	STATUS_TABLES	12	2	318	1,556	1	2	203
2	STATUS_COLUMNS	11	6	32,588	257,088	1	3	304
3	STATUS_INDEXES	8	3	476	701	2	2	203
4	STATUS_CHUNKS	10	3	1,312	2,575	1	3	212
5	STATUS_SYNONYM_DICTIONARIES	10	6	2,108	4,152	1	1	121

IOA

Memory for creating the data image to be stored in the database

Use the following formula to determine this value.

Formula (kilobytes)

IOA＝MAX（*IOA_UNLOAD*，*IOA_RELOAD*）

IOA_UNLOAD

Memory to be used for unloading

Use the following formula to determine this value.

Formula

IOA_UNLOAD＝
sql_size＋（*scan_buff_size*＋*offset_area*×2）×*scan_rthd*×2

sql_size

Use the following formula to determine this value.

sql_size＝1＋1,024

scan_buff_size

Use the following formula to determine this value.

scan_buff_size＝16×1,024

offset_area

Use the following formula to determine this value.

offset_area＝
（8×*col_num*×（*scan_buff_size*×1,024÷Σ*(DATASIZE)*））÷1,024

col_num

Number of columns in the table to be processed

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

Σ(*DATASIZE*)

Sum of the largest data lengths (in character format) for the data types of all columns in the processing-target table

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

scan_rthd

Substitute 1.

IOA_RELOAD

Memory to be used for reloading

Use the following formula to determine this value.

Formula

```
IOA_RELOAD=
↑（max_rowsz+436+448×col_num+124×idx_num+4×dba_num）
  ÷1,024↑×load_rthd
```

max_rowsz

Maximum row length

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

col_num

Number of columns in the table to be processed

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

idx_num

Number of indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

dba_num

Substitute 1.

load_rthd

Substitute 1.

LOD

Memory required for reloading

Use the following formula to determine this value.

Formula (kilobytes)

```
LOD=
  ↑（（（read_size×read_buff_num）×1,024×load_rthd）
  +（（5,200+input_edit_area）×load_rthd）
  +（511×input_file_num））÷1,024↑
```

read_size

Substitute 1,024.

read_buff_num

Substitute 3.

load_rthd

Substitute 1.

input_edit_area

Length of the input data editing area

Use the following formula to determine this value.

Formula

input_edit_area=
MIN（MAX（↑*input_recsize*÷1,024↑×1,024 , 32,768）, 536,870,912）

input_recsize

input record length

Use the following formula to determine this value.

Formula

input_recsize= Σ *(DATASIZE)* ＋ *col_num*×8×2

Σ(*DATASIZE*)

Sum of the largest data lengths (in character format) for the data types of all columns in the processing-target table

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

col_num

Number of columns in the table to be processed

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

input_file_num

Substitute 1,024.

IMPORTBUF_CTL

Buffer control information for reorganizing system tables

Use the following formula to determine this value.

Formula (kilobytes)

IMPORTBUF_CTL=↑（248＋*BUFBLK*）÷1,024↑×*load_rthd*

BUFBLK

For details, see the description of the variable *BUFBLK* in (f) Determining the variable PROC_UPDSZ under (1) Determining the process common memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

load_rthd

Substitute 1.

STS

Memory used for saving status information

Use the following formula to determine this value.

Formula (kilobytes)

```
STS=
 ↑（256＋（1,185×idx_num）＋（2,067×idx_num×load_rthd）＋1,025）÷1,024↑
```

idx_num

Number of indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table and substitute the value for the table to be processed.

load_rthd

Substitute 1.

PAGEALLOC

Page allocation control information for reloading

Use the following formula to determine this value.

Formula (kilobytes)

```
PAGEALLOC= MAX（dataload_rthd , dividx_rthd）×310
```

dataload_rthd

Substitute 1.

dividx_rthd

Substitute 1.

EXPF

Memory to be used for processing files

Use the following formula to determine this value.

Formula (kilobytes)

```
EXPF=552＋512×exp_rthd
```

exp_rthd

Substitute 1.

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(b) Determining the variable PROC_IDX

Use the following formula to determine the value of variable *PROC_IDX*.

Formula (kilobytes)

```
PROC_IDX=
 ↑（idxf_buff_size×1,024＋（13,151＋（33,928＋（13×max_keysz））
  ×idx_num×load_rthd）×cmd_d_opt
  ＋（152＋rd_buff_size×load_rthd×2×1,024＋2×ld_buff_size×1,024）
  ×dividx_rthd）÷1,024↑
```

Explanation of variables

idxf_buff_size

Substitute 1,024.

max_keysz

Maximum index key length among indexes defined for the table to be processed (bytes)

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

idx_num

Number of indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

load_rthd

Substitute 1.

cmd_d_opt

Substitute 1.

rd_buff_size

Substitute 1,024.

ld_buff_size

Substitute 1,024.

dividx_rthd

Substitute 1.

(c) Determining the variable PROC_IDXBUILD

Use the following formula to determine the variable *PROC_IDXBUILD*.

Formula (kilobytes)

$$PROC\_IDXBUILD = \max_{k=1,\ldots idx\_num} PROC\_IDXCREATE\_MEM(k)$$

Explanation of variables

idx_num

Number of indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

PROC_IDXCREATE_MEM(k)

Memory required to batch-create *k*-th index defined for the table to be processed

Use the following formula to determine this value.

Formula (kilobytes)

```
PROC_IDXCREATE_MEM(k)=
 ↑（456＋256×idx_col_num＋（24＋144×↑idx_lv÷16↑）
  ×（dividx_rthd＋1）＋（KEYSZ＋（idx_col_num＋1）×2＋12）
  ×2＋page_size×2＋268）÷1,024↑
```

idx_col_num

Number of indexed columns in index

Refer to Table 6-16: Values of variables idx_col_num and KEYSZ for indexes of system tables and substitute the value that corresponds to the index name.

idx_lv

Number of levels in the index

Calculate the recursion formula *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment and substitute the value of *n* when *PIDX(n)* results in 1.

Note that in the case of *PIDX(1)* = 1, the number of levels in the B-tree index is 2.

The following shows the values to be substituted for the variables used in the preceding formula:

- *page_size*
  Substitute 4,096.
- *pctfree*
  Substitute 0.
- *dbarea_file_num*
  Substitute 1.

dividx_rthd

Substitute 1.

KEYSZ

Key length of index (bytes)

Refer to Table 6-16: Values of variables idx_col_num and KEYSZ for indexes of system tables and substitute the value that corresponds to the index name.

page_size

Substitute 4,096.

Table 6-16: Values of variables idx_col_num and KEYSZ for indexes of system tables

No.	Name of table to be processed	Index name	Value of idx_col_num variable	Value of KEYSZ variable
1	STATUS_TABLES	STATUSINDEXM01	2	203
2	STATUS_COLUMNS	STATUSINDEXM02	3	304
3	STATUS_INDEXES	STATUSINDEXM03	2	203
4		STATUSINDEXM04	2	203
5	STATUS_CHUNKS	STATUSINDEXM05	3	212
6	STATUS_SYNONYM_DICTIONARIES	STATUSINDEXS01	1	121

(d) Determining the variable PROC_DBUPDINF

Use the following formula to determine the value of variable *PROC_DBUPDINF*.

Formula to be used when the multi-node function is not used (kilobytes)

```
PROC_DBUPDINF=
      ↑（（40＋↑（DBUPDINF_ENT_NUM÷1,024）↑×131,136）÷1,024）↑
```

Formula to be used when the multi-node function is used (kilobytes)

```
PROC_DBUPDINF=
      ↑（（80＋（↑（DBUPDINF_ENT_NUM÷1,024）↑＋1）×131,136）÷1,024）↑
```

Explanation of variables

DBUPDINF_ENT_NUM

Number of entries in the DB area, table, index, and chunk update information

This value changes depending on whether the multi-node function is used.

When the multi-node function is not used

Substitute 1.

When the multi-node function is used

Use the following formula to determine this value.

Formula (count)

```
DBUPDINF_ENT_NUM=5＋idx_num×10
```

idx_num

Number of indexes defined for the table to be processed

(e) Determining the variable PROC_RTHDUPDINF

Use the following formula to determine the variable *PROC_RTHDUPDINF*.

Formula (kilobytes)

```
PROC_RTHDUPDINF=↑（664＋（192×rthd_num））÷1,024↑
```

Explanation of variables

rthd_num

Substitute 1.

(f) Determining the variable PROC_UNLOAD_BUFF

Substitute the following value for the variable *PROC_UNLOAD_BUFF*.

Value (kilobytes)

```
PROC_UNLOAD_BUFF= 8,192
```

(2) Determining the real thread private memory requirement (for executing the adbreorgsystemdata command)

Use the following formula to determine the amount of real thread private memory (*RTHD_REORGSZ*) required to execute the `adbreorgsystemdata` command.

Formula (kilobytes)

RTHD_REORGSZ=MAX（*RTHD_UNLOADSZ* , *RTHD_RELOADSZ*）

Explanation of variables

RTHD_UNLOADSZ

Real thread private memory that the `adbreorgsystemdata` command uses during unloading

Determine the value as explained in (a) Determining the variable RTHD_UNLOADSZ.

RTHD_RELOADSZ

Real thread private memory that the `adbreorgsystemdata` command uses during reloading

Determine the value as explained in (b) Determining the variable RTHD_RELOADSZ.

(a) Determining the variable RTHD_UNLOADSZ

Use the following formula to determine the value of variable *RTHD_UNLOADSZ*.

Formula (kilobytes)

The formula to be used differs depending on the table to be processed.

- For the `STATUS_TABLES` table
  RTHD_UNLOADSZ = 809 + 5 × *uthd_num* + 6 × MAX(128 , *uthd_num*) + *SGTBL*
- For the `STATUS_COLUMNS` table
  RTHD_UNLOADSZ = 33,452 + 5 × *uthd_num* + 511 × MAX(128 , *uthd_num*) + 33 × *SGTBL*
- For the `STATUS_INDEXES` table
  RTHD_UNLOADSZ = 877 + 5 × *uthd_num* + 7 × MAX(128 , *uthd_num*) + *SGTBL*
- For the `STATUS_CHUNKS` table
  RTHD_UNLOADSZ = 1,808 + 5 × *uthd_num* + 22 × MAX(128 , *uthd_num*) + 2 × *SGTBL*
- For the `STATUS_SYNONYM_DICTIONARIES` table
  RTHD_UNLOADSZ = 2,612 + 5 × *uthd_num* + 35 × MAX(128 , *uthd_num*) + 3 × *SGTBL*

Explanation of variables

uthd_num

Value specified for the `adb_sys_uthd_num` operand in the server definition

SGTBL

Number of segments in table to be processed

Refer to the following table and substitute the value for the table to be processed.

Table 6-17: Estimating the segment size of system tables

No.	Name of table to be processed	Determining the SGTBL variable
1	STATUS_TABLES	↑(*STBLTABLESSIZE* ÷ 64)↑
2	STATUS_COLUMNS	↑(*STBLCOLUMNSSIZE* ÷ 64)↑
3	STATUS_INDEXES	↑(*STBLINDEXESSIZE* ÷ 64)↑
4	STATUS_CHUNKS	↑(*STBLCHUNKSSIZE* ÷ 64)↑
5	STATUS_SYNONYM_DICTIONARIES	↑(*STBLSYNONYMDICSIZE* ÷ 64)↑

For details about the variables *STBLTABLESSIZE*, *STBLCOLUMNSSIZE*, *STBLINDEXESSIZE*, *STBLCHUNKSSIZE*, and *STBLSYNONYMDICSIZE*, see 5.12 Estimating the size of the system-table DB area.

(b) Determining the variable RTHD_RELOADSZ

Use the following formula to determine the value of variable *RTHD_RELOADSZ*.

Formula (kilobytes)

```
RTHD_RELOADSZ=
    RELOADBUF+SORTIOBUF+SORTBUF
    +RTHD_DATALOAD+RTHD_IDXREC+RTHD_IDXBUILD
```

Explanation of variables

RELOADBUF

Use the following formula to determine this value.

Formula (kilobytes)

```
RELOADBUF = ↑(IMP_SQBLK + IMP_SQIO + IMP_SQPGE + IMP_SQHS) /1,024↑
```

IMP_SQBLK

Use the following formula to determine this value.

Formula (bytes)

```
IMP_SQBLK = imp_blknum × (112 + ↑IMP_BLKSZ ÷ imp_pagesize ÷ 64↑ × 24)
```

imp_blknum

Substitute 64.

IMP_BLKSZ

Use the following formula to determine this value.

```
IMP_BLKSZ = 4,096 x 1,024
```

imp_pagesize

Substitute 4,096.

IMP_SQIO

Use the following formula to determine this value.

Formula (bytes)

```
IMP_SQIO = 568 × imp_blknum
```

IMP_SQPGE

Use the following formula to determine this value.

Formula (bytes)

```
IMP_SQPGE = (176 + imp_pagesize + BUFLOG) x (IMP_BLKSZ / imp_pagesize) x imp_blknum
```

BUFLOG

For details, see the description of the variable *BUFLOG* in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

IMP_SQHS

Use the following formula to determine this value.

Formula (bytes)

```
IMP_SQHS = (8 x imp_blknum) + (40 x imp_blknum)
```

SORTIOBUF

Use the following formula to determine this value.

Formula (kilobytes)

```
SORTIOBUF = sort_io_buff_size x sort_rthd
```

sort_io_buff_size

Substitute 16.

sort_rthd

Substitute 1.

SORTBUF

Use the following formula to determine this value.

Formula (kilobytes)

```
SORTBUF = (48 + sort_buff_size x 1,024) x sort_rthd
```

sort_buff_size

Substitute 256.

RTHD_DATALOAD

Use the following formula to determine this value.

Formula (kilobytes)

```
RTHD_DATALOAD=
↑(↓(80+8×(col_num+var_col_num)+32×var_col_num+7)÷8↓×8)÷1,024↑
```

col_num

Number of columns in the table to be processed

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

var_col_num

Number of `VARCHAR`-type and `VARBINARY`-type columns in the table to be processed

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

RTHD_IDXREC

Use the following formula to determine this value.

Formula (kilobytes)

```
RTHD_IDXREC =
↑(386 + 328 x (idx_num-1) + 32 x max_idx_col_num + (64 + buf_size x 1,024) x 2 x idx_num) /1,024↑
```

idx_num

Number of indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

max_idx_col_num

Maximum number of indexed columns among indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

buf_size

Substitute 1,024.

RTHD_IDXBUILD

Use the following formula to determine this value.

Formula (kilobytes)

$$RTHD\_IDXBUILD = \max_{1 \leqq k \leqq idx\_num} RTHD\_IDXBUILD\_MEM(k)$$

idx_num

Number of indexes defined for the table to be processed

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

RTHD_IDXBUILD_MEM(k)

Memory required for creating at one time the *k*-th B-tree index defined for the table to be processed

Use the following formula to determine this value.

Formula (kilobytes)

```
RTHD_IDXBUILD_MEM（k）＝
  ↑（61,474＋↑MAX（255，KEYSZ）÷2↑×2＋3,070
  ＋↑KEYSZ＋CTRL÷8↑×8＋MAX（255，KEYSZ）
  ＋14＋↓8＋page_size×95÷100↓×2）÷1,024↑
```

KEYSZ

Key length of a B-tree index (bytes)

Refer to Table 6-15: Variable value corresponding to system table under (a) Determining the variable PROC_MNG in (1) Determining the process common memory requirement (for executing the adbreorgsystemdata command) of 6.3.18 Determining the memory requirement for executing the adbreorgsystemdata command, and substitute the value for the table to be processed.

CTRL

Substitute 12 bytes.

page_size

Substitute 4,096 bytes.

(3) Determining the heap memory requirement (for executing the adbreorgsystemdata command)

Use the following formula to determine the heap memory (*HEAP_REORGSZ*) required when executing the `adbreorgsystemdata` command.

Formula (kilobytes)

$$HEAP\_REORGSZ = 505 \times sort\_rthd$$

Explanation of variables

sort_rthd

Substitute 1.

6.3.19 Determining the memory requirement for executing the adbclientdefmang command

When the `adbclientdefmang` command is executed, the HADB server uses the following types of memory. Determine the requirement for each type of memory.

▪ **Shared memory**

- Process common memory (*PROC_CLTDEFMNG*)
- Real thread private memory (*RTHD_CLTDEFMNG*)

▪ **Process memory**

- Heap memory (*HEAP_CLTDEFMNG*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbclientdefmang command)

Use the following formula to determine the process common memory (*PROC_CLTDEFMNG*) required for executing the `adbclientdefmang` command.

Formula (kilobytes)

$$PROC\_CLTDEFMNG = \uparrow(CMI + CDM\_WORK) \div 1{,}024\uparrow + PROC\_AUDINFSZ^{\#}$$

#

Add this value when the audit trail facility is enabled.

Explanation of variables

CMI

For details, see the description of the variable *CMI* in (a) Determining the variable SCI under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

The value of each variable in the formula for determining the variable *CMI* changes according to the contents of the client management definition file used when the `adbclientdefmang` command is executed.

Note that during execution of the `adbclientdefmang` command, both of the following memory areas for the variable *CMI* are concurrently secured:

- Memory area for the variable *CMI* secured before the `adbclientdefmang` command is executed

  This area stores the contents of the client management definition file that existed before the command was executed. Note, however, that this area does not exist if the centralized management of client definitions is not used when the `adbclientdefmang` command is executed.

- Memory area for the variable *CMI* secured when the `adbclientdefmang` command is executed

  This area stores the contents of the client management definition file that exists after the command is executed.

Note that the memory area to be released differs depending on the execution result of the `adbclientdefmang` command.

- **If the adbclientdefmang command terminates normally**

  The HADB server releases the memory area for the variable *CMI* that was secured before the `adbclientdefmang` command was executed.

- **If the adbclientdefmang command results in an error**

  The HADB server releases the memory area for the variable *CMI* that was secured when the `adbclientdefmang` command was executed.

CDM_WORK

Use the following formula to determine this value.

Formula (bytes)

CDM_WORK = ↑8 x *authid_num* / 32↑ x 32 + 64

authid_num

Total number of authorization identifiers specified for the `adbclientmang` operands in the client-managing definition

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

(2) Determining the real thread private memory requirement (for executing the adbclientdefmang command)

Use the following formula to determine the amount of real thread private memory (*RTHD_CLTDEFMNG*) required to execute the `adbclientdefmang` command.

Note that you need to determine the variable *RTHD_CLTDEFMNG* if you use the multi-node function. You do not need to determine that variable if you do not use the multi-node function.

Formula (kilobytes)

RTHD_CLTDEFMNG = ↑*MLT_COMP* / 1,024↑

Explanation of variables

MLT_COMP

Use the following formula to determine this value.

Formula (bytes)

MLT_COMP = ↑(8 + 608 x *authid_num*) / 32↑ x 32 + 64

authid_num

Total number of authorization identifiers specified for the `adbclientmang` operands in the client-managing definition

(3) Determining the heap memory requirement (for executing the adbclientdefmang command)

Use the following formula to determine the heap memory (*HEAP_CLTDEFMNG*) required to execute the `adbclientdefmang` command.

Formula (kilobytes)

HEAP_CLTDEFMNG = ↑(*MNGANA* + *CLTANA*) / 1,024↑

Explanation of variables

MNGANA

Use the following formula to determine this value.

Formula (bytes)

MNGANA = 512 x *cltmng_num*

cltmng_num

Total number of `adbclientmang` operands (for client-managing definition) specified in the client management definition file

CLTANA

Substitute the following value.

Value (bytes)

CLTANA = 10,240

6.3.20 Determining the memory requirement for executing the adbsyndict command

When the `adbsyndict` command is executed, the HADB server uses the following types of memory. Determine the requirement for each type of memory.

▪ **Shared memory**

- Process common memory (*PROC_SYNDICTSZ*)
- Real thread private memory (*RTHD_SYNDICTSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbsyndict command)

Use the following formula to determine the process common memory (*PROC_SYNDICTSZ*) required for executing the `adbsyndict` command.

Formula (kilobytes)

PROC_SYNDICTSZ ＝ *SYNDICDEFSZ* ＋ *PROC_AUDINFSZ*#

#

Add this value when the audit trail facility is enabled.

Explanation of variables

SYNDICDEFSZ

Memory for managing synonym dictionary information

Substitute the following value.

Value (kilobytes)

SYNDICDEFSZ ＝ 3

PROC_AUDINFSZ

Determine the value of the variable *PROC_AUDINFSZ* according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

(2) Determining the real thread private memory requirement (for executing the adbsyndict command)

Use the following formula to determine the amount of real thread private memory (*RTHD_SYNDICTSZ*) required to execute the `adbsyndict` command.

Formula (kilobytes)

RTHD_SYNDICTSZ＝168,800＋↑（↑（*syn_group_num*÷100,000）↑×2,400÷1,024）↑＋*SYNDICACCSZ*

Explanation of variables

syn_group_num

Number of synonym groups

SYNDICACCSZ

Memory for manipulating synonym dictionary information

Substitute the following value.

Value (kilobytes)

```
SYNDICACCSZ = 6
```

6.3.21 Determining the memory requirement for executing the adbaudittrail command

When the `adbaudittrail` command is executed, the HADB server uses the types of memory described in this subsection. Determine the requirement for each type of memory.

▪ Shared memory

- Process common memory (*PROC_AUDTRAILSZ*)
- Real thread private memory (*RTHD_AUDTRAILSZ*)

The following subsections describe the formulas for determining the required amounts of these types of memory.

(1) Determining the process common memory requirement (for executing the adbaudittrail command)

Use the following formula to determine the process common memory (*PROC_AUDTRAILSZ*) required for executing the `adbaudittrail` command.

Formula (kilobytes)

```
PROC_AUDTRAILSZ = AUDINF + AUDTBLDEFSZ
```

Explanation of variables

AUDINF

Audit trail management information

Determine the value according to (r) Determining the variable AUDINF in (3) Determining the process common memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

The memory determined for the *AUDINF* variable is reserved when the audit trail facility is enabled, and released when the facility is disabled.

AUDTBLDEFSZ

Management information for audit target definitions

Add this value only when executing the `adbaudittrail` command with the `--start` option specified. When specifying an option other than `--start`, you do not need to add the *AUDTBLDEFSZ* value.

Substitute the following value.

Value (kilobytes)

```
AUDTBLDEFSZ = 4
```

(2) Determining the real thread private memory requirement (for executing the adbaudittrail command)

Use the following formula to determine the real thread private memory (*RTHD_AUDTRAILSZ*) required for executing the `adbaudittrail` command.

Formula (kilobytes)

```
RTHD_AUDTRAILSZ = AUDTHDINF  + AUDTBLACCSZ
```

Explanation of variables

AUDTHDINF

Determine the value according to the description of the variable *AUDTHDINF* in (4) Determining the real thread private memory requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

The memory determined for the *AUDTHDINF* variable is reserved when the audit trail facility is enabled, and released when the facility is disabled.

AUDTBLACCSZ

Operation information for audit target definitions

Add this value only when executing the `adbaudittrail` command with the `--start` option specified. When specifying an option other than `--start`, you do not need to add the *AUDTBLACCSZ* value.

Substitute the following value.

Value (kilobytes)

```
AUDTBLACCSZ = 1
```

6.3.22 Determining the memory requirement for executing the adbconvertaudittrailfile command

When the `adbconvertaudittrailfile` command is executed, the HADB server uses the following types of memory.

■ Shared memory

- Real thread private memory (*RTHD_CONVERTAUDSZ*)

Use the following formula to determine the value of *RTHD_CONVERTAUDSZ*, which is the amount of real thread private memory required to execute the `adbconvertaudittrailfile` command.

Formula (kilobytes)

```
RTHD_CONVERTAUDSZ = AUDREADSZ + AUDBLOCKSZ
```

Explanation of variables

AUDREADSZ

Audit trail file management area

Substitute the following value.

Value (kilobytes)

AUDREADSZ = 1

AUDBLOCKSZ

Audit trail file work area

Substitute the following value.

Value (kilobytes)

AUDBLOCKSZ = 262,144

6.4 Files that increase continuously over time when using the HADB server

This section describes files that increase continuously over time when using the HADB server.

Files that increase continuously over time when using the HADB server are created in the following directories:

- Server directory
- DB directory
- Audit trail directory#

#

This directory is required to use the audit trail facility.

Important

There is no upper limit to the number and file size of files that increase continuously over time. Therefore, with files of this nature, you need to take care to avoid a situation in which there is insufficient free space on the disk.

The following table shows the files in the server directory that increase continuously over time.

Table 6-18: Files in server directory that increase continuously over time

No.	File name	File description	File deletion method
1	`$ADBDIR/spool/adbdump`*YYYYMMDDhhmmss.server-process-process-id*#1	HADB dump file	• Deleted when the HADB server terminates normally. • Deleted when the `adbinfosweep` command is executed.
2	`$ADBDIR/spool/adbdumperror`*YYYYMMDDhhmmssSSSSSS_TTTTTTTTTTTTTTTTTTT.server-process-process-id*#1,#2		Deleted when the `adbinfosweep` command is executed.

#1

YYYYMMDDhhmmss in the file name indicates the time when the file was generated.

#2

SSSSSS in the file name indicates the microsecond portion of the time when the HADB dump file was generated. *TTTTTTTTTTTTTTTTTTT* indicates the thread ID of the real thread in which the HADB server's internal conflict error was detected.

The following table shows the files in the DB directory that increase continuously over time.

Table 6-19: Files that DB directory that increase continuously over time

No.	File name	File description	File deletion method
1	`$DBDIR/SPOOL/core.`*server-process-process-id*#1	Error information (core files)	Deleted when the `adbinfosweep` command is executed.

No.	File name	File description	File deletion method
2	$DBDIR/ADBWORK/*xxxxx*[#2,#3]	Temporary work files created during command execution[#4]	Deleted when the command finished executing.

#1

If the adb_core_path operand in the server definition is specified, error information (core files) is output to the directory specified in the adb_core_path operand instead of to the $DBDIR/SPOOL directory.

#2

xxxxx in the file name is an arbitrary value.

#3

If the executed command omits the -w option, temporary work files are created in this directory. If the -w option is specified for the command, temporary work files are created in the directory specified in the -w option.

#4

Temporary work files are created when you execute the following commands:

- adbimport command
- adbidxrebuild command
- adbmergechunk command
- adbunarchivechunk command
- adbreorgsystemdata command

The following table shows the files in the audit trail directory that increase continuously over time.

Table 6-20: Files in audit trail directory that increase continuously over time

No.	File name	File description	File deletion method
1	$AUDITDIR/adbaud-*YYYYMMDD-hhmmss-nnn*.aud[#1,#2]	Audit trail files[#3]	Use the cp and rm OS commands to move the audit trail file to the audit trail storage directory. For details, see 12.3.1 Moving audit trail files (to audit trail storage directory).

#1

- $AUDITDIR in the file name represents the audit trail directory. It is the path name specified in the adb_audit_log_path operand in the server definition.
- *YYYYMMDD* and *hhmmss* in the file name are the date and time at which the audit trail file was renamed.
- *nnn* in the file name is the millisecond component of the time at which the audit trail file was renamed.

#2

When using the multi-node function, the file name is $AUDITDIR/adbaud-*YYYYMMDD-hhmmss-nnn-N*.aud. *N* is the node number of the node that output the audit trail file.

#3

If 0 is specified for the adb_audit_log_max_num operand in the server definition or the adb_audit_log_max_num operand is omitted, there is no limit to the number of audit trail files that can be created.

6.5 Estimating the size of the server message log file

This section explains how to estimate the size of the server message log file. For details about the server message log file, see (3) Server message log file in 10.4.1 Message output destinations.

Use the following formula to determine the size of the server message log file.

Formula (kilobytes)

```
server message log file size = (env_msglog_size × 1,024 × 4)
```

Explanation of variables

- *env_msglog_size*
  Value specified for the environment variable ADBMSGLOGSIZE
  If no value is specified, assume 16.

6.6 Estimating the size of error information (core file)

This section explains how to estimate the size of error information (core file).

Determine the size of error information (core file) from the following formula.

Formula (kilobytes)

error information (core file) size = 2,097,152

6.7 Estimating the size of HADB dump files

An HADB dump file contains the troubleshooting information that is output when the server process of the HADB server terminates abnormally.

Use the following formula to determine the size of the HADB dump file.

Formula (megabytes)

```
size-of-HADB-dump-files = 25,600 + max_users × 106 + rthd_num × 1,821
```

Explanation of the variables

max_users

Value specified for the `adb_sys_max_users` operand in the server definition

rthd_num

Value specified for the `adb_sys_rthd_num` operand in the server definition

6.8 Estimating the size of a statistics log file

This section explains how to estimate the size of a statistics log file.

Use the following formula to determine the size of a statistics log file.

Formula (kilobytes)

```
size-of-statistics-log-file = sta_log_max_size × sta_log_size_unit × 1,024 × 4 + 1
,024
```

Explanation of the variables

sta_log_max_size

Value specified for the `adb_sta_log_max_size` operand in the server definition

For details, see the description of the adb_sta_log_max_size operand in 7.2.7 Operands related to statistical information (set format).

sta_log_size_unit

- If the `adb_sta_log_size_unit` operand is omitted or `G` is specified for the `adb_sta_log_size_unit` operand in the server definition:

  Substitute `1,024`.

- If `M` is specified for the `adb_sta_log_size_unit` operand in the server definition:

  Substitute `1`.

For details about the `adb_sta_log_size_unit` operand in the server definition, see the description of the adb_sta_log_size_unit operand in 7.2.7 Operands related to statistical information (set format).

Note that the output destination of statistics log files differs depending on whether the `adb_sta_log_path` operand is specified in the server definition.

- If the `adb_sta_log_path` operand is not specified in the server definition

  Statistics log files are output to the `$ADBDIR/spool` directory.

- If the `adb_sta_log_path` operand is specified in the server definition

  Statistics log files are output to the directory specified for the `adb_sta_log_path` operand in the server definition.

For details about the `adb_sta_log_path` operand in the server definition, see the description of the adb_sta_log_path operand in 7.2.7 Operands related to statistical information (set format).

6.9 Estimating the size of an SQL trace file

This section explains how to estimate the size of an SQL trace file.

Use the following formula to determine the size of an SQL trace file.

Formula (kilobytes)

Total capacity of the SQL trace file = *sqltxt_size* × 1,024 × 8

Explanation of the variables

sqltxt_size

Value specified for the `adb_sql_trc_txtfile_size` operand in the server definition

For details, see the explanation of the adb_sql_trc_txtfile_size operand in 7.2.5 Operands related to SQL statements (set format).

6.10 Estimating the size of files required for the updated-row columnizing facility

This section explains the size of files required for the updated-row columnizing facility.

The following shows the total size of files required for the updated-row columnizing facility (*COLUMNIZELOG*).

Total size of files (kikobytes)

```
COLUMNIZELOG = 131,072
```

Note

- The files required for the updated-row columnizing facility are created when the HADB server starts regardless of whether the updated-row columnizing facility is enabled.
- For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).

6.11 Estimating the size of access path search information log files

This section explains the size of access path search information log files.

The following shows the total size of access path search information log files (*OPTLOG*).

Total size of files (kilobytes)

```
OPTLOG = 131,072 × 2
```

Access path search information log files are used by the system and created when the HADB server starts. The maintenance information about the building of access paths is output to these files when SQL statements are executed.

6.12 Estimating the size of the system log files

This section explains how to estimate the size of the system log files.

System log files consist of a master log file and a set of user log files. The master log file and user log files are created with their initial size in the DB directory (`$DBDIR/ADBSYS/ADBSLG`) when the HADB server starts. The master log file is created automatically, and the user log files are created with the value specified in the server definition. Therefore, to determine the size of the system log files, you need to determine the sizes of the master log file and the user log files.

When an increase in the size of the system logs causes a system log file to exceed its initial capacity, the system log file is automatically extended. Because extending a system log file greatly impacts update performance, make sure that you estimate the size of the system log files so that you can specify an initial capacity that will accommodate your needs.

Use the formula shown below to determine the size of the system log files. Because unusable areas might occur in system log files, specify a size that is at least 1.2 times the value determined here.

Formula (kilobytes)

```
system log file size = MSTLOG + USRLOG × user_file_num
```

Explanation of variables

- *MSTLOG*
  Size of the master log file (kilobytes)
  Determine the value as explained in 6.12.1 Determining the size of the master log file.
- *USRLOG*
  Size of user log files (kilobytes)
  Determine the value as explained in 6.12.2 Determining the size of the user log files.
- *user_file_num*
  Number of user log files (files)
  Determine the value as explained in 6.12.15 Determining the number of user log files.

6.12.1 Determining the size of the master log file

Event log data is output to the master log file. Because event log data is output continuously while the HADB server is running, the size of the master log file will increase over time. However, the size of the master log file will be automatically reduced if both of the following conditions are met during operation of the HADB server:

- The size of the master log file exceeds 2 MB
- An update transaction is committed while there are no other pending update transactions

When the HADB server is restarted, the event log data that was output the last time the HADB server was running is deleted from the master log file.

Important

If there is insufficient free space on the disk where the master log file is created, issues such as an inability to commit or recover a transaction resulting in abnormal termination of the HADB server might occur. Therefore, take care to estimate the size required for the master log file accurately, and make sure that enough disk space is allocated.

Use the following formula to determine the size of the master log file.

Tip

Determine the estimation period that appears in the explanation of the variables based on one of the following:

- The time period during which the HADB server is running
- The time period after which the master log file is automatically reduced in size

For example, if the HADB server is stopped once every six months for server maintenance or other reasons, use six months as the estimation period. If a scenario in which the master log file can be automatically reduced in size occurs every day, use one day as the estimation period.

Formula (kilobytes)

$$\textit{size of the master log file} = \mathrm{MAX}\left\{16, \left\lceil \frac{\left(\textit{tran\_count} + \left(\sum_{i=1}^{\textit{utlexec\_count}} \textit{UTi}\right)\right) \times 184 + \textit{arcdir\_remove\_count} \times 712}{65{,}512} \right\rceil + 1 \right\} \times 64$$

Explanation of variables

tran_count

Number of times update transactions are executed during the estimation period

arcdir_remove_count

Number of times the files in the archive directory are deleted during the estimation period

Substitute the number of times an archive directory is deleted by executing a relevant SQL statement or command is executed for an archivable multi-chunk table. The following table shows the number of times an archive directory is deleted.

Table 6-21: Number of times an archive directory is deleted

No.	SQL statement or command to be executed	Number of times an archive directory is deleted
1	• `DROP SCHEMA` statement • `REVOKE` statement with `SCHEMA` specified	Substitute the number of archivable multi-chunk tables that exist in the schema to be deleted.
2	• `ALTER TABLE` statement (when executed to change an archivable multi-chunk table to a regular multi-chunk table) • `DROP TABLE` statement • `TRUNCATE TABLE` statement	Substitute 1.
3	`PURGE CHUNK` statement	Substitute the number of chunks to be deleted.
4	`adbunarchivechunk` command	Substitute the number of chunks to be processed.

utlexec_count

Number of times commands are executed during the estimation period

This is the sum total of the number of times the following commands are executed:

- adbimport command
- adbidxrebuild command
- adbmodarea command
- adbmergechunk command
- adbgetcst command
- adbchgchunkcomment command
- adbchgchunkstatus command
- adbarchivechunk command
- adbunarchivechunk command
- adbreorgsystemdata command
- adbsyndict command

UTi

Number of database-updating threads for each command that is executed during the estimation period
Use the following formula to determine the number of database-updating threads for each command.

- **Formula (when the adbimport command is executed)**

  UTi (when executing adbimport command) =
  (*value-specified-for*-adb_import_rthd_num-*import-option* + 1) × 9

- **Formula (when the adbidxrebuild command is executed)**

  UTi (when executing adbidxrebuild command) =
  (*value-specified-for*-adb_idxrebuild_rthd_num-*index-rebuild-option* + 1) × 6

- **Formula (when the adbmodarea command is executed)**

  UTi (when the adbmodarea command is executed) = 2

- **Formula (when the adbmergechunk command is executed)**

  UTi (when executing adbmergechunk command) =
  (*value-specified-for*-adb_mergechunk_rthd_num-*merge-chunk-option* + 1) × 2

- **Formula (when the adbgetcst command is executed)**

  UTi (when the adbgetcst command is executed) = 1

- **Formula (when the adbchgchunkcomment command is executed)**

  UTi (when the adbchgchunkcomment command is executed) = 1

- **Formula (when the adbchgchunkstatus command is executed)**

  UTi (when the adbchgchunkstatus command is executed) = 1

- **Formula (when the adbarchivechunk command is executed)**

UTi (when executing `adbarchivechunk` command) =
(*value-specified-for-*`adb_arcv_rthd_num`*-archive-chunk-option* + 1)
× *number of chunks to be processed*

- **Formula (when the adbunarchivechunk command is executed)**

UTi (when executing `adbunarchivechunk` command) =
(*value-specified-for-*`adb_unarcv_rthd_num`*-unarchive-chunk-option* + 1) × 11
× *number of chunks to be processed*

- **Formula (when the adbreorgsystemdata command is executed)**

UTi（when the `adbreorgsystemdata` command is executed）＝ 10

- **Formula (when the adbsyndict command is executed)**

UTi（when the `adbsyndict` command is executed）＝ 1

Note

For details about each of these commands, see the manual *HADB Command Reference*.

6.12.2 Determining the size of the user log files

Any time a database is updated, a user log containing database update history information is stored in a user log file. If a rollback occurs, the HADB server uses the user logs to recover the transaction.

Use the following formula to determine the size of the user log files.

Formula (kilobytes)

$$\textit{user log file size} = MAX\left\{\textit{user\_file\_init\_size} \times 1{,}024, \left(\left\uparrow \frac{\textit{max\_user\_log}}{65{,}512} \right\uparrow + 1 \right) \times 64 \right\}$$

Explanation of variables

- *user_file_init_size*
  Value specified for the `adb_log_usrfile_size` operand in the server definition
- *max_user_log*
  Size of the user log for the transaction that outputs the largest volume of user log data (bytes)

When determining the variable *max_user_log*, first determine the size of the user log for the transactions in each of the processes described below. Then substitute the value of the largest user log size in the variable *max_user_log*.

- 6.12.3 Determining the size of the user logs that are output during execution of a definition SQL statement (variable max_user_log)
- 6.12.4 Determining the size of the user logs that are output during execution of the adbimport command (variable max_user_log)

- 6.12.5 Determining the size of the user logs that are output during execution of the adbidxrebuild command (variable max_user_log)
- 6.12.6 Determining the size of the user logs that are output during execution of the adbmergechunk command (variable max_user_log)
- 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log)
- 6.12.8 Determining the size of the user logs that are output during execution of the TRUNCATE TABLE statement (variable max_user_log)
- 6.12.9 Determining the size of the user logs that are output during execution of the PURGE CHUNK statement (variable max_user_log)
- 6.12.10 Determining the size of the user logs that are output during execution of the adbchgchunkstatus command (variable max_user_log)
- 6.12.11 Determining the size of the user logs that are output during execution of the adbarchivechunk command (variable max_user_log)
- 6.12.12 Determining the size of the user logs that are output during execution of the adbunarchivechunk command (variable max_user_log)
- 6.12.13 Determining the size of the user logs that are output during execution of the adbreorgsystemdata command (variable max_user_log)
- 6.12.14 Determining the size of user logs that are required for the maintenance processing of the updated-row columnizing facility (variable max_user_log)

Tip

User logs are deleted when transaction processing ends. Therefore, calculate the user log file size based on the size of the user log for the transaction that outputs the largest volume of user log data.

Note

For SQL statements and commands not listed above, there is no need to estimate the size of user logs since the user logs that are output are extremely small.

6.12.3 Determining the size of the user logs that are output during execution of a definition SQL statement (variable max_user_log)

This subsection explains how to determine the size of the user logs that are output during execution of any of the following definition SQL statements:

- `CREATE TABLE` statement (base table definition)
- `CREATE INDEX` statement (index definition)
- `CHANGE OPTION CHUNK` of the `ALTER TABLE` statement (change to the maximum number of chunks)
- `ALTER TABLE` statement (when executed to change an archivable multi-chunk table to a regular multi-chunk table)
- `DROP USER` statement (HADB user deletion)
- `DROP SCHEMA` statement (schema deletion)
- `DROP TABLE` statement (base table deletion)

- DROP INDEX statement (index deletion)
- SCHEMA of the REVOKE statement (revoking schema operation privileges)

For the definition SQL statements other than the preceding ones, there is no need to estimate the size of user log files because user log data larger than the user log file capacity is not output.

(1) Determining the size of the user logs that are output during execution of the CREATE TABLE statement

Use the following formula to determine the size of the user logs that are output during execution of the CREATE TABLE statement (variable *CRTTBLLOG*).

Formula (bytes)

$$CRTTBLLOG = 1{,}296 + \left\{ dbarea\_file\_num \times 6 \right\} \times page\_size$$

Explanation of variables

dbarea_file_num: Number of DB area files in the data DB area for storing the table to be defined

page_size: Page size in the data DB area for storing the table to be defined (bytes)

(2) Determining the size of the user logs that are output during execution of the CREATE INDEX statement

Use the following formula to determine the size of the user logs that are output during execution of the CREATE INDEX statement (variable *CRTIDXLOG*).

Formula (bytes)

$$CRTIDXLOG = 812 + \left\{ dbarea\_file\_num \times 6 \right\} \times page\_size$$

Explanation of variables

dbarea_file_num: Number of DB area files in the data DB area for storing the indexes to be defined

page_size: Page size in the data DB area for storing the indexes to be defined (bytes)

(3) Determining the size of the user logs that are output during execution of CHANGE OPTION CHUNK of the ALTER TABLE statement

Determine the size of the user logs output during execution of CHANGE OPTION CHUNK of the ALTER TABLE statement based on the explanation in (7) Determining the size of the user logs that are output during the execution of the DROP TABLE statement. When doing so, assume 0 for the variable *SGDATA*.

If indexes are defined for the table to be processed, determine the size of the user logs for each defined index based on the explanation in (8) Determining the size of the user logs that are output during the execution of the DROP INDEX statement. When doing so, assume 0 for the variable *SGIDX*.

The sum total of these determined values becomes the size of the user logs that are output during execution of CHANGE OPTION CHUNK of the ALTER TABLE statement.

(4) Determining the size of user log data output when the ALTER TABLE statement is executed (when an archivable multi-chunk table is changed to a regular multi-chunk table)

When the ALTER TABLE statement is used to change an archivable multi-chunk table to a regular multi-chunk table, the location table and the index defined for the location table are deleted. Therefore, you must determine the size of user log data that is output when the location table and the index defined for the location table are deleted.

Note that the location table and the index defined for the location table are stored in the data DB area that stores the target archivable multi-chunk table.

Use the following formula to determine the value of the variable *ALTTBLLOG* as the size of user log data that is output when the ALTER TABLE statement is executed to change an archivable multi-chunk table to a regular multi-chunk table.

Formula (bytes)

ALTTBLLOG = DRPLTBLLOG + DRPLIDXLOG

Explanation of variables

DRPLTBLLOG

Location table deletion log

Use the following formula to determine this value.

Formula (bytes)

$$DRPLTBLLOG = 263{,}712 + \left\lceil \frac{\left(dbarea\_file\_num \times 6 + \left\lceil \frac{SGDATA}{SEGBF} \right\rceil + \left\lceil \frac{SGDATA}{\left\lfloor \frac{page\_size \times 125}{16} \right\rfloor \times SEGBF} \right\rceil \times page\_size \right)}{usrlog\_file\_num} \right\rceil$$

dbarea_file_num

Number of data DB area files for the data DB area that stores the location table to be deleted

SGDATA

Number of segments of the data DB area that stores the location table to be deleted

SEGBF

Segment block factor of the data DB area that stores the location table to be deleted

page_size

Page size of the data DB area that stores the location table to be deleted (bytes)

usrlog_file_num

Use the following formula to determine this value.

usrlog_file_num =
number of user log files required to execute the ALTER TABLE statement# - 1

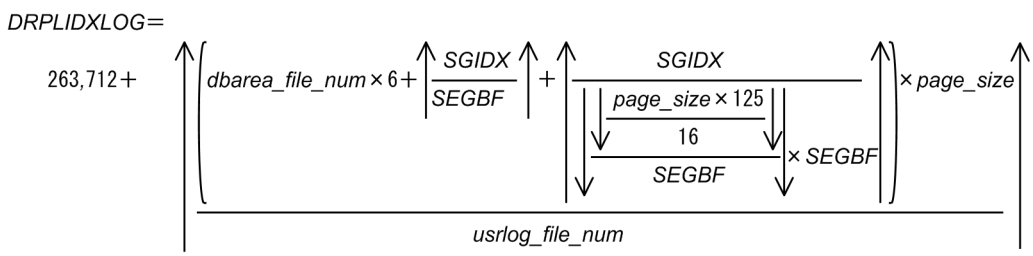
#
The ALTER TABLE statement that is used to change an archivable multi-chunk table to a regular multi-chunk table applies.

DRPLIDXLOG

Location-table index deletion log

Use the following formula to determine this value.

Formula (bytes)



dbarea_file_num

Number of data DB area files for the data DB area that stores the index defined for the location table to be deleted

SGIDX

Number of segments of the data DB area that stores the index defined for the location table to be deleted

SEGBF

Segment block factor of the data DB area that stores the index defined for the location table to be deleted

page_size

Page size of the data DB area that stores the index defined for the location table to be deleted (bytes)

usrlog_file_num

Use the following formula to determine this value.

usrlog_file_num =
number of user log files required to execute the ALTER TABLE statement# - 1

#
The `ALTER TABLE` statement that is executed to change an archivable multi-chunk table to a regular multi-chunk table applies.

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required when the `ALTER TABLE` statement is executed to change an archivable multi-chunk table to a regular multi-chunk table, see (1) SQL statements for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(5) Determining the size of the user logs that are output during the execution of the DROP USER statement

If there are schemas owned by HADB users who are subject to deletion, determine the size of the user logs for the schemas to be deleted based on the explanation in (6) Determining the size of the user logs that are output during the execution of the DROP SCHEMA statement.

(6) Determining the size of the user logs that are output during the execution of the DROP SCHEMA statement

If tables are defined in the schema to be deleted, determine the user log size for each of the tables to be deleted based on the explanation in (7) Determining the size of the user logs that are output during the execution of the DROP TABLE statement. Then, determine their sum total value.

If no table is defined in the schema to be deleted, there is no need to estimate the size of user log files because user logs are not output if they exceed the user log file capacity.

(7) Determining the size of the user logs that are output during the execution of the DROP TABLE statement

Use the following formula to determine the size of the user logs that are output during execution of the `DROP TABLE` statement (variable *DRPTBLLOG*).

Formula (bytes)

$$DRPTBLLOG = \left\lceil \left(263{,}712 + \left\lceil dbarea\_file\_num \times 6 + \left\lceil \frac{SGDATA}{SEGBF} \right\rceil + \left\lceil \frac{SGDATA}{\left\lceil \frac{\left\lceil \frac{page\_size \times 125}{16} \right\rceil}{SEGBF} \right\rceil \times SEGBF} \right\rceil \times page\_size \right) \right\rceil \Big/ usrlog\_file\_num \right\rceil$$

Explanation of variables

dbarea_file_num
: Number of DB area files in data DB area that stores the table to be deleted

SGDATA
: Number of segments in the data DB area that stores the table to be deleted

SEGBF
: Segment block factor of the data DB area that stores the table to be deleted

page_size
: Page size of the data DB area that stores the table to be deleted (bytes)

usrlog_file_num
: Use the following formula to determine its value.

$$usrlog\_file\_num = number\ of\ user\ log\ files\ required\ at\ execution\ of\ the\ DROP\ TABLE\ statement - 1$$

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `DROP TABLE` statement, see (1) SQL statements for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

Important

Add the value determined in (8) Determining the size of the user logs that are output during the execution of the DROP INDEX statement if an index is defined for the table to be deleted.

Add the value determined in (4) Determining the size of user log data output when the ALTER TABLE statement is executed (when an archivable multi-chunk table is changed to a regular multi-chunk table) if the table to be deleted is an archivable multi-chunk table.

(8) Determining the size of the user logs that are output during the execution of the DROP INDEX statement

Use the following formula to determine the size of the user logs that are output during execution of the `DROP INDEX` statement (variable *DRPIDXLOG*).

Formula (bytes)

$$DRPIDXLOG = 263{,}712 + \left\lceil \left(dbarea\_file\_num \times 6 + \left\lceil \frac{SGIDX}{SEGBF} \right\rceil + \left\lceil \frac{SGIDX}{\left\lceil \frac{\left\lceil \frac{page\_size \times 125}{16} \right\rceil}{SEGBF} \right\rceil \times SEGBF} \right\rceil \times page\_size \right) \Big/ usrlog\_file\_num \right\rceil$$

Explanation of variables

dbarea_file_num
Number of DB area files in data DB area that stores the index to be deleted

SGIDX
Number of segments in the data DB area that stores the index to be deleted

SEGBF
Segment block factor of the data DB area that stores the index to be deleted

page_size
Page size of the data DB area that stores the index to be deleted (bytes)

usrlog_file_num
Use the following formula to determine its value.

usrlog_file_num =
number of user log files required at execution of the DROP INDEX statement − 1

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `DROP INDEX` statement, see (1) SQL statements for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(9) Determining the size of the user logs that are output during execution of SCHEMA in the REVOKE statement

Determine the size of the user logs output during execution of `SCHEMA` of the `REVOKE` statement based on the explanation in (6) Determining the size of the user logs that are output during the execution of the DROP SCHEMA statement.

6.12.4 Determining the size of the user logs that are output during execution of the adbimport command (variable max_user_log)

Use the following formula to determine the size of the user logs that are output during the execution of the `adbimport` command.

Formula (bytes)

Size of the user logs that are output during execution of the adbimport command =

$$\mathrm{MAX}\left\{TBLPRGLOG + IDXPRGLOG,\ IMPTBLLOG,\ IMPSTBLLOG + IMPRNGIDXLOG,\ IMPIDXLOG + IMPTIXLOG\right\}$$

Explanation of variables

- *TBLPRGLOG*
  Row data deletion log for the table to be imported
  See (1) Determining the variable TBLPRGLOG.
- *IDXPRGLOG*
  Index data deletion log for the table to be imported
  See (2) Determining the variable IDXPRGLOG.
- *IMPTBLLOG*
  Row data storage log for the table to be imported
  See (3) Determining the variable IMPTBLLOG.
- *IMPSTBLLOG*
  Update log for the table `STATUS_CHUNKS`
  See (4) Determining the variable IMPSTBLLOG.
- *IMPRNGIDXLOG*
  Range index data creation log for the table to be imported
  See (5) Determining the variable IMPRNGIDXLOG.
- *IMPIDXLOG*
  B-tree index data creation log for the table to be imported
  See (6) Determining the variable IMPIDXLOG.
- *IMPTIXLOG*
  Text index data creation log for the table to be imported
  See (7) Determining the variable IMPTIXLOG.

(1) Determining the variable TBLPRGLOG

Use the formula below to determine the value of variable *TBLPRGLOG*.

Note that in the following cases, you do not need to estimate the value of this variable. Substitute 0 for *TBLPRGLOG*.

- When the `adbimport` command is executed with the `-d` option omitted for a single-chunk table
- When the `adbimport` command is executed with the `-d` and `-b` options omitted for a multi-chunk table
- When the `adbimport` command is executed with the `-b` option specified and the `-d` option omitted for a multi-chunk table

Formula (bytes)

$$TBLPRGLOG = \left\lceil \frac{\left(dbarea\_file\_num \times 6 + \left\lceil \frac{SGTBL}{SEGBF} \right\rceil + \left\lceil \frac{SGTBL}{\left\lfloor \frac{\left\lfloor \frac{page\_size \times 125}{16} \right\rfloor}{SEGBF} \right\rfloor \times SEGBF} \right\rceil \right) \times page\_size}{usrlog\_file\_num} \right\rceil$$

Explanation of variables

dbarea_file_num
: Number of DB area files in data DB area that stores the table

SGTBL
: Number of segments used to store the table

SEGBF
: Segment block factor of the data DB area used to store the table

page_size
: Page size of the data DB area used to store the table (bytes)

usrlog_file_num
: Use the following formula to determine its value.

$$usrlog\_file\_num = \text{number of user log files required at execution of the adbimport command} - 1$$

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `adbimport` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(2) Determining the variable IDXPRGLOG

Use the formula below to determine the value of variable *IDXPRGLOG*.

Note that in the following cases, you do not need to estimate the value of this variable. Substitute 0 for *IDXPRGLOG*.

- When the `adbimport` command is executed with the `-d` option omitted for a single-chunk table
- When the `adbimport` command is executed with the `-d` and `-b` options omitted for a multi-chunk table
- When the `adbimport` command is executed with the `-b` option specified and the `-d` option omitted for a multi-chunk table

Formula (bytes)

$$IDXPRGLOG = \sum_{i=1}^{idx\_num} \left\{ \left\lceil \left(dbarea\_file\_num_{(i)} \times 6 + \left\lceil \frac{SGIDX_{(i)}}{SEGBF_{(i)}} \right\rceil + \left\lceil \frac{SGIDX_{(i)}}{\left\lceil \left\lceil \frac{page\_size_{(i)} \times 125}{16} \right\rceil \times SEGBF_{(i)} \right\rceil} \right\rceil \times page\_size_{(i)} \right) \middle/ usrlog\_file\_num \right\rceil \right\}$$

Explanation of variables

idx_num
: Number of indexes defined for the table to be deleted

dbarea_file_num$_{(i)}$
: Number of DB area files in data DB area that stores the indexes

SGIDX$_{(i)}$
: For each index, number of segments that store indexes

SEGBF$_{(i)}$
: For each index, segment block factor for the data DB area

page_size$_{(i)}$
: Page size in the data DB area of each index (bytes)

usrlog_file_num
: Use the following formula to determine its value.

$$usrlog\_file\_num = \text{number of user log files required at execution of the adbimport command} - 1$$

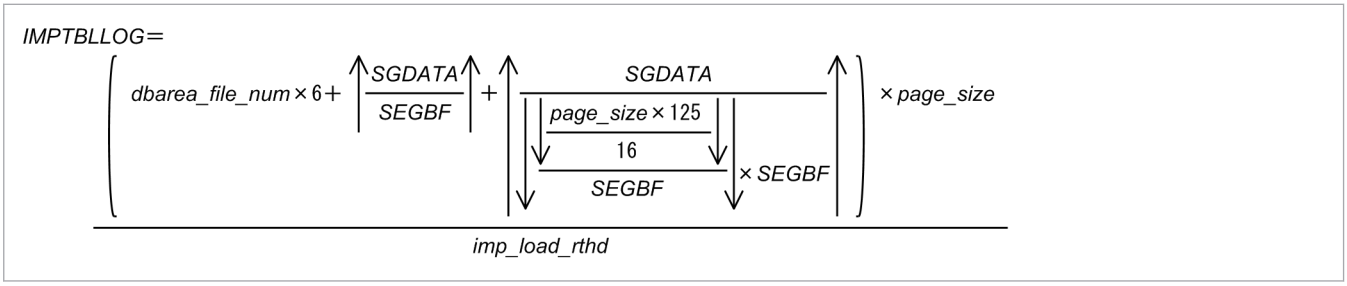
For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `adbimport` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(3) Determining the variable IMPTBLLOG

Use the following formula to determine the value of variable *IMPTBLLOG*.

Formula (bytes)



Explanation of variables

- *dbarea_file_num*
  Number of DB area files in data DB area that stores the table
- *SGDATA*
  Number of segments that store the table to be allocated during import processing
- *SEGBF*
  Segment block factor of the data DB area that stores the table
- *page_size*
  Page size of the data DB area used to store the table (bytes)
- *imp_load_rthd*
  Use the following formula to determine the value:

  ```
  value-specified-for-import-option-adb_import_rthd_num - 1
  ```

  If the import option `adb_import_rthd_num` is omitted, substitute the number of threads used when the import option `adb_import_rthd_num` is omitted. For details, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Estimate the number of segments to be allocated based on the size of the data to be imported.

(4) Determining the variable IMPSTBLLOG

Use the formula below to determine the value of variable *IMPSTBLLOG*.

Note that in the following cases, you do not need to estimate the value of this variable. Substitute 0 for *IMPSTBLLOG*.

- When the `adbimport` command is executed for a single-chunk table
- When the `adbimport` command is executed with the `-d` and `-b` options omitted for a multi-chunk table

Formula (bytes)

```
IMPSTBLLOG = IMPSTBLPRGLOG + IMPSTBLUPDLOG
```

In the following case, a value needs to be estimated for variable *IMPSTBLPRGLOG* only:

- When the `adbimport` command is executed with the `-d` option specified for a multi-chunk table

In the following case, a value needs to be estimated for variable *IMPSTBLUPDLOG* only:

- When the adbimport command is executed with the -b option specified and the -d option omitted for a multi-chunk table

Explanation of variables

IMPSTBLPRGLOG

Update log created when data is deleted from or added to the STATUS_CHUNKS table

Use the following formula to determine its value.

Formula (bytes)

$$IMPSTBLPRGLOG = 488 \times \left\{ chunk\_num + 1 \right\}$$

chunk_num

Number of chunks created

IMPSTBLUPDLOG

Update log created when data is updated in or added to the STATUS_CHUNKS table

Substitute the following value.

Value (bytes)

```
IMPSTBLUPDLOG = 1,464
```

(5) Determining the variable IMPRNGIDXLOG

Use the following formula to determine the value of variable *IMPRNGIDXLOG*.

Formula (bytes)

$$IMPRNGIDXLOG = MAX \left\{ IMPRNGIDX_{(1)}LOG, IMPRNGIDX_{(2)}LOG, \ldots, IMPRNGIDX_{(n)}LOG \right\}$$

Explanation of variables

- *n*

  Number of range indexes defined for the table to be imported

- $IMPRNGIDX_{(i)}LOG$

  Log size of the i-th range index defined for the table to be imported

  Determine this value from the formula shown below. However, the formula depends on whether the adbimport command is executed with the -d option specified (whether the creation mode or the addition mode is used).

■ Formula (when the -d option is specified (creation mode is used)) (bytes)

$$IMPRNGIDX_{(i)}LOG = DIRRNGPGNO_{(i)} \times page\_size_{(i)}$$

- $page\_size_{(i)}$

  Page size in the data DB area for storing the i-th range index defined for the table to be imported (bytes)

- $DIRRNGPGNO_{(i)}$

Number of pages in the management area for the i-th range index defined for the table to be imported
Use the following formula to determine its value:

$$DIRRNGPGNO_{(i)} = dbarea\_file\_num_{(i)} \times 6 + \left\lceil \frac{SGRNGDATA_{(i)}}{SEGBF_{(i)}} \right\rceil + \left\lceil \left(\left\lceil \frac{page\_size_{(i)} \times 125}{16} \right\rceil \div SEGBF_{(i)} \right) \times SEGBF_{(i)} \right\rceil$$

- $dbarea\_file\_num_{(i)}$
  Number of DB area files in the data DB area for storing range indexes (files)
- $SGRNGDATA_{(i)}$
  Number of range index segments to be allocated in the data import processing
  For details, see 5.8.6 Determining the number of segments for storing each range index.
- $SEGBF_{(i)}$
  Segment blocking factor for the data DB area for storing range indexes
  See the explanation of the variable *SEGBF* in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area.
- *page_size*(*i*)
  Page size in the data DB area for storing the i-th range index defined for the table to be imported (bytes)

■ **Formula (when the -d option is omitted (addition mode is used)) (bytes)**

$$IMPRNGIDX_{(i)}LOG = \left\{ DIRRNGPGNO_{(i)} + OLDSGRNGDATA \times SEGSIZE_{(i)} \right\} \times page\_size_{(i)}$$

- $DIRRNGPGNO_{(i)}$
  Number of pages in the management area for the i-th range index defined for the table to be imported
  See the variable $DIRRNGPGNO_{(i)}$ in *Formula (when the -d option is specified (creation mode is used))*.
- *OLDSGRNGDATA*
  Number of segments in the data DB area for storing the range indexes that had been allocated before the data was imported
  For details, see the description of the variable *RS*(*i*) in 5.8.6 Determining the number of segments for storing each range index.
- $SEGSIZE_{(i)}$
  Segment size in the data DB area for storing the i-th range index defined for the table to be imported
  Use the following formula to determine its value.

  $$SEGSIZE = 4{,}194{,}304 \div page\_size_{(i)}$$

- $page\_size_{(i)}$
  Page size in the data DB area for storing the i-th range index defined for the table to be imported (bytes)

(6) Determining the variable IMPIDXLOG

Use the following formula to determine the value of variable *IMPIDXLOG*.

Formula (bytes)

$$IMPIDXLOG = IMPIDX_{(1)}LOG + IMPIDX_{(2)}LOG + \ \ldots \ + IMPIDX_{(n)}LOG$$

Explanation of variables

- *n*
  Number of B-tree indexes defined for the table to be imported
- $IMPIDX_{(i)}LOG$
  Log size of the i-th B-tree index defined for the table to be imported
  Use the following formula to determine its value.
  Formula (bytes)

$$IMPIDX_{(i)}LOG = \left(\frac{(DIRPGNO_{(i)} + unique\_key\_num) \times page\_size_{(i)}}{imp\_dividx\_rthd} \right)$$

- *unique_key_num*
  Number of unique key values
  • Substitute 0 if the `-d` option is specified for the `adbimport` command.
  • In all other cases, for the i-th B-tree index defined for the table to be imported, remove duplicates from the key values generated from the imported data. Then, use a number that is between the minimum and maximum key values already stored in the index.
- $page\_size_{(i)}$
  Page size of the data DB area that stores the i-th B-tree index (bytes)
- *imp_dividx_rthd*
  Use the following formula to determine the value:

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

  If the import option `adb_import_rthd_num` is omitted, substitute the number of threads used when the import option `adb_import_rthd_num` is omitted. For details, see *adbimport (Import Data)* in the manual *HADB Command Reference*.
- $DIRPGNO_{(i)}$
  Number of pages in the management area for the i-th B-tree index defined for the table to be imported
  Use the following formula to determine its value.
  Formula

$$DIRPGNO_{(i)} = dbarea\_file\_num_{(i)} \times 6 + \left\uparrow \frac{SGIDX_{(i)}}{SEGBF_{(i)}} \right\uparrow + \left\uparrow \frac{SGIDX_{(i)}}{\left\downarrow \frac{\left\downarrow \frac{page\_size_{(i)} \times 125}{16} \right\downarrow}{SEGBF_{(i)}} \right\downarrow \times SEGBF_{(i)}} \right\uparrow$$

- $dbarea\_file\_num_{(i)}$
  Number of DB area files in data DB area that stores the B-tree indexes
- $SGIDX_{(i)}$
  Number of segments that store the i^{th} B-tree index to be allocated during import processing

- $SEGBF_{(i)}$
  Segment block factor of the data DB area that stores the i^{th} B-tree indexes

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Estimate the number of segments to be allocated based on the size of the data to be imported.

(7) Determining the variable IMPTIXLOG

Use the following formula to determine the value of variable *IMPTIXLOG*.

Formula (bytes)

$$IMPTIXLOG = \mathrm{MAX}\left\{ IMPTIX_{(1)}LOG,\ IMPTIX_{(2)}LOG,\ \ldots,\ IMPTIX_{(n)}LOG \right\}$$

Explanation of variables

n

Number of text indexes defined for the table to be imported

IMPTIX(i)LOG

Log size of the i-th text index defined for the table to be imported

Use the following formula to determine its value. However, the formula depends on whether the `adbimport` command is executed with the `-d` option specified (whether the creation mode or the addition mode is used).

■ Formula (when the `-d` option is specified (creation mode is used)) (bytes)

$$IMPTIX_{(i)}LOG = DIRTIXPGNO_{(i)} \times page\_size_{(i)}$$

page_size(i)

Page size of the data DB area for storing the i-th text index defined for the table to be imported (bytes)

DIRTIXPGNO(i)

Number of pages in the management area for the i-th text index defined for the table to be imported

Use the following formula to determine its value.

$$DIRTIXPGNO_{(i)} = dbarea\_file\_num_{(i)} \times 6 + \left\lceil \frac{SGTXTDATA_{(i)}}{SEGBF_{(i)}} \right\rceil + \left\lceil \left\lceil \frac{SGTXTDATA_{(i)}}{\left\lfloor \frac{page\_size_{(i)} \times 125}{16} \right\rfloor} \right\rceil \times SEGBF_{(i)} \right\rceil$$

dbarea_file_num(i)

Number of DB area files in the data DB area for storing text indexes (files)

SGTXTDATA(i)

Number of text index segments to be allocated in the data import processing

SEGBF(i)

Segment blocking factor for the data DB area for storing text indexes

See the explanation of the variable *SEGBF* in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area.

■ **Formula (when the -d option is omitted (addition mode is used)) (bytes)**

$$IMPTIX_{(i)}LOG = \left\{ DIRTIXPGNO_{(i)} + OLDPGTIXDATA_{(i)} \right\} \times page\_size_{(i)}$$

DIRTIXPGNO(i)

Number of pages in the management area for the i-th text index defined for the table to be imported

See the variable $DIRTIXPGNO_{(i)}$ in *Formula (when the -d option is specified (creation mode is used))*.

OLDPGTIXDATA(i)

Number of pages for storing the text indexes that had been allocated before the data was imported

Use the following formula to determine its value.

```
OLDPGTIXDATA(i) = ↑TIP_STRSEG(i)↑ + ↑TIP_APPSEG(i)↑
```

TIP_STRSEG(i)

See (1) Determining the number of storage pages used in the string control segment (variable TIP_STRSEG(i)) in 5.8.5 Determining the number of storage pages for each text index segment.

TIP_APPSEG(i)

See (2) Determining the number of storage pages used in the position control segment (variable TIP_APPSEG(i)) in 5.8.5 Determining the number of storage pages for each text index segment.

6.12.5 Determining the size of the user logs that are output during execution of the adbidxrebuild command (variable max_user_log)

Use the following formula to determine the size of the user logs that are output during execution of the `adbidxrebuild` command.

Formula (bytes)

$$\text{Size of the user logs that are output during execution of the adbidxrebuild command} = \mathrm{MAX}(IDXPRGLOG + TBLPRGLOG^{\#}, IDXMAKELOG)$$

Explanation of variables

IDXPRGLOG

Index data deletion log for the table for which indexes are to be rebuilt

See (1) Determining the variable IDXPRGLOG.

TBLPRGLOG

Row data deletion log when background import is performed on the target table

See (2) Determining the variable TBLPRGLOG.

IDXMAKELOG

Index data creation log for the table for which indexes are to be rebuilt

See (3) Determining the variable IDXMAKELOG.

#

Obtain the *TBLPRGLOG* variable if you execute the `adbidxrebuild` command with the `--force` option specified after the `adbimport` command with the `-b` option has been interrupted.

(1) Determining the variable IDXPRGLOG

Use the following formula to determine the value of variable *IDXPRGLOG*.

Formula (bytes)

$$IDXPRGLOG = \sum_{i=1}^{idx\_num} \left\{ \left\lceil \left(dbarea\_file\_num_{(i)} \times 6 + \left\lceil \frac{SGIDX_{(i)}}{SEGBF_{(i)}} \right\rceil + \left\lceil \frac{SGIDX_{(i)}}{\left\lfloor \frac{page\_size_{(i)} \times 125}{16} \right\rfloor \times SEGBF_{(i)}} \right\rceil \times page\_size_{(i)} \right) \Big/ usrlog\_file\_num \right\rceil \right\}$$

Explanation of variables

idx_num

Number of indexes to be rebuilt

$dbarea\_file\_num_{(i)}$

Number of DB area files in data DB area that stores the indexes

$SGIDX_{(i)}$

Number of segments that store indexes

$SEGBF_{(i)}$

Number of segment block factors in the data DB area that stores indexes

$page\_size_{(i)}$

Page size of the data DB area that stores indexes (bytes)

usrlog_file_num

Use the following formula to determine its value.

usrlog_file_num =
number of user log files required at execution of the adbidxrebuild command − 1

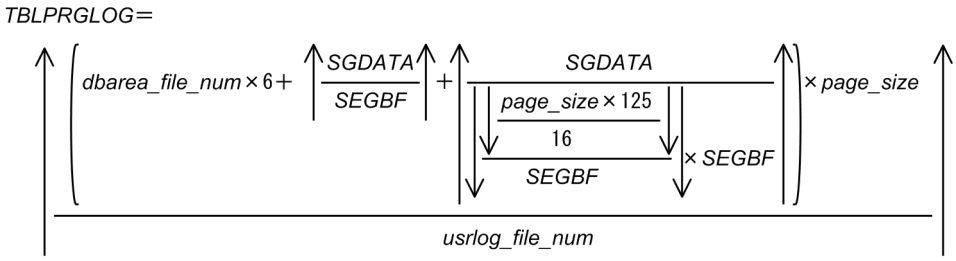
For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `adbidxrebuild` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(2) Determining the variable TBLPRGLOG

Use the following formula to determine the value of variable *TBLPRGLOG*.

Formula (bytes)



Explanation of variables

dbarea_file_num

Number of DB area files in data DB area that stores the table

SGDATA

Number of table segments to be allocated for background import

Estimate the number of table segments to be allocated for background import based on the amount of data that is stored during background import.

SEGBF

Number of segment block factors in the data DB area that stores the table

page_size

Page size of the data DB area used to store the table (bytes)

usrlog_file_num

Use the following formula to determine its value.

usrlog_file_num =
number of user log files required at execution of the adbidxrebuild command − 1

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `adbidxrebuild` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(3) Determining the variable IDXMAKELOG

Use the following formula to determine the value of variable *IDXMAKELOG*.

Formula (bytes)

Explanation of variables

n

Number of indexes to be rebuilt

IDXMAKE$_{(i)}$LOG

Log size of i-th index to be rebuilt

Use the following formula to determine its value.

Formula (bytes)

$$IDXMAKE_{(i)}LOG = \left(\frac{DIRPGNO_{(i)} \times page\_size_{(i)}}{rbld\_dividx\_rthd} \right)$$

page_size$_{(i)}$

Page size of the data DB area that stores the i-th index to be rebuilt (bytes)

rbld_dividx_rthd

Use the following formula to determine the value:

```
↓(value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1) ÷ 2↓
```

If the index rebuild option `adb_idxrebuild_rthd_num` is omitted, substitute the number of threads used when the index rebuild option `adb_idxrebuild_rthd_num` is omitted. For details, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

DIRPGNO$_{(i)}$

Number of pages in the management area for the i-th index to be rebuilt

Use the following formula to determine its value.

Formula

$$DIRPGNO_{(i)} = dbarea\_file\_num_{(i)} \times 6 + \left\uparrow \frac{SGIDX_{(i)}}{SEGBF_{(i)}} \right\uparrow + \left\uparrow \frac{SGIDX_{(i)}}{\left\downarrow \frac{\left\downarrow \frac{page\_size_{(i)} \times 125}{16} \right\downarrow}{SEGBF_{(i)}} \right\downarrow \times SEGBF_{(i)}} \right\uparrow$$

dbarea_file_num$_{(i)}$

Number of DB area files in data DB area that stores the indexes

SGIDX$_{(i)}$

Maximum number of segments that store indexes to be allocated to each chunk during index rebuild processing

Use the following formula to determine its value.

Formula

$$SGIDX_{(i)} = \mathrm{MAX}\left\{ CHUNKSGIDX_{(1)}LOG,\ CHUNKSGIDX_{(2)}LOG,\ \ldots,\ CHUNKSGIDX_{(m)}LOG \right\}$$

- *m*

  Number of chunks created by the index to be rebuilt

- *CHUNKSGIDX$_{(j)}$LOG*

  Number of segments that store indexes to be allocated to each chunk of the index to be rebuilt

SEGBF$_{(i)}$

Number of segment block factors in the data DB area that stores indexes

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

6.12.6 Determining the size of the user logs that are output during execution of the adbmergechunk command (variable max_user_log)

Use the following formula to determine the size of the user logs that are output during execution of the `adbmergechunk` command.

Formula (bytes)

Size of user logs output during execution of the adbmergechunk command =
MAX(*MERPRGLOG*, *MERMAKELOG* + *MERLOCLOG*)

Explanation of variables

MERPRGLOG
Deletion log for the index data that is defined for the table to be processed by the merge chunk command
See (1) Determining the variable MERPRGLOG.

MERMAKELOG
Creation log for the index data that is defined for the table to be processed by the merge chunk command
See (2) Determining the variable MERMAKELOG.

MERLOCLOG
Location table update log
See (3) Determining the variable MERLOCLOG.

(1) Determining the variable MERPRGLOG

Use the following formula to determine the value of variable *MERPRGLOG*.

Formula (bytes)

$$MERPRGLOG = \sum_{i=1}^{idx\_num} \left\{ \left\lceil \frac{\left(dbarea\_file\_num_{(i)} \times 6 + \left\lceil \frac{SGIDX_{(i)}}{SEGBF_{(i)}} \right\rceil + \left\lceil \frac{SGIDX_{(i)}}{\left\lfloor \frac{\left\lfloor \frac{page\_size_{(i)} \times 125}{16} \right\rfloor}{SEGBF_{(i)}} \right\rfloor \times SEGBF_{(i)}} \right\rceil \right) \times page\_size_{(i)}}{usrlog\_file\_num} \right\rceil \right\}$$

Explanation of variables

idx_num
Number of indexes defined for the table to be processed by the merge chunk command

dbarea_file_num$_{(i)}$
Number of DB area files in data DB areas that store indexes

$SGIDX_{(i)}$

Number of segments that store indexes

$SEGBF_{(i)}$

Number of segment block factors in the data DB area that stores indexes

$page\_size_{(i)}$

Page size of the data DB area that stores indexes (bytes)

usrlog_file_num

Use the following formula to determine its value.

$$usrlog\_file\_num = number\ of\ user\ log\ files\ required\ at\ execution\ of\ the\ adbmergechunk\ command - 1$$

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `adbmergechunk` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(2) Determining the variable MERMAKELOG

Use the following formula to determine the value of variable *MERMAKELOG*.

Formula (bytes)

$$MERMAKELOG = \mathrm{MAX}\left\{MERMAKE_{(1)}LOG, MERMAKE_{(2)}LOG, \ldots, MERMAKE_{(n)}LOG\right\}$$

Explanation of variables

n

Number of indexes defined for the table to be processed by the merge chunk command

$MERMAKE_{(i)}LOG$

Log size of the i-th index defined for the table to be processed by the merge chunk command

Use the following formula to determine its value.

Formula (bytes)

$$MERMAKE_{(i)}LOG = \left(\frac{DIRPGNO_{(i)} \times page\_size_{(i)}}{merc\_dividx\_rthd}\right)$$

$page\_size_{(i)}$

Page size of the data DB area that stores the i-th index defined for the table to be processed by the merge chunk command (bytes)

merc_dividx_rthd

Use the following formula to determine the value:

```
↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
```

If the merge chunk option `adb_mergechunk_rthd_num` is omitted, substitute the number of threads used when the merge chunk option `adb_mergechunk_rthd_num` is omitted. For details, see *adbmergechunk (Merge Chunks)* in the manual *HADB Command Reference*.

$DIRPGNO_{(i)}$

Number of pages in the management area for the i-th index defined for the table to be processed by the merge chunk command

Use the following formula to determine its value.

Formula

$$DIRPGNO_{(i)} = dbarea\_file\_num_{(i)} \times 6 + \left\uparrow \frac{SGIDX_{(i)}}{SEGBF_{(i)}} \right\uparrow + \left\uparrow \left\downarrow \frac{SGIDX_{(i)}}{\left\downarrow \frac{page\_size_{(i)} \times 125}{16} \right\downarrow} \right\downarrow \times SEGBF_{(i)} \right\uparrow$$

$dbarea\_file\_num_{(i)}$

Number of DB area files in data DB area that stores the indexes

$SGIDX_{(i)}$

Maximum number of index segments to be allocated to each chunk during merge chunk processing (index rebuilding).

Use the following formula to determine its value.

Formula

$$SGIDX_{(i)} = \mathrm{MAX}\left\{CHUNKSGIDX_{(1)}, CHUNKSGIDX_{(2)}, \ldots, CHUNKSGIDX_{(m)}\right\}$$

- m

  Number of chunks to be merged

- $CHUNKSGIDX_{(j)}$

  Number of index segments to be allocated to each chunk of the index to be rebuilt

$SEGBF_{(i)}$

Number of segment block factors in the data DB area that stores indexes

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

(3) Determining the variable MERLOCLOG

Use the following formula to determine the value of variable *MERLOCLOG*.

If the table to be processed is not an archivable multi-chunk table, substitute 0. Also, if the chunk to be processed is not archived, substitute 0.

Formula (bytes)

$$MERLOCLOG = UPDLOG + LBP \times \left(\uparrow \frac{84 + page\_size}{8} \uparrow \times 8 + 4 \right)$$

Explanation of variables

UPDLOG

Determine the value from the formula in *Size of user logs that are output when an UPDATE statement is executed* in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

VRWLOG

Substitute 0.

TIXSLOG

Substitute 0.

IDXSLOG

To determine the value of the variable *IDXSLOG*, see (3) Determining the size of the B-tree index user log in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *idx_num*

  Substitute 1.

- *IDXLOG(i)*

  Use the following formula to determine this value.

  Formula

  $$IDXLOG_{(i)} = (296 + SPLITLOG_{(i)}) \times LROWNUM + (\uparrow(84 + page\_size) \div 8 \uparrow \times 8 + 4) \times \uparrow LROWNUM \div 256 \uparrow$$

- *SPLITLOG(i)*

  Determine this value according to *Formula (when the number of duplicate index keys in the update-target rows is 256 or more)* in the description of the $SPLITLOG_{(i)}$ variable in (3) Determining the size of the B-tree index user log under 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log).

LROWNUM

Number of archive files created by the `adbarchivechunk` command

Use the formula for the variable *LROWNUM* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *archive_chunk_num*

  Substitute 0.

- *merge_chunk_num*

  Substitute 1.

upd_row_num

Substitute the same value as the variable *LROWNUM* determined before.

SGMTLOG

Use the following formula to determine this value.

Formula

SGMTLOG = 1,228 × ↑*LBP* ÷ *SEGSIZE*↑

LBP

Use the formula for the variable *LBP(i)* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *LROWNUM*

  Substitute the same value as the variable *LROWNUM* determined before.

SEGSIZE

Segment size of the data DB area that stores the archivable multi-chunk table

Use the following formula to determine this value.

Formula (pages)

SEGSIZE = 4,194,304 ÷ *page_size*

page_size

Page size of the data DB area that stores the archivable multi-chunk table (bytes)

6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log)

This subsection explains how to calculate the size of the user logs that are output during the execution of an `INSERT` statement (row insertion), a `DELETE` statement (row deletion), and an `UPDATE` statement (row update).

First, determine the user log sizes based on (1) Determining the base row user log size through (6) Determining the user log size for allocating segments. Then, calculate the user log size by substituting the variables described in Formula 1 to Formula 3, as follows:

Formula 1: Size of user logs that are output when an INSERT statement is executed (bytes)

Formula 1: *Size of the user logs that are output when an INSERT statement is executed* =
(*BRWLOG* + *VRWLOG* + *IDXSLOG* + *TIXSLOG*) × *ins_row_num* + *SGMTLOG*

Formula 2: Size of user logs that are output when a DELETE statement is executed (bytes)

Formula 2: *Size of the user logs that are output when a DELETE statement is executed* =
(*BRWLOG* + *VRWLOG* + *COLUMN_DELLOG* + *IDXSLOG*) × *del_row_num* + *SGMTLOG*

Formula 3: Size of user logs that are output when an UPDATE statement is executed (bytes)

Formula 3: *Size of the user logs that are output when an UPDATE statement is executed* =
(*BRWLOG* + *VRWLOG* + *COLUMN_DELLOG* + *IDXSLOG* + *TIXSLOG*) × *upd_row_num* + *SGMTLOG*

Explanation of variables

BRWLOG

Base row user log size (bytes)

Determine the value for *BRWLOG* based on the explanation in (1) Determining the base row user log size.

VRWLOG

Branch row user log size (bytes)

Determine the value for *VRWLOG* based on the explanation in (2) Determining the branch row user log size.

COLUMN_DELLOG

Log size of the invalid row information pages of column store tables (bytes)

For details about *COLUMN_DELLOG*, see (5) Determining the log size of the invalid row information pages for column store tables.

IDXSLOG

Combined sizes of all the user logs for the B-tree indexes defined for the table (bytes)

Determine the value for *IDXSLOG* based on the explanation in (3) Determining the size of the B-tree index user log.

TIXSLOG

Combined sizes of all the user logs for the text indexes defined for the table (bytes)

Determine the value for *TIXSLOG* based on the explanation in (4) Determining the size of the text index user log.

SGMTLOG

User log size for allocating segments (bytes)

Determine the value for *SGMTLOG* based on the explanation in (6) Determining the user log size for allocating segments.

ins_row_num

Number of rows inserted by the `INSERT` statement

del_row_num

Number of rows deleted by the `DELETE` statement

upd_row_num

Number of rows updated by the `UPDATE` statement

(1) Determining the base row user log size

The following table shows the base row user log size (variable *BRWLOG*).

Table 6-22: Base row user log size (variable BRWLOG)

No.	Operation description	Base row user log size (bytes)
1	`INSERT` statement	92
2	`DELETE` statement	
3	`UPDATE` statement	184

(2) Determining the branch row user log size

The following table shows the formulas for the branch row user log size (variable *VRWLOG*).

Table 6-23: Branch row user log size (variable VRWLOG)

No.	Operation description	Branch row user log size (bytes)
1	INSERT statement	$92 \times \sum_{i=1}^{var\_num} \left\uparrow \frac{var\_size_{(i)}}{page\_size - 70} \right\uparrow$
2	DELETE statement	Same as for row 1
3	UPDATE statement	$184 \times \sum_{i=1}^{var\_num\_upd} \left\uparrow \frac{var\_size_{(i)}}{page\_size - 70} \right\uparrow$

Explanation of variables

var_num

Number of columns managed as branch rows (columns)

Determine the number of columns as described in the following.

If the BRANCH ALL table option is specified

- Number of VARCHAR and VARBINARY columns

If the BRANCH ALL table option is not specified

Sum total of VARCHAR and VARBINARY columns that satisfy any of the following conditions:

- Column for which YES is specified in the BRANCH column definition
- Column whose definition length is 256 bytes or longer and for which AUTO is specified in the BRANCH column definition
- Column whose definition length is 256 bytes or longer and for which the BRANCH column definition is omitted

var_num_upd

Number of columns updated out of those columns that are managed as branch rows (columns)

var_size

Average size of the data in the columns that are managed as branch rows (bytes)

Determine the value according to Table 5-17: Data length of columns that become branch rows in (2) Determining the number of pages for branch rows (variable VP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

page_size

Page size of data DB area (bytes)

(3) Determining the size of the B-tree index user log

Use the following formula to determine the size of the B-tree index user log (variable *IDXSLOG*).

Formula

$$index\ user\ log\ size = \sum_{i=1}^{idx\_num} IDXLOG_{(i)}$$

Explanation of variables

idx_num

Number of B-tree indexes defined for the column to be updated

$IDXLOG_{(i)}$

Size of the index user log related to the i-th B-tree index defined for the table to be updated (bytes)

Determine $IDXLOG_{(i)}$ from the following table:

No.	Update type	Number of duplicate update-target keys before updating	IDXLOG(i) value
1	`INSERT` statement	0 (new key)	$AFTERLOG_{(i)} + SPLITLOG_{(i)}$
2		1 to 254	Same as for row 1
3		255	$300 + AFTERLOG_{(i)}$
4		256 or more	Same as for row 1
5	`DELETE` statement		$BEFORELOG_{(i)}$
6	`UPDATE` statement		$BEFORELOG_{(i)}$ + *size of user log that is output when the INSERT statement is executed*[#]

#

Same log size as if the updated data is being inserted into the row that is the target of the update.

$AFTERLOG_{(i)}$

For each update target row, updated logical log size for the updated data (bytes)

$BEFORELOG_{(i)}$

For each update target row, updated logical log size for the original data (bytes)

Use the following formula to determine the sizes of $AFTERLOG_{(i)}$ and $BEFORELOG_{(i)}$.

Formula

$$AFTERLOG_{(i)} \text{ and } BEFORELOG_{(i)} = \left\lceil \frac{120 + (8 \times idx\_col\_num) + 4 + KEYSZDB}{8} \right\rceil \times 8 + 4$$

idx_col_num

Number of columns indexed by the targeted index (columns)

KEYSZDB

Database storage length of the targeted index key (bytes)

The updated index key is used for $AFTERLOG_{(i)}$ and original index key is used for $BEFORELOG_{(i)}$.

$SPLITLOG_{(i)}$

User log size when an index page split occurs (bytes)

Use the following formula to determine the user log size when an index page split occurs in the target index.

Formula (when the number of duplicate index keys in the update-target rows is 254 or fewer)

$$SPLITLOG_{(i)} = 260 + \left(\uparrow \frac{84 + page\_size}{8} \uparrow \times 8 + 4 \right) \times idx\_level$$

Formula (when the number of duplicate index keys in the update-target rows is 256 or more)

$$SPLITLOG_{(i)} = 260 + \left(\uparrow \frac{84 + page\_size}{8} \uparrow \times 8 + 4 \right) \times 2$$

page_size

Page size of the data DB area in which the target B-tree index is defined (bytes)

idx_level

Number of levels in the target B-tree index (steps)

(4) Determining the size of the text index user log

Use the following formula to determine the size of the text index user log (variable *TIXSLOG*).

Formula

$$TIXSLOG = \sum_{i=1}^{tix\_num} TIXLOG_{(i)}$$

Explanation of variables

tix_num

Number of text indexes defined for the column to be updated

$TIXLOG_{(i)}$

Size of the text index user log related to the i-th text index defined for the table to be updated (bytes)

Determine $TIXLOG_{(i)}$ from the following table:

Table 6-24: TIXLOG(i) value

No.	Update type	TIXLOG(i) value
1	`INSERT` statement	$SPLITLOG_{(i)}$
2	`UPDATE` statement	$SPLITLOG_{(i)}$

$SPLITLOG_{(i)}$

Log size in the event of a text index page split

Use the following formula to determine its value.

Formula (bytes)

$$SPLITLOG_{(i)} = 260 + \left(\uparrow \frac{84 + page\_size}{8} \uparrow \times 8 + 4 \right) \times 2$$

page_size

Page size of the data DB area in which the target text index is defined (bytes)

(5) Determining the log size of the invalid row information pages for column store tables

Use the following formula to determine the value of variable *COLUMN_DELLOG*.

Note that if the processing-target table is a row store table, assume that this value is 0 during estimation.

Formula (bytes)

```
COLUMN_DELLOG = 332 + 2 × page_size
```

Explanation of the variables

page_size
Page size of the DB area in which the column store table is defined (bytes)

(6) Determining the user log size for allocating segments

If a row or index entry cannot be stored in the active page when an `INSERT` or `UPDATE` statement is executed to add a row or index entry, it is allocated to an unused page. Also, when rows that have been stored in column store format are invalidated by executing the `UPDATE` or `DELETE` statement, the information indicating that the rows became invalid sometimes cannot be stored on the invalid row information page. In such a case, the information is allocated to an unused page.

If all pages within a segment are being used, a new segment is allocated.

The table below shows the user log size for allocating new segments (variable *SGMTLOG*).

Based on the row data to be added by the `INSERT` statement and the row data to be updated by the `UPDATE` statement, determine the number of segments to be allocated as described in 5.8.1 Determining the total number of pages in the data DB area, 5.8.2 Determining the number of pages for storing each type of row, 5.8.3 Determining the number of storage pages for each B-tree index segment, and 5.8.5 Determining the number of storage pages for each text index segment.

If the processing-target table is a column store table, take into consideration the number of rows to be deleted by using the `DELETE` statement during estimation.

Table 6-25: User log size for allocating new segments

No.	Operation description	User log size (bytes)
1	`INSERT` statement	1,228 × *number of allocated segments*
2	`UPDATE` statement	
3	`DELETE` statement#	

#
Note that if the deletion-target table is a row store table, assume that this value is 0 during estimation.

6.12.8 Determining the size of the user logs that are output during execution of the TRUNCATE TABLE statement (variable max_user_log)

Use the following formula to determine the size of the user logs that are output during execution of the `TRUNCATE TABLE` statement.

Formula (bytes)

Size of user logs output during execution of the TRUNCATE TABLE statement =

$$263{,}712 + \left\lceil \frac{\left(dbarea\_file\_num \times 5 + \left\lceil \frac{SGDATA}{SEGBF} \right\rceil + \left\lceil \frac{SGDATA}{\left\lfloor \frac{\left\lfloor \frac{page\_size \times 125}{16} \right\rfloor}{SEGBF} \right\rfloor \times SEGBF} \right\rceil \times page\_size \right)}{usrlog\_file\_num} \right\rceil$$

$$+ TRUNCLOCLOG$$

Explanation of variables

dbarea_file_num

Number of DB area files in data DB area that stores the base table to be processed

SGDATA

Number of data DB area segments that store the base table to be processed

SEGBF

Number of segment block factors in the data DB area that stores the base table to be processed

page_size

Page size of the data DB area that stores the base table to be processed (bytes)

usrlog_file_num

Use the following formula to determine its value.

usrlog_file_num =
number of user log files required at execution of the TRUNCATE TABLE statement − 1

TRUNCLOCLOG

Location table update log

Use the following formula to determine this value.

If the table to be processed is not an archivable multi-chunk table, substitute 0. Also, if no archived chunks exist in the table to be processed, substitute 0.

Formula (bytes)

$$TRUNCLOCLOG = DELLOG + LBP \times \left(\left\lceil \frac{84 + page\_size}{8} \right\rceil \times 8 + 4 \right)$$

DELLOG

Determine the value from the formula in *Size of user logs that are output when a DELETE statement is executed* in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note

that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

VRWLOG

Substitute 0.

IDXSLOG

To determine the value of the variable *IDXSLOG*, see (3) Determining the size of the B-tree index user log in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *idx_num*

  Substitute 1.

- *IDXLOG(i)*

  Use the following formula to determine this value.

  Formula

  $$IDXLOG_{(i)} = 148 \times LROWNUM + (\uparrow(84 + page\_size) \div 8\uparrow \times 8 + 4) \times \uparrow LROWNUM \div 256\uparrow$$

LROWNUM

Number of archive files created by the adbarchivechunk command

Use the formula for the variable *LROWNUM* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *archive_chunk_num*

  Substitute 1.

- *merge_chunk_num*

  Substitute 0.

LBP

Use the formula for the variable *LBP(i)* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *LROWNUM*

  Substitute the same value as the variable *LROWNUM* determined before.

page_size

Page size of the data DB area that stores the archivable multi-chunk table (bytes)

del_row_num

Substitute the same value as the variable *LROWNUM* determined before.

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `TRUNCATE TABLE` statement, see (1) SQL statements for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

If indexes are defined for the base table on which the TRUNCATE TABLE statement is to be executed, use the formula below to determine the value for each index. Then, add the determined values to the result of the above formula.

Formula (when an index is defined for the base table) (bytes)

Size of user logs when the TRUNCATE TABLE statement is used to delete indexes =

$$263{,}712 + \left\lceil \frac{\left(dbarea\_file\_num \times 5 + \left\lceil \frac{SGIDX}{SEGBF} \right\rceil + \left\lceil \left\lceil \frac{page\_size \times 125}{16} \right\rceil \times \frac{SGIDX}{SEGBF} \right\rceil \right) \times page\_size}{usrlog\_file\_num} \right\rceil$$

Explanation of variables

dbarea_file_num
Number of DB area files in data DB area that stores the indexes to be processed

SGIDX
Number of segments in the data DB area that stores the indexes to be processed

SEGBF
Number of segment block factors in the data DB area that stores the indexes to be processed

page_size
Page size of the data DB area that stores the indexes to be processed (bytes)

usrlog_file_num
Use the following formula to determine its value.

usrlog_file_num =
number of user log files required at execution of the TRUNCATE TABLE statement − 1

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the TRUNCATE TABLE statement, see (1) SQL statements for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

6.12.9 Determining the size of the user logs that are output during execution of the PURGE CHUNK statement (variable max_user_log)

Use the following formula to determine the size of the user logs that are output during execution of the PURGE CHUNK statement.

Formula (bytes)

$$\text{Size of user logs output during execution of the PURGE CHUNK statement} = \sum_{j=1}^{chunk\_num} \left(TBLCHPRGLOG(j) + IDXCHPRGLOG(j) + STBLUPDLOG + PRGLOCLOG \right)$$

Explanation of the variables

chunk_num

Number of chunks to be deleted by one PURGE CHUNK statement

TBLCHPRGLOG$_{(j)}$

Deletion log for the row data held by the chunks in a base table

Use the following formula to determine the value for each chunk to be deleted.

Formula (bytes)

$$TBLCHPRGLOG = \left\lceil \left\{ dbarea\_file\_num \times 6 + \left\lceil \frac{SGCHTBL}{SEGBF} \right\rceil + \left\lceil \frac{SGCHTBL}{\left\lceil \frac{\left\lceil \frac{page\_size \times 125}{16} \right\rceil}{SEGBF} \right\rceil \times SEGBF} \right\rceil \right\} \times page\_size \right\rceil \Big/ usrlog\_file\_num$$

dbarea_file_num

Number of DB area files in data DB area that stores the base table

SGCHTBL

Number of segments allocated by the chunks of the base table

SEGBF

Number of segment blocking factors in the data DB area that stores the base table

page_size

Page size of the data DB area that stores the base table (bytes)

usrlog_file_num

Use the following formula to determine its value.

$$usrlog\_file\_num = \text{number of user log files required at execution of the PURGE CHUNK statement} - 1$$

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the PURGE CHUNK statement, see (1) SQL statements for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

IDXCHPRGLOG$_{(j)}$

Deletion log for the index data in each chunk of the table to be processed

Use the following formula to determine the value for each chunk to be deleted.

Formula (bytes)

$$IDXCHPRGLOG = \sum_{i=1}^{idx\_num} \left\{ \left\lceil \frac{\left(dbarea\_file\_num_{(i)} \times 6 + \left\lceil \frac{SGCHIDX_{(i)}}{SEGBF_{(i)}} \right\rceil + \left\lceil \frac{SGCHIDX_{(i)}}{\left\lfloor \frac{page\_size_{(i)} \times 125}{16} \right\rfloor \times SEGBF_{(i)}} \right\rceil \times page\_size_{(i)} \right)}{usrlog\_file\_num} \right\rceil \right\}$$

idx_num

Number of indexes defined for the base table to be deleted

dbarea_file_num$_{(i)}$

Number of DB area files in data DB area that stores the indexes

SGCHIDX$_{(i)}$

Number of segments allocated by the chunks of indexes

SEGBF$_{(i)}$

Number of segment blocking factors in the data DB area of each index

page_size$_{(i)}$

Page size in the data DB area of each index (bytes)

usrlog_file_num

Use the following formula to determine its value.

usrlog_file_num =
number of user log files required at execution of the PURGE CHUNK statement − 1

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `PURGE CHUNK` statement, see (1) SQL statements for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

STBLUPDLOG

Update log created when data is deleted from system tables (base tables)

Substitute 480 bytes in the variable *STBLUPDLOG*.

PRGLOCLOG

Location table update log

Use the following formula to determine this value.

If the table to be processed is not an archivable multi-chunk table, substitute 0. Also, if the chunk to be processed is not archived, substitute 0.

Formula (bytes)

$$PRGLOCLOG = DELLOG + LBP \times \left(\left\lceil \frac{84 + page\_size}{8} \right\rceil \times 8 + 4 \right)$$

DELLOG

Determine the value from the formula in *Size of user logs that are output when a DELETE statement is executed* in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note

that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

VRWLOG

Substitute 0.

IDXSLOG

To determine the value of the variable *IDXSLOG*, see (3) Determining the size of the B-tree index user log in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *idx_num*

  Substitute 1.

- *IDXLOG(i)*

  Use the following formula to determine this value.

  Formula

  $$IDXLOG_{(i)} = 148 \times LROWNUM + (\uparrow(84 + page\_size) \div 8\uparrow \times 8 + 4) \times \uparrow LROWNUM \div 256\uparrow$$

LROWNUM

Number of archive files created by the `adbarchivechunk` command

Use the formula for the variable *LROWNUM* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *archive_chunk_num*

  Substitute 1.

- *merge_chunk_num*

  Substitute 0.

LBP

Use the formula for the variable *LBP(i)* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *LROWNUM*

  Substitute the same value as the variable *LROWNUM* determined before.

page_size

Page size of the data DB area that stores the archivable multi-chunk table (bytes)

del_row_num

Substitute the same value as the variable *LROWNUM* determined before.

6.12.10 Determining the size of the user logs that are output during execution of the adbchgchunkstatus command (variable max_user_log)

Use the following formula to determine the size of the user logs that are output during execution of the `adbchgchunkstatus` command.

Formula (bytes)

```
Size of user logs output during execution of adbchgchunkstatus command =
          TBLCHKSTATLOG + IDXCHKSTATLOG + 5,760,360 × (1 + idx_num)
```

Explanation of the variables

idx_num

Number of indexes defined for the base table in the chunk whose status is to be changed

TBLCHKSTATLOG

Size of logs for a base table in a chunk whose status is to be changed.

Use the following formula to determine its value.

Formula (bytes)

$$TBLCHKSTATLOG = dbarea\_file\_num \times \left\{ 2 + MIN \left(\left\lceil \frac{32{,}000}{\left\lfloor \frac{page\_size}{160} \right\rfloor} \right\rceil , change\_chunk\_num \right) \right\} \times page\_size$$

- *dbarea_file_num*
  Number of DB area files in data DB area that stores the base table
- *change_chunk_num*
  Number of chunks whose status is to be changed
- *page_size*
  Page size of the data DB area that stores the base table (bytes)

IDXCHKSTATLOG

Log size for the indexes defined for a base table in a chunk whose status is to be changed.

Use the following formula to determine its value.

Formula (bytes)

$$IDXCHKSTATLOG = \sum_{i=1}^{idx\_num} \left\{ dbarea\_file\_num_{(i)} \times \left(2 + MIN \left(\left\lceil \frac{32{,}000}{\left\lfloor \frac{page\_size_{(i)}}{160} \right\rfloor} \right\rceil , change\_chunk\_num \right) \right) \times page\_size_{(i)} \right\}$$

- *idx_num*
  Number of indexes defined for the base table in the chunk whose status is to be changed
- $dbarea\_file\_num_{(i)}$
  Number of DB area files in the data DB areas that store the indexes

- *change_chunk_num*
  Number of chunks whose status is to be changed
- $page\_size_{(i)}$
  Page size in the data DB area of each index (bytes)

6.12.11 Determining the size of the user logs that are output during execution of the adbarchivechunk command (variable max_user_log)

Use the following formula to determine the size of the user logs that are output during execution of the `adbarchivechunk` command.

Formula (bytes)

Size of the user logs that are output during execution of the adbarchivechunk command =
TBLARCCHKLOG + *IDXARCCHKLOG* + *STBLUPDLOG* + 1,920,120 × (1 + *idx_num*) + *ARCCHKLOCLOG*

Note

If you execute the `adbarchivechunk` command for multiple chunks, determine the value based on the chunk that uses most segments among all chunks.

Explanation of variables

TBLARCCHKLOG

Log data size for the archivable multi-chunk table that is output when the `adbarchivechunk` command is executed

See (1) Determining the variable TBLARCCHKLOG.

IDXARCCHKLOG

Log data size for the indexes that are defined for the archivable multi-chunk table that is output when the `adbarchivechunk` command is executed

See (2) Determining the variable IDXARCCHKLOG.

STBLUPDLOG

Size of log data that is output when the data of the system tables (base tables) is updated when the `adbarchivechunk` command is executed

See (3) Determining the variable STBLUPDLOG.

idx_num

Number of indexes that are defined for the archivable multi-chunk table for which the `adbarchivechunk` command is executed

ARCCHKLOCLOG

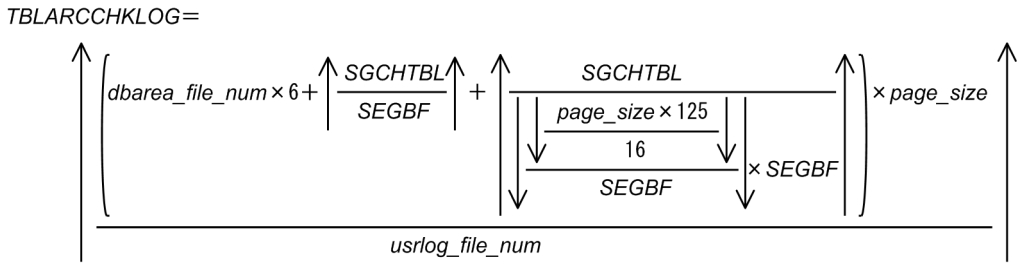
Location table update log

See (4) Determining the variable ARCCHKLOCLOG.

(1) Determining the variable TBLARCCHKLOG

Use the following formula to determine the value of variable *TBLARCCHKLOG*.

Formula (bytes)



Explanation of variables

dbarea_file_num

Number of data DB area files for the data DB area that stores the archivable multi-chunk table

SGCHTBL

Number of segments that are used by the archivable multi-chunk table in the chunk to be processed by the `adbarchivechunk` command

SEGBF

Segment block factor of the data DB area that stores the archivable multi-chunk table

page_size

Page size of the data DB area that stores the archivable multi-chunk table (bytes)

usrlog_file_num

Use the following formula to determine this value.

usrlog_file_num =
number of user log files required to execute the adbarchivechunk command - 1

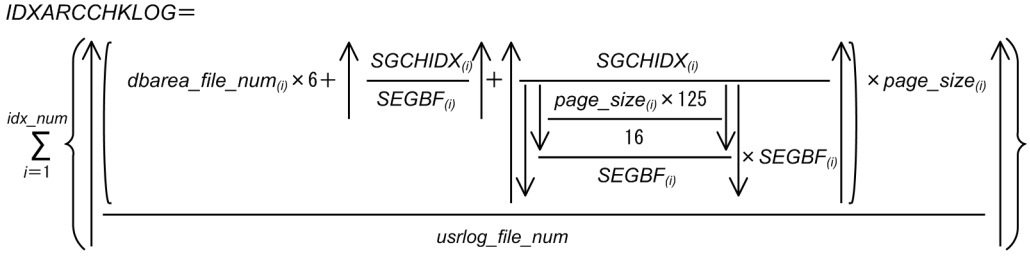
For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `adbarchivechunk` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(2) Determining the variable IDXARCCHKLOG

Use the following formula to determine the value of variable *IDXARCCHKLOG*.

Formula (bytes)



Explanation of variables

idx_num

Number of indexes that are defined for the archivable multi-chunk table for which the `adbarchivechunk` command is executed

dbarea_file_num(i)

Number of data DB area files for the data DB area that contains the indexes

SGCHIDX(i)

Number of segments used by the indexes defined for the archivable multi-chunk table in the chunks to be processed by the `adbarchivechunk` command

SEGBF(i)

Segment block factor of the data DB area that stores the indexes defined for the archivable multi-chunk table

page_size(i)

Page size of the data DB area that stores the indexes defined for the archivable multi-chunk table (bytes)

usrlog_file_num

Use the following formula to determine this value.

usrlog_file_num =
number of user log files required to execute the adbarchivechunk command - 1

For details about the number of segments in a data DB area and the segment block factors, see (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

For details about the number of user log files required during execution of the `adbarchivechunk` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

(3) Determining the variable STBLUPDLOG

Substitute the following value for the variable *STBLUPDLOG*.

Value (bytes)

$$STBLUPDLOG = 968$$

(4) Determining the variable ARCCHKLOCLOG

Use the following formula to determine the value of variable *ARCCHKLOCLOG*.

Formula (bytes)

$$ARCCHKLOCLOG = INSLOG + LBP \times \left(\left\uparrow \frac{84 + page\_size}{8} \right\uparrow \times 8 + 4 \right)$$

Explanation of variables

INSLOG

Determine the value from the formula in *Size of user logs that are output when an INSERT statement is executed* in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note

that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

VRWLOG

Substitute 0.

TIXSLOG

Substitute 0.

IDXSLOG

To determine the value of the variable *IDXSLOG*, see (3) Determining the size of the B-tree index user log in 6.12.7 Determining the size of the user logs that are output during database updating (variable max_user_log). Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *idx_num*

  Substitute 1.

- *IDXLOG(i)*

  Use the following formula to determine this value.

  Formula

  $$IDXLOG_{(i)} = (148 + SPLITLOG_{(i)}) \times LROWNUM + (\uparrow(84 + page\_size) \div 8 \uparrow \times 8 + 4) \times \uparrow LROWNUM \div 256 \uparrow$$

- *SPLITLOG(i)*

  Determine this value according to *Formula (when the number of duplicate index keys in the update-target rows is 256 or more)* in the description of the $SPLITLOG_{(i)}$ variable in (3) Determine the size of the B-tree index user log under 6.12.7 Determine the size of the user logs that are output during database updating (variable max_user_log).

LROWNUM

Number of archive files created by the `adbarchivechunk` command

Use the formula for the variable *LROWNUM* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *archive_chunk_num*

  Substitute 1.

- *merge_chunk_num*

  Substitute 0.

ins_row_num

Substitute the same value as the variable *LROWNUM* determined before.

SGMTLOG

Use the following formula to determine this value.

Formula

$$SGMTLOG = 1{,}228 \times \uparrow LBP \div SEGSIZE \uparrow$$

LBP

Use the formula for the variable *LBP(i)* in (f) Determining the variable SGROWTBL (for a multi-chunk table) under (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area. Note

that for some of the variables in the formula, you must substitute values that are different from the values indicated in the preceding section. The following shows the relevant variables and the values to be substituted.

- *LROWNUM*

  Substitute the same value as the variable *LROWNUM* determined before.

SEGSIZE

Segment size of the data DB area that stores the archivable multi-chunk table

Use the following formula to determine this value.

Formula (pages)

SEGSIZE＝ 4,194,304÷*page_size*

page_size

Page size of the data DB area that stores the archivable multi-chunk table (bytes)

6.12.12 Determining the size of the user logs that are output during execution of the adbunarchivechunk command (variable max_user_log)

The size of the user log data that is output when the `adbunarchivechunk` command is executed is the same as the size of user log data that is output in the following case:

- When the `adbimport` command is executed with the `-b` option specified and the `-d` option omitted for an archivable multi-chunk table

  The data in the chunk to be unarchived is used for input.

Determine the size of the user logs that are output during the execution of the `adbimport` command according to the explanation in 6.12.4 Determining the size of the user logs that are output during execution of the adbimport command (variable max_user_log).

Note that if you execute the `adbunarchivechunk` command for multiple chunks, determine the value based on the chunk that uses most segments among all chunks.

Also, if you determine the number of segments to be used, execute the `adbdbstatus` command with the `-c archivechunk` option specified to output the summary information about archived chunks. You can check the number of segments from the following output items:

- *Unarchive_table_size*

  Size of segments that were used to store a table in the chunk before being archived (bytes)
- *Unarchive_index_size*

  Size of segments that were used to store indexes in the chunk before being archived (bytes)

6.12.13 Determining the size of the user logs that are output during execution of the adbreorgsystemdata command (variable max_user_log)

The size of the user log data that is output when the `adbreorgsystemdata` command is executed is the same as the size of user log data that is output in the following case:

- When the `adbimport` command is executed with the `-d` option specified for a multi-chunk table

Determine the size of the user logs that are output during the execution of the `adbimport` command according to the explanation in 6.12.4 Determining the size of the user logs that are output during execution of the adbimport command (variable max_user_log). When determining the size of user log data (variable *max_user_log*), substitute the following values for the variables:

IMPSTBLLOG

Substitute 0.

IMPRNGIDXLOG

Substitute 0.

IMPTIXLOG

Substitute 0.

dbarea_file_num

Substitute 1.

dbarea_file_num(i)

Substitute 1.

SGTBL

Execute the `adbdbstatus` command for the reorganization-target system table (base table) to output the summary information about tables. Then, substitute the value of the output item `Used_segments`.

The following shows an example of executing the `adbdbstatus` command when the table `STATUS_CHUNKS` is to be reorganized:

Example:

```
adbdbstatus -d summary -c table -n HADB.STATUS_CHUNKS
```

SEGBF

Substitute 56.

SEGBF(i)

Substitute 56.

page_size

Substitute 4,096.

page_size(i)

Substitute 4,096.

pctfree

Substitute 0.

The variable *pctfree* is used to determine the values of the following variables, which are used to solve the formula in 6.12.4 Determining the size of the user logs that are output during execution of the adbimport command (variable max_user_log):

- Variable *IP_LOWER(i)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment
- Variable *IP_UPPER(i)* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment

usrlog_file_num

Substitute the following value.

```
number-of-user-log-files-required-during-execution-of-the-adbreorgsystemdata-command - 1
```

For details about the number of user log files required during execution of the `adbreorgsystemdata` command, see (2) Commands for which the number of user log files needs to be estimated in 6.12.15 Determining the number of user log files.

idx_num

Substitute the number of indexed that are defined for the system table (base table) to be reorganized.

For details about the number of indexes defined for the system table (base table) to be reorganized, see C.8 B-tree indexes of system tables (base tables).

SGIDX(i)

Execute the `adbdbstatus` command for the indexes that are defined for the reorganization-target system table (base table) to output the summary information about indexes. Then, substitute the value of the output item `Used_segments`.

Note that you must execute the `adbdbstatus` command for each of the defined indexes.

For details about the indexes defined for the system table (base table) to be reorganized, see C.8 B-tree indexes of system tables (base tables).

The following shows an example of executing the `adbdbstatus` command when an index (`STATUSINDEXM05`) of the table `STATUS_CHUNKS` is to be reorganized.

Example:

```
adbdbstatus -d summary -c index -n HADB.STATUSINDEXM05
```

SEGDATA

Substitute the value of *SGTBL*.

imp_load_rthd

Substitute 1.

imp_dividx_rthd

Substitute 1.

6.12.14 Determining the size of user logs that are required for the maintenance processing of the updated-row columnizing facility (variable max_user_log)

Before you enable the updated-row columnizing facility, determine the size of user logs that are required for the maintenance processing of the updated-row columnizing facility.

The maintenance processing of the updated-row columnizing facility is performed for each chunk of a column store table. Therefore, first, determine the size of user logs (*COLUMNIZELOG*) for each chunk of the column store table to which the updated-row columnizing facility is applied. Then, use the largest determined size as the size of user logs that

are required for the maintenance processing of the updated-row columnizing facility. Use the following formula to determine the size of user logs (*COLUMNIZELOG*) for each chunk.

Formula (bytes)

```
COLUMNIZELOG = 1,428
               + (rowdata_sgmt_num × 112 + var_sgmt_num × 1,428
                   + dbarea_file_num × SEGSIZE × 144) × COMPRESSION_RATE
               + dbarea_file_num × 2,080
```

Explanation of variables

rowdata_sgmt_num

Number of row-data segments for which the maintenance processing of the updated-row columnizing facility is performed

Determine the number of row-data segments in the chunk by referring to the explanation of the variable *UPDATESEGNUM(i)* in (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table) in (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

var_sgmt_num

Number of segments used to store the branch rows that are created

Determine the value by referring to the following variable explanation on the assumption that the data added or updated by the `INSERT` or `UPDATE` statement in the chunk is imported by using the `adbimport` command:

- Variable *ROWDATASEGNUM(i,k)* in (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table) in (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area

dbarea_file_num

Number of DB area files in the DB area in which the column store table is to be stored

SEGSIZE

Segment size (pages)

Use the following formula to determine this value:

```
SEGSIZE = 4,194,304 ÷ page_size
```

page_size

Page size of data DB area (bytes)

COMPRESSION_RATE

Compression rate of source data

Determine the value by referring to the explanation of the variable *COMPRESSION_RATE* in (g) Determining the variable SGCOLUMNTBL (for a multi-chunk table) in (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

If the compression rate is unknown, assume 0.5.

6.12.15 Determining the number of user log files

When specific SQL statements and commands are executed, as many user log files as needed are allocated. As the SQL statements and commands terminate, the user log files are released. Therefore, when you estimate the number of user log files (the variable *user_file_num* in 6.12 Estimating the size of the system log files), take into account the SQL statements and commands that will be executed concurrently.

If you will be executing multiple SQL statements and commands, use the follow the procedure to determine the number of user log files.

Procedure

1. Determine the number of user log files for all SQL statements and commands that will be executed concurrently.
2. Sum up the numbers of user log files that were obtained.
3. Specify in the `adb_log_usrfile_num` operand in the server definition a value that is equal to or greater than the obtained sum.

For details about the `adb_log_usrfile_num` operand in the server definition, see the description of the adb_log_usrfile_num operand in 7.2.3 Operands related to system logs (set format).

(1) SQL statements for which the number of user log files needs to be estimated

You need to estimate the number of user log files for the following SQL statements:

■ Definition SQL

- `ALTER TABLE` statement (when executed to change an archivable multi-chunk table to a regular multi-chunk table)
- `DROP INDEX` statement
- `DROP TABLE` statement
- `DROP SCHEMA` statement
- `DROP USER` statement
- `SCHEMA` of the `REVOKE` statement (revoking schema operation privileges)
- Other definition SQL statements

■ Data Manipulation SQL

- `PURGE CHUNK` statement
- `TRUNCATE TABLE` statement
- `DELETE` statement
- `INSERT` statement
- `UPDATE` statement

The following formuals explain how to determine the number of user log files for each SQL statement. If the resulting value is 0, use 1.

■ Definition SQL statements

- **Formula for determining the number of user log files that will be needed to execute the `ALTER TABLE` statement (when executed to change an archivable multi-chunk table to a regular multi-chunk table) (count)**

Number of user log files (during execution of the ALTER TABLE statement#) = MIN(*A*, *B*) + 1

A: Value of the `adb_sql_exe_max_rthd_num` server definition operand
B: Maximum number of data DB area files for the data DB area that stores the location tables to be deleted and stores the indexes for those location tables

#: When the `ALTER TABLE` statement is used to change an archivable multi-chunk table to a regular multi-chunk table

- **Formula for determining the number of user log files needed to execute the `DROP INDEX` statement (count)**

Number of user log files (when the `DROP INDEX` statement is executed) = MIN(*A*, *B*) + 1

A: Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition
B: Number of data DB area files in the data DB area that stores the indexes to be processed

- **Formula for determining the number of user log files needed to execute the `DROP TABLE` statement (count)**

Number of user log files (when the `DROP TABLE` statement is executed) = MIN(*A*, *B*) + 1

A: Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition
B: Maximum value for the number of data DB area files in the data DB area that stores the tables and indexes to be processed

- **Formula for determining the number of user log files needed to execute the `DROP SCHEMA` statement (count)**

See *Formula for determining the number of user log files needed to execute the DROP TABLE statement* and determine the value for all tables that are included in the target schema. Then, use the largest such value.

- **Formula for determining the number of user log files needed to execute the `DROP USER` statement (count)**

See *Formula for determining the number of user log files needed to execute the DROP SCHEMA statement* and determine the value for the schemas owned by the target user.

- **Formula for determining the number of user log files that will be needed to execute `SCHEMA` of the `REVOKE` statement (count)**

See *Formula for determining the number of user log files needed to execute the DROP SCHEMA statement* to determine the value.

- **Formula for determining the number of user log files that will be needed to execute other definition SQL statements (count)**

The value is 1.

■ Data manipulation SQL statements

- **Formula for determining the number of user log files needed to execute the `PURGE CHUNK` statement (count)**

Number of user log files (when the `PURGE CHUNK` statement is executed) = MIN(*A*, *B*) + 1

A: Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition
B: Maximum value for the number of data DB area files in the data DB area that stores the tables and indexes to be processed

- **Formula for determining the number of user log files needed to execute the `TRUNCATE TABLE` statement (count)**

Number of user log files (when the `TRUNCATE TABLE` statement is executed) = MIN(*A*, *B*) + 1

A: Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition
B: Maximum value for the number data DB area files in the data DB area that stores the tables and indexes to be processed

- **Formula for determining the number of user log files that will be needed to execute the `DELETE` statement (count)**
  The value is 1.
- **Formula for determining the number of user log files that will be needed to execute the `INSERT` statement (count)**
  The value is 1.
- **Formula for determining the number of user log files that will be needed to execute the `UPDATE` statement (count)**
  The value is 1.

Important

If you have specified the `adb_sql_exe_max_rthd_num` operand in the client definition, assign that same value in the client definition, not the value of the `adb_sql_exe_max_rthd_num` operand in the server definition.

However, if the value of the `adb_sql_exe_max_rthd_num` operand in the client definition is greater than the value of the `adb_sql_exe_max_rthd_num` operand in the server definition, assign the value of the `adb_sql_exe_max_rthd_num` operand in the server definition.

For details about the `adb_sql_exe_max_rthd_num` operand in the server definition, see the description of the adb_sql_exe_max_rthd_num operand in 7.2.2 Operands related to performance (set format).

(2) Commands for which the number of user log files needs to be estimated

You need to estimate the number of user log files for the following commands:

- `adbimport` command
- `adbidxrebuild` command
- `adbmodarea` command
- `adbmergechunk` command
- `adbgetcst` command
- `adbchgchunkcomment` command
- `adbchgchunkstatus` command
- `adbarchivechunk` command
- `adbunarchivechunk` command
- `adbreorgsystemdata` command
- `adbsyndict` command

The following formulas explain how to determine the number of user log files for each command.

- **Formula for determining the number of user log files that will be needed to execute the `adbimport` command (count)**

Number of user log files (when executing `adbimport` command) =
value-specified-for-`adb_import_rthd_num`*-import-option* + 1

For details about the import option `adb_import_rthd_num`, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

- **Formula for determining the number of user log files that will be needed to execute the `adbidxrebuild` command (count)**

  Number of user log files (when executing `adbidxrebuild` command) =
  value-specified-for-`adb_idxrebuild_rthd_num`*-index-rebuild-option* + 1

  For details about the index rebuild option `adb_idxrebuild_rthd_num`, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

- **Formula for determining the number of user log files that will be needed to execute the `adbmodarea` command (count)**

  The value is 2.

- **Formula for determining the number of user log files that will be needed to execute the `adbmergechunk` command (count)**

  Number of user log files (when executing `adbmergechunk` command) =
  value-specified-for-`adb_mergechunk_rthd_num`*-merge-chunk-option* + 1

  For details about the merge chunk option `adb_mergechunk_rthd_num`, see *adbmergechunk (Merge Chunks)* in the manual *HADB Command Reference*.

- **Formula for determining the number of user log files that will be needed to execute the `adbgetcst` command (count)**

  The value is 1.

- **Formula for determining the number of user log files that will be needed to execute the `adbchgchunkcomment` command (count)**

  The value is 1.

- **Formula for determining the number of user log files that will be needed to execute the `adbchgchunkstatus` command (count)**

  The value is 1.

- **Formula for determining the number of user log files that will be needed to execute the `adbarchivechunk` command (count)**

  Number of user log files (when executing `adbarchivechunk` command) =
  (*value-specified-for-*`adb_arcv_rthd_num`*-archive-chunk-option* + 1)
  × *Number of chunks to be processed*

  For details about the archive chunk option `adb_arcv_rthd_num`, see *adbarchivechunk (Archive Chunk)* in the manual *HADB Command Reference*.

- **Formula for determining the number of user log files that will be needed to execute the `adbunarchivechunk` command (count)**

  Number of user log files (when executing `adbunarchivechunk` command) =
  (*value-specified-for-*`adb_unarcv_rthd_num`*-unarchive-chunk-option* + 1)
  × 11 × *Number of chunks to be processed*

For details about the unarchive chunk option `adb_unarcv_rthd_num`, see *adbunarchivechunk (Unarchive Chunk)* in the manual *HADB Command Reference*.

- **Formula for determining the number of user log files that will be needed to execute the `adbreorgsystemdata` command (count)**

  The value is 5.

- **Formula for determining the number of user log files that will be needed to execute the `adbsyndict` command (count)**

  The value is 1.

6.13 Estimating the size of the archive directory

This section describes how to determine the size of the archive directory. You must determine the size of the archive directory before you define archivable multi-chunk tables.

Use the following formula to determine the size of the archive directory (*ARCVDIRSIZE*).

Formula (gigabytes)

$$ARCVDIRSIZE = \sum_{i=1}^{tbl\_num\_in\_arcvdir} (ALLARCVCHKSIZE_{(i)})$$

Explanation of variables

tbl_num_in_arcvdir

Number of archivable multi-chunk tables that use the target archive directory

Substitute for this variable the number of archivable multi-chunk tables that are defined by specifying the target archive directory for `ARCHIVEDIR` in the chunk-archive specification when the `CREATE TABLE` statement is executed.

ALLARCVCHKSIZE(i)

Total size of all archived chunks in the i-th archivable multi-chunk table

Use the following formula to determine the variable *ALLARCVCHKSIZE* (total size of all archived chunks in the archivable multi-chunk table).

Formula (gigabytes)

$$ALLARCVCHKSIZE = \sum_{i=1}^{arcvchk\_num\_in\_tbl} (ARCVCHKSIZE_{(i)})$$

arcvchk_num_in_tbl

Total number of archived chunks in the archivable multi-chunk table

ARCVCHKSIZE(i)

Size of the i-th archived chunk in the archivable multi-chunk table

Use the following formula to determine the variable *ARCVCHKSIZE* (size of archived chunks).

Formula (gigabytes)

$$ARCVCHKSIZE = chunk\_data \times compression\_rate$$

chunk_data

Size of input data to be stored in a chunk (gigabytes)

compression_rate

Compression rate of input data to be stored in a chunk

To determine the variable *compression_rate*, perform the following procedure by using part of input data.

Procedure:

1. Define an archivable multi-chunk table.

   For details about how to define an archivable multi-chunk table, see *CREATE TABLE (define a table)* in *Definition SQL* in the manual *HADB SQL Reference*.

2. Execute the `adbimport` command with the `-b` option specified to store part of input data in the archivable multi-chunk table.

   For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

3. Use the `adbdbstatus` command to check the size of the chunk that was created in step 2.

   Execute the `adbdbstatus` command with the `-d used` and `-c table` options specified to output the information about the usage of DB areas, tables, and indexes. Determine the size of the chunk that was created in step 2 by using the following formula based on the values in the output information.

   Formula

   Chunk size = *Used_segments* x *Segment_size* x *Page_size*

   For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

4. Use the `adbarchivechunk` command to archive the chunk that was created in step 2.

   For details about the `adbarchivechunk` command, see *adbarchivechunk (Archive chunk)* in the manual *HADB Command Reference*.

5. Use the `adbdbstatus` command to check the size of the chunk that was archived in step 4.

   Execute the `adbdbstatus` command with the `-d used` option and `-c archivechunk` options specified to output information about the usage of archived chunks. Check the size of the archived chunk from the value of the output item `Archive_file_size`.

6. Determine the compression rate from the chunk sizes that you checked in steps 3 and 5.

   Use the following formula to determine the value.

   Formula

   compression_rate＝ *after_comp_somedata*÷*before_comp_somedata*

after_comp_somedata

Size of the chunk that you checked in step 5 (bytes)

This is the size of the chunk that was archived by using the `adbarchivechunk` command. The input data has been compressed because the chunk has been archived.

before_comp_somedata

Size of the chunk that you checked in step 3 (bytes)

This is the size of the chunk that existed before being archived by using the `adbarchivechunk` command. The input data has not been compressed because the chunk has not been archived.

6.14 Estimating the size of the synonym dictionary file directory

This section describes how to determine the size of the synonym dictionary file directory. You must determine the size of the synonym dictionary file directory before performing synonym search.

> **Note**
>
> The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

Use the following formula to determine the size of the synonym dictionary file directory (*SYNDIRSIZE*).

Formula (gigabytes)

$$SYNDIRSIZE = \sum_{i=1}^{syndict\_num} (SYNDICTSIZE_{(i)})$$

Explanation of variables

syndict_num
Number of synonym dictionaries that will be created

SYNDICTSIZE(i)
Size of the synonym dictionary file of each synonym dictionary

Use the following formula to determine this value.

Formula (gigabytes)

$$SYNDICTSIZE = \frac{513 + \frac{syn\_num \times 32 + syn\_list\_file\_size + 4 \times syn\_group\_num}{1{,}024}}{1{,}024 \times 1{,}024} + CORSYNSIZE$$

syn_num
Total number of synonyms specified in the synonym list definition file

syn_list_file_size
Size of the synonym list definition file (bytes)

syn_group_num
Number of synonym groups

CORSYNSIZE
Size of a synonym dictionary supporting correction search

Add this variable when you create a synonym dictionary by specify the correction search option `CORRECTIONRULE`.

Use the following formula to determine this value.

Formula (gigabytes)

$$CORSYNSIZE = \frac{513 + \frac{syn\_num \times 32 + syn\_list\_file\_size \times 8 + 4 \times syn\_group\_num}{1,024}}{1,024 \times 1,024} \times 2$$

6.15 Estimating the size of the audit trail directory

This section describes how to determine the size of the *audit trail directory*. You must determine the size of the audit trail directory before using the audit trail facility.

> **Note**
>
> The audit trail directory is the directory specified for the `adb_audit_log_path` operand in the server definition.

Use the following formula to determine the size of the audit trail directory (*AUDFILE*).

Formula (megabytes)

AUDFILE = aud_log_max_size × aud_log_max_num

Explanation of variables

aud_log_max_size

Value specified for the `adb_audit_log_max_size` operand in the server definition

For details about the `adb_audit_log_max_size` operand, see 7.2.9 Operands related to audit trail facility (set format).

aud_log_max_num

Value specified for the `adb_audit_log_max_num` operand in the server definition

For details about the `adb_audit_log_max_num` operand, see 7.2.9 Operands related to audit trail facility (set format).

If 0 is specified for the `adb_audit_log_max_num` operand, there is no limit to the number of audit trail files that can be created. For this reason, you cannot determine a value for the variable *AUDFILE*. When specifying 0 for the `adb_audit_log_max_num` operand, determine the size of the audit trail directory by considering the following aspects of audit trail facility operation:

- The size of the output audit trail data and the number of audit trail files that will be created
- How often audit trail files are moved from the audit trail directory to the audit trail storage directory

For details about the operation of the audit trail facility, see 12. Audit Trail Facility Operations.

6.16 Estimating the size of the output-directory for common format audit trails

This section describes how to determine the size of the *output-directory for common format audit trails*. Common format audit trail files are output to the output-directory for common format audit trails.

> **Note**
>
> The output-directory for common format audit trails is the directory specified for the `-d` option of the `adbconvertaudittrailfile` command.

Use the following formula to determine the size of the output-directory for common format audit trails (*CMNAUDFILE*).

Formula (megabytes)

$$CMNAUDFILE = 512 \times 4 \times 2^{\#}$$

#

The size of each common format audit trail file that is created by the HADB server is 512 megabytes. A maximum of four files are created. Note that when old files are overwritten by new files, the data of the old files that are to be deleted might temporarily remain on the disk. Taking into account the size of data that might temporarily remain, when determining the size of the output-directory for common format audit trails, multiply the determined size by 2.

6.17 Estimating the size of the multi-node synonym dictionary storage directory

The method of determining the size of the multi-node synonym dictionary storage directory is the same as that in 6.14 Estimating the size of the synonym dictionary file directory. You need to determine the size of the multi-node synonym dictionary storage directory if you intend to use the multi-node function.

You need to estimate the size of the multi-node synonym dictionary storage directory for all nodes.

> **Note**
>
> The multi-node synonym dictionary storage directory is the directory specified for the `adb_syndict_node_storage_path` operand in the server definition.

6.18 Estimating the size of a file output by the adbexport command

This section describes how to determine the size of a data file that is output by the `adbexport` command.

To determine the total size of data files that are output (*EXPORTFILESIZE*), use the following formula.

Formula (kilobytes)

```
EXPORTFILESIZE = ↑(HEADERSIZE × file_num + ROWSIZE × row_num) ÷ 1,024↑
```

Explanation of variables

HEADERSIZE

Size of the column names to be output to the output data file

Add this value if you specify the `--with-column-name` option when running the `adbexport` command. If you do not specify the `--with-column-name` option, substitute 0.

Use the following formula to determine its value.

Formula (bytes)

```
HEADERSIZE = 202 × column_num
```

column_num

Number of columns of the data to be output

file_num

Number of output data files specified in the output data path file

ROWSIZE

Data size of one row

Use the following formula to determine its value.

Formula (bytes)

$$ROWSIZE = column\_num + \sum_{k=1}^{column\_num} column\_data\_size_{(k)}$$

column_data_size(k)

Column data length of the k-th column

The value to be substituted for the *column_data_size* variable differs depending on the data type of the column data. See the following table, and substitute the applicable value.

Table 6-26: Value to be substituted for column_data_size

No.	Classification	Data type	Data length (bytes)
1	Numeric data	`INTEGER`	20
2		`SMALLINT`	11
3		`DECIMAL`(m, n)#	$m + 2$
4		`DOUBLE PRECISION`	24
5	Character string data	`CHARACTER`(n)	$2 \times n + 2$
6		`VARCHAR`(n)	$2 \times d + 2$

No.	Classification	Data type	Data length (bytes)
7	Datetime data	DATE	12
8		TIME(*p*)	11 + *p*
9		TIMESTAMP(*p*)	22 + *p*
10	Binary data	BINARY(*n*)	2 × *n* + 2
11		VARBINARY(*n*)	2 × *d* + 2

Legend:

m, *n*: Positive integers

p: 0, 3, 6, 9 or 12

d: Actual data length

#

Indicates a fixed-point number that has a total of *m* digits, with *n* digits following the decimal point. If *m* is omitted, 38 is assumed.

row_num

Number of rows output by the adbexport command

6.19 Estimating the size of files required to reorganize a base table

This section describes how to determine the size of files that are required to reorganize a base table.

To reorganize a base table, use the `adbexport` command to export the data from the target base table, and then use the `adbimport` command to restore the data to the base table. For details about reorganization of a base table, see the following sections:

- 11.1.10 Reorganizing a single-chunk table
- 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization
- 11.4.15 Reorganizing a multi-chunk table: Reorganization of an entire table
- 11.4.16 Reorganizing a multi-chunk table: Reorganization using a sample shell script

If your system operation requires reorganization of a base table, estimate the total size of files that will be used for reorganization (*REORGFILESIZE*). Also, prepare a disk that has a sufficient amount of free space.

Use the following formula to determine the total size of files that are required for reorganization (*REORGFILESIZE*).

Formula (kilobytes)

```
REORGFILESIZE = EXPORTFILESIZE + IMPORTWRKFILESIZE
```

Explanation of variables

EXPORTFILESIZE

Total size of output data files that are output by the `adbexport` command to export the reorganization-target base table or chunk (kilobytes)

Determine the value as explained in 6.18 Estimating the size of a file output by the adbexport command.

IMPORTWRKFILESIZE

Size of the temporary work file that is used by the `adbimport` command to restore the data of the reorganization-target base table or chunk (kilobytes)

Determine the value by referring to ■ *Temporary work files for creating indexes* in 6.21.1 Estimating the size of the temporary work file for executing the adbimport command.

6.20 Estimating the size of an unload file

This section describes how to estimate the size of the unload file that is output when the `adbreorgsystemdata` command is executed. The size of an unload file is determined by the usage of system table (base table) to be reorganized.

Use the following formula to determine the size of an unload file (*UNLDFILESIZE*).

Formula (kilobytes)

$$UNLDFILESIZE = \uparrow ROWSIZE \times ROWNUM \div 1{,}024 \uparrow + 1$$

Explanation of variables

ROWSIZE

Data length of one row in an unload file (bytes)

This value differs depending on the system table (base table). See the following table to determine the value.

Table 6-27: Row data length of the unload file for each system table (base table)

No.	system table (base table)	Data length of one row in an unload file (bytes)
1	`STATUS_TABLES` table	417
2	`STATUS_COLUMNS` table	32,721
3	`STATUS_INDEXES` table	472
4	`STATUS_CHUNKS` table	1,436
5	`STATUS_SYNONYM_DICTIONARIES` table	2,237

ROWNUM

Number of rows in the system table (base table)

Obtain the number of rows in the reorganization-target system table (base table) by executing an SQL statement. For details, see (2) Checking the storage efficiency of a system table (base table) in 11.17.5 Checking the status and amount of use of system tables.

6.21 Estimating the size of the temporary work file for executing a command

This section explains how to determine the size of the temporary work file that is created when any of the following commands is executed:

- `adbimport` command
- `adbidxrebuild` command
- `adbmergechunk` command
- `adbunarchivechunk` command
- `adbreorgsystemdata` command
- `adbsyndict` command

If you plan to execute the commands listed above concurrently, determine the size of the temporary work file for each command. Then, add up the determined sizes.

Note that no temporary work file is created if a command not listed above is executed.

6.21.1 Estimating the size of the temporary work file for executing the adbimport command

This section explains how to estimate the size of the temporary work file that is created when the `adbimport` command is executed to import data.

The following temporary work files are created:

- Temporary work files for creating indexes
- Temporary work files for data compression

■ Temporary work files for creating indexes

When the `adbimport` command is executed on a table for which an index is defined, a temporary work file for creating indexes is created under the following directory:

- If the `-w` option is specified

  A temporary work file is created under the specified directory.
- If the `-w` option is not specified

  A temporary work file is created under the DB directory (`$DBDIR/ADBWORK`).

Use the following formula to determine the size of the temporary work file for creating indexes:

Formula (kilobytes)

$$\text{size-of-temporary-work-file-for-index-creation} = \left\lceil \sum_{i=1}^{idx\_num} \left(RECFILE(i) + SORTFILE(i) + SORTWORK(i) + IDXWORK(i) \right) \div 1{,}024 \right\rceil + \left\lceil \sum_{i=1}^{tix\_num} \left(TIXWORK(i) + TIXCORWORK(i) \right) \div 1{,}024 \right\rceil$$

Explanation of variables

idx_num

Number of B-tree indexes defined

RECFILE(i)

Index record file

Use the following formula to determine this value.

Formula (bytes)

$$RECFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times \left\{ row\_num \right\} + 640$$

KEYSZ(i)

Key length of the i-th B-tree index (bytes)

Determine the key length of the B-tree index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL(i)

Control information for the i-th B-tree index

- If all indexed columns have fixed-length keys: 10 bytes
- If any of the indexed columns have variable-length keys: 12 bytes

row_num

Number of rows to be stored by the `adbimport` command

SORTFILE(i)

Sort result file

Use the following formula to determine this value. For the variables *KEYSZ(i)*, *CTRL(i)*, and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times row\_num + TCTRL_{(i)}$$

TCTRL(i)

Control information for the i-th text index

- For a text index

  Use the formula shown below to determine its value.

  Formula

$$TCTRL_{(i)} = (26 + number\text{-}of\text{-}DB\text{-}area\text{-}files\text{-}in\text{-}the\text{-}data\text{-}DB\text{-}area\text{-}that\text{-}stores\text{-}tables \times 72) \times 16 + 4$$

- For an index other than a text index

  Assign 0.

SORTWORK(i)

Sort work file

Use the following formula to determine this value. For the variables *RECFILE(i)* and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTWORK_{(i)} = \left\{ RECFILE_{(i)} + 8 \times (row\_num) \right\} \times 2$$

IDXWORK(i)

Work file for creating B-tree indexes

Use the following formula to determine this value.

Formula (bytes)

$$IDXWORK_{(i)} = \left\{12 + KEYSZDB_{(i)}\right\} \times \left\{PIDX\_LEAF + PIDX_{(2)}\right\}$$

KEYSZDB(i)

Length of the i-th database storage key (bytes)

For details, see the description of the variable *KEYSZDB* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment. For variable-length data, always use the definition length as the key length.

PIDX_LEAF

Number of pages in the first level of the B-tree index (leaf pages)

Determine this value by using *Formula 1 (for determining PIDX_LEAF)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

PIDX(2)

Number of pages in the second level of the B-tree index

Determine this value by using *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

tix_num

Number of text indexes

TIXWORK(i)

Work file for creating text indexes

Use the following formula to determine its value.

Formula (bytes)

$$TIXWORK_{(i)} = row\_num \times string\_num_{(i)} \times 172 + row\_num \times (define\_num_{(i)} + 12)$$

row_num

Number of rows to be stored by the `adbimport` command

string_num(i)

Average number of characters stored in the columns in a text index

define_num(i)

Defined length of columns in a text index

TIXCORWORK(i)

Work file for creating text indexes (for correction search)

You must determine this variable if you add the notation-correction-search text-index specification when defining a text index.

Use the following formula to determine this value.

Formula (bytes)

$$TIXCORWORK_{(i)} = row\_num \times string\_num_{(i)} \times 202$$

row_num

Number of rows to be stored by the `adbimport` command

string_num(i)

Average number of characters stored in the columns in a text index

■ Temporary work files for data compression

A temporary work file for data compression is created only when storing data in a column store table.

Use the following formula to determine the size of the temporary work file for data compression:

Formula (kilobytes)

$$\text{Size of temporary work file for data compression} = \left\lceil \left\{ \left(\sum_{i=1}^{col\_num} \left(\left\lfloor \frac{col\_size(i) + 15}{16} \right\rfloor \times 16 \right) + \left\lfloor \frac{col\_num + 15}{16} \right\rfloor \times 16 \right) \times 262{,}144 \right\} \div 1{,}024 \right\rceil \times imp\_load\_rthd$$

Explanation of variables

col_num

Total number of columns in the table to be processed (columns)

col_size(i)

Data length of each column in table to be processed (bytes)

For details about the data length of each column, see Table 6-9: Data length of each data type.

imp_load_rthd

Use the following formula to determine the value:

```
value-specified-for-import-option-adb_import_rthd_num - 1
```

6.21.2 Estimating the size of the temporary work file for executing the adbidxrebuild command

This section explains how to estimate the size of the temporary work file that is created when the `adbidxrebuild` command is executed to rebuild an index.

When the `adbidxrebuild` command is executed on a table for which an index is defined, a temporary work file is created under the following directory:

- If the `-w` option is specified

  A temporary work file is created under the specified directory.
- If the `-w` option is not specified

  A temporary work file is created under the DB directory (`$DBDIR/ADBWORK`).

Use the following formula to determine the size of the temporary work file.

Formula (kilobytes)

$$\text{Size of the temporary work file} = \left\lceil \sum_{i=1}^{idx\_num} \left(RECFILE(i) + SORTFILE(i) + SORTWORK(i) + IDXWORK(i) \right) \div 1{,}024 \right\rceil + \left\lceil \sum_{i=1}^{tix\_num} \left(TIXWORK(i) + TIXCORWORK(i) \right) \div 1{,}024 \right\rceil$$

Explanation of variables

idx_num

Number of B-tree indexes defined

RECFILE(i)

Index record file

Use the following formula to determine this value.

Formula (bytes)

$$RECFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times \left\{ row\_num \right\} + 640$$

KEYSZ(i)

Key length of the i-th B-tree index (bytes)

Determine the key length of the B-tree index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL(i)

Control information for the i-th B-tree index

- If all indexed columns have fixed-length keys: 10 bytes
- If any of the indexed columns have variable-length keys: 12 bytes

row_num

Number of rows that are stored in the table targeted by the `adbidxrebuild` command

SORTFILE(i)

Sort result file

Use the following formula to determine this value. For the variables *KEYSZ(i)*, *CTRL(i)*, and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTFILE_{(i)} = \left\{KEYSZ_{(i)} + CTRL_{(i)}\right\} \times row\_num + TCTRL_{(i)}$$

TCTRL(i)

Control information for the i-th text index

- For a text index

  Use the following formula to determine its value.

  Formula

  $$TCTRL_{(i)} = (26 + \text{number of DB area files in the data DB area that stores tables} \times 72) \times 16 + 4$$

- For an index other than a text index

  Substitute 0.

SORTWORK(i)

Sort work file

Use the following formula to determine this value. For the variables *RECFILE(i)* and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTWORK_{(i)} = \left\{RECFILE_{(i)} + 8 \times (row\_num)\right\} \times 2$$

IDXWORK(i)

Work file for creating B-tree indexes

Use the following formula to determine this value.

Formula (bytes)

$$IDXWORK_{(i)} = \left\{12 + KEYSZDB_{(i)}\right\} \times \left\{PIDX\_LEAF + PIDX_{(2)}\right\}$$

KEYSZDB(i)

Length of the i-th database storage key (bytes)

For details, see the description of the variable *KEYSZDB* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment. For variable-length data, always use the definition length as the key length.

PIDX_LEAF

Number of pages in the first level of the B-tree index (leaf pages)

Determine this value by using *Formula 1 (for determining PIDX_LEAF)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

PIDX(2)

Number of pages in the second level of the B-tree index

Determine this value by using *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

tix_num

Number of text indexes

TIXWORK(i)

Work file for creating text indexes

Use the following formula to determine its value.

Formula (bytes)

$$TIXWORK_{(i)} = row\_num \times string\_num_{(i)} \times 172 + row\_num \times (define\_num_{(i)} + 12)$$

row_num

Number of rows that are stored in the table targeted by the `adbidxrebuild` command

string_num(i)

Average number of characters stored in the columns in a text index

define_num(i)

Defined length of columns in a text index

TIXCORWORK(i)

Work file for creating text indexes (for correction search)

You must determine this variable if you add the notation-correction-search text-index specification when defining a text index.

Use the following formula to determine this value.

Formula (bytes)

$$TIXCORWORK_{(i)} = row\_num \times string\_num_{(i)} \times 202$$

row_num

Number of rows that are stored in the table targeted by the `adbidxrebuild` command

string_num(i)

Average number of characters stored in the columns in a text index

6.21.3 Estimating the size of the temporary work file for executing the adbmergechunk command

This section explains how to estimate the size of the temporary work file that is created when the `adbmergechunk` command is executed to merge multiple chunks.

When the adbmergechunk command is executed on a table for which an index is defined, a temporary work file is created under the following directory:

- If the -w option is specified

  A temporary work file is created under the specified directory.
- If the -w option is not specified

  A temporary work file is created under the DB directory (\$DBDIR/ADBWORK).

Use the following formula to determine the size of the temporary work file.

Formula (kilobytes)

$$\text{Size of the temporary work file} = \left\lceil \sum_{i=1}^{idx\_num} \left(RECFILE_{(i)} + SORTFILE_{(i)} + SORTWORK_{(i)} + IDXWORK_{(i)}\right) \div 1{,}024 \right\rceil + \left\lceil \sum_{i=1}^{tix\_num} \left(TIXWORK_{(i)} + TIXCORWORK_{(i)}\right) \div 1{,}024 \right\rceil$$

Explanation of variables

idx_num

Number of B-tree indexes defined

RECFILE(i)

Index record file

Use the following formula to determine this value.

Formula (bytes)

$$RECFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times \left\{ row\_num \right\} + 640$$

KEYSZ(i)

Key length of the i-th B-tree index (bytes)

Determine the key length of the B-tree index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL(i)

Control information for the i-th B-tree index

- If all indexed columns have fixed-length keys: 10 bytes
- If any of the indexed columns have variable-length keys: 12 bytes

row_num

Number of rows that are stored in the merge-source chunks of the table targeted by the adbmergechunk command

SORTFILE(i)

Sort result file

Use the following formula to determine this value. For the variables *KEYSZ(i)*, *CTRL(i)*, and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTFILE_{(i)} = \left\{KEYSZ_{(i)} + CTRL_{(i)}\right\} \times row\_num + TCTRL_{(i)}$$

TCTRL(i)

Control information for the i-th text index

- For a text index

  Use the formula below to determine its value.

  Formula

  $$TCTRL_{(i)} = (26 + number\ of\ DB\ area\ files\ in\ the\ data\ DB\ area\ that\ stores\ tables \times 72) \times 16 + 4$$

- For an index other than a text index

  Substitute 0.

SORTWORK(i)

Sort work file

Use the following formula to determine this value. For the variables *RECFILE(i)* and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTWORK_{(i)} = \left\{RECFILE_{(i)} + 8 \times (row\_num)\right\} \times 2$$

IDXWORK(i)

Work file for creating B-tree indexes

Use the following formula to determine this value.

Formula (bytes)

$$IDXWORK_{(i)} = \left\{12 + KEYSZDB_{(i)}\right\} \times \left\{PIDX\_LEAF + PIDX_{(2)}\right\}$$

KEYSZDB(i)

Length of the i-th database storage key (bytes)

For details, see the description of the variable *KEYSZDB* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment. For variable-length data, always use the definition length as the key length.

PIDX_LEAF

Number of pages in the first level of the B-tree index (leaf pages)

Determine this value by using *Formula 1 (for determining PIDX_LEAF)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

PIDX(2)

Number of pages in the second level of the B-tree index

Determine this value by using *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

tix_num

Number of text indexes

TIXWORK(i)

Work file for creating text indexes

Use the following formula to determine its value.

Formula (bytes)

$$TIXWORK_{(i)} = row\_num \times string\_num_{(i)} \times 172 + row\_num \times (define\_num_{(i)} + 12)$$

row_num

Number of rows that are stored in the merge-source chunks of the table targeted by the `adbmergechunk` command

string_num(i)

Average number of characters stored in the columns in a text index

define_num(i)

Defined length of columns in a text index

TIXCORWORK(i)

Work file for creating text indexes (for correction search)

You must determine this variable if you add the notation-correction-search text-index specification when defining a text index.

Use the following formula to determine this value.

Formula (bytes)

$$TIXCORWORK_{(i)} = row\_num \times string\_num_{(i)} \times 202$$

row_num

Number of rows that are stored in the merge-source chunks of the table targeted by the `adbmergechunk` command

string_num(i)

Average number of characters stored in the columns in a text index

6.21.4 Estimating the size of the temporary work file for executing the adbunarchivechunk command

This subsection explains how to estimate the size of the temporary work file that is created when the `adbunarchivechunk` command is executed to unarchive a chunk.

When the `adbunarchivechunk` command is executed on a table for which an index is defined, a temporary work file is created under the following directory:

- If the `-w` option is specified

A temporary work file is created under the specified directory.

- If the `-w` option is not specified

  A temporary work file is created under the DB directory (`$DBDIR/ADBWORK`).

Use the following formula to determine the size of the temporary work file.

Formula (kilobytes)

$$\text{Size of the temporary work file} = \left\lceil \sum_{i=1}^{idx\_num} \left(RECFILE(i) + SORTFILE(i) + SORTWORK(i) + IDXWORK(i) \right) \div 1{,}024 \right\rceil + \left\lceil \sum_{i=1}^{tix\_num} \left(TIXWORK(i) + TIXCORWORK(i) \right) \div 1{,}024 \right\rceil$$

Explanation of variables

idx_num

Number of B-tree indexes defined

RECFILE(i)

Index record file

Use the following formula to determine this value.

Formula (bytes)

$$RECFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times \left\{ row\_num \right\} + 640$$

KEYSZ(i)

Key length of the i-th B-tree index (bytes)

Determine the key length of the B-tree index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL(i)

Control information for the i-th B-tree index

- If all indexed columns have fixed-length keys: 10 bytes
- If any of the indexed columns have variable-length keys: 12 bytes

row_num

Number of rows that are stored in the archive file for the chunk of the table to be processed by the `adbunarchivechunk` command

SORTFILE(i)

Sort result file

Use the following formula to determine this value. For the variables *KEYSZ(i)*, *CTRL(i)*, and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times row\_num + TCTRL_{(i)}$$

TCTRL(i)

Control information for the i-th text index

- For a text index

  Use the following formula to determine this value.

  Formula

  TCTRL(i) =
  (26 + *number of DB area files making up the data DB area for storing tables* x 72) x 16 + 4

- For an index other than a text index

  Substitute 0.

SORTWORK(i)

Sort work file

Use the following formula to determine this value. For the variables *RECFILE(i)* and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTWORK_{(i)} = \left\{ RECFILE_{(i)} + 8 \times (row\_num) \right\} \times 2$$

IDXWORK(i)

Work file for creating B-tree indexes

Use the following formula to determine this value.

Formula (bytes)

$$IDXWORK_{(i)} = \left\{ 12 + KEYSZDB_{(i)} \right\} \times \left\{ PIDX\_LEAF + PIDX_{(2)} \right\}$$

KEYSZDB(i)

Length of the i-th database storage key (bytes)

For details, see the description of the variable *KEYSZDB* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment. For variable-length data, always use the definition length as the key length.

PIDX_LEAF

Number of pages in the first level of the B-tree index (leaf pages)

Determine this value by using *Formula 1 (for determining PIDX_LEAF)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

PIDX(2)

Number of pages in the second level of the B-tree index

Determine this value by using *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

tix_num

Number of text indexes

TIXWORK(i)

Work file for creating text indexes

Use the following formula to determine this value.

Formula (bytes)

$$TIXWORK_{(i)} = row\_num \times string\_num_{(i)} \times 172 + row\_num \times (define\_num_{(i)} + 12)$$

row_num

Number of rows that are stored in the archive file for the chunk of the table to be processed by the `adbunarchivechunk` command

string_num(i)

Average number of characters stored in the columns in a text index

define_num(i)

Defined length of columns in a text index

TIXCORWORK(i)

Work file for creating text indexes (for correction search)

You must determine this variable if you add the notation-correction-search text-index specification when defining a text index.

Use the following formula to determine this value.

Formula (bytes)

$$TIXCORWORK_{(i)} = row\_num \times string\_num_{(i)} \times 202$$

row_num

Number of rows that are stored in the archive file for the chunk of the table to be processed by the `adbunarchivechunk` command

string_num(i)

Average number of characters stored in the columns in a text index

6.21.5 Estimating the size of the temporary work file for executing the adbreorgsystemdata command

This subsection explains how to estimate the size of the temporary work file that is created when the `adbreorgsystemdata` command is executed to reorganize a system table (base table).

When the `adbreorgsystemdata` command is executed on a table for which an index is defined, a temporary work file is created under the following directory:

- If the `-w` option is specified

  A temporary work file is created under the specified directory.
- If the `-w` option is not specified

  A temporary work file is created under the DB directory (`$DBDIR/ADBWORK`).

Use the following formula to determine the size of the temporary work file.

Formula (kilobytes)

$$\text{Size of the temporary work file} = \left\uparrow \sum_{i=1}^{idx\_num} \left(RECFILE(i) + SORTFILE(i) + SORTWORK(i) + IDXWORK(i)\right) / 1{,}024 \right\uparrow$$

Explanation of variables

idx_num

Number of B-tree indexes defined for the system table (base table) to be reorganized

For details about the B-tree indexes defined for the system table (base table) to be reorganized, see C.8 B-tree indexes of system tables (base tables).

RECFILE(i)

Index record file

Use the following formula to determine this value.

Formula (bytes)

$$RECFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times \left\{ row\_num \right\} + 640$$

KEYSZ(i)

Key length of the i-th B-tree index (bytes)

Determine the key length of the B-tree index based on 5.8.4 Determining the key length (KEYSZ) of a B-tree index. For variable-length data, use the actual data length.

CTRL(i)

Control information for the i-th B-tree index

- If all indexed columns have fixed-length keys: 10 bytes
- If any of the indexed columns have variable-length keys: 12 bytes

row_num

Number of rows that are stored in the table targeted by the `adbreorgsystemdata` command

SORTFILE(i)

Sort result file

Use the following formula to determine this value. For the variables *KEYSZ(i)*, *CTRL(i)*, and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTFILE_{(i)} = \left\{ KEYSZ_{(i)} + CTRL_{(i)} \right\} \times row\_num$$

SORTWORK(i)

Sort work file

Use the following formula to determine this value. For the variables *RECFILE(i)* and *row_num*, determine their values by referring to the formula for the preceding index record file (variable *RECFILE(i)*).

Formula (bytes)

$$SORTWORK_{(i)} = \left\{ RECFILE_{(i)} + 8 \times (row\_num) \right\} \times 2$$

IDXWORK(i)

Work file for creating B-tree indexes

Use the following formula to determine this value.

Formula (bytes)

$$IDXWORK_{(i)} = \left\{ 12 + KEYSZDB_{(i)} \right\} \times \left\{ PIDX\_LEAF + PIDX_{(2)} \right\}$$

KEYSZDB(i)

Length of the i-th database storage key (bytes)

For details, see the description of the variable *KEYSZDB* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment. For variable-length data, always use the definition length as the key length.

PIDX_LEAF

Number of pages in the first level of the B-tree index (leaf pages)

Determine this value by using *Formula 1 (for determining PIDX_LEAF)* in (1) Determining the number of storage pages used in the lower page segment (variable IP_LOWER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

PIDX(2)

Number of pages in the second level of the B-tree index

Determine this value by using *Formula 1 (for determining PIDX(k))* in (2) Determining the number of storage pages used in the upper page segment (variable IP_UPPER(i)) under 5.8.3 Determining the number of storage pages for each B-tree index segment.

6.21.6 Estimating the size of the temporary work file for executing the adbsyndict command

This subsection explains how to determine the size of the temporary work file that is created when the `adbsyndict` command is executed.

Use the following formula to determine the size of the temporary work file.

Formula (kilobytes)

$$\text{Size of the temporary work file} = \frac{syn\_list\_file\_size \times 8}{1{,}024}$$

Explanation of variables

syn_list_file_size

Size of the synonym list definition file (bytes)

6.22 Estimating the increase in the amount of data that occurs during command execution

This section explains how to estimate the increase in the amount of data that occurs when the following command is executed:

- `adbmergechunk` command

6.22.1 Estimating the increase in the amount of data that occurs when the adbmergechunk command is executed

When indexes are defined for a table designated for merge chunk processing, the amount of index data increases while the `adbmergechunk` command is being executed and when its execution is halted.

Therefore, when you execute the `adbmergechunk` command, add the amount of data needed to rebuild the index at the merge-target chunk to the values estimated in sections 5.8.3 Determining the number of storage pages for each B-tree index segment, 5.8.5 Determining the number of storage pages for each text index segment, and 5.8.6 Determining the number of segments for storing each range index. For the amount of data needed to rebuild an index, assume a value equivalent to the sum total of the amount of index data in the merge-target chunk.

For details about why the amount of index data increases during execution of the `adbmergechunk` command, see the note in (3) Using the adbmergechunk command to merge chunks under 11.4.9 Merging chunks (to reduce the number of chunks).

6.23 Points to consider when executing commands concurrently

This subsection explains points to consider when you execute any of the following commands concurrently:

- `adbarchivechunk` command
- `adbchgchunkcomment` command
- `adbchgchunkstatus` command
- `adbdbstatus` command
- `adbexport` command
- `adbgetcst` command
- `adbidxrebuild` command
- `adbimport` command
- `adbmergechunk` command
- `adbmodbuff` command
- `adbreorgsystemdata` command
- `adbsql` command
- `adbstat` command
- `adbsyndict` command
- `adbunarchivechunk` command

When you execute these commands concurrently, you must also take into consideration the application programs that are running at the same time.

6.23.1 Maximum number of commands that can be executed concurrently

For the commands listed in 6.23 Points to consider when executing commands concurrently, the maximum number of concurrent executions is determined by the value specified for the `adb_sys_max_users` operand in the server definition and the number of applications programs that are connected to the HADB server. Use the following formula to determine the maximum number of commands that can be executed concurrently.

Formula

```
Maximum number of commands that can be executed concurrently = MAX_USERS - AP_CNCT
_NUM
```

Explanation of the variables

MAX_USERS

Maximum number of concurrent connections specified in the `adb_sys_max_users` operand in the server definition

For details, see the explanation of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).

AP_CNCT_NUM

Number of application programs currently connected to the HADB server

You can use the `adbls -d cnct` command to check the number of application programs currently connected to the HADB server. For details about the `adbls -d cnct` command, see *adbls -d cnct (Display the Connection Status)* in the manual *HADB Command Reference*.

No more than the maximum number of commands determined by the above formula can be executed concurrently. If the maximum number of concurrent executions is exceeded, an error occurs (the `KFAA30932-E` message is output).

Note that there is no upper limit on the maximum number of concurrent executions of the `adbstat` command.

6.23.2 Points to consider about the number of processing real threads to be used during command execution

(1) Maximum number of processing real threads that can be used during command execution

When you execute any of the commands listed in Table 6-28: Operands and command options for specifying the number of processing real threads to be used for command execution, make sure that the number of processing real threads that can be used during command execution as determined using the following formula can be allocated:

Formula

```
Maximum number of processing real threads that can be used during command executio
n =
                                        SYS_RTHD_NUM - AP_USE_RTHD_NUM
```

If you execute multiple commands concurrently, the value to be used is the sum total of the number of processing real threads required for all commands that are executed concurrently.

Explanation of the variables

SYS_RTHD_NUM

Value specified for the `adb_sys_rthd_num` operand in the server definition

For the `adb_sys_rthd_num` operand, specify the maximum number of processing real threads that are used to process SQL statements and commands. For details about the `adb_sys_rthd_num` operand, see the explanation of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

AP_USE_RTHD_NUM

Total number of processing real threads used for executing application programs

For details about the number of processing real threads used for executing application programs, see the following operands:

- `adb_sql_exe_max_rthd_num` operand in the server definition

  See the explanation of the adb_sql_exe_max_rthd_num operand in 7.2.2 Operands related to performance (set format).

- `adb_sql_exe_max_rthd_num` operand in the client definition

  See *Operands related to performance* in the *HADB Application Development Guide*.

(2) Operands and command options for specifying the number of processing real threads to be used for command execution

Use the operands and command options described in the following table to specify the number of processing real threads to be used for command execution.

Table 6-28: Operands and command options for specifying the number of processing real threads to be used for command execution

No.	Command name	Operand name	Command option name	Minimum number of processing real threads required
1	`adbarchivechunk` command	`adb_sql_exe_max_rthd_num` operand in the server definition	Archive chunk option `adb_arcv_rthd_num`	3
2	`adbexport` command		Export option `adb_export_rthd_num`	3
3	`adbgetcst` command		Cost-information collection option `adb_getcst_rthd_num`	2
4	`adbidxrebuild` command		Index rebuild option `adb_idxrebuild_rthd_num`	3
5	`adbimport` command		Import option `adb_import_rthd_num`	2
6	`adbmergechunk` command		Merge chunk option `adb_mergechunk_rthd_num`	3
7	`adbunarchivechunk` command		Unarchive chunk option `adb_unarcv_rthd_num`	2
8	`adbsql` command	• `adb_sql_exe_max_rthd_num` operand in the server definition • `adb_sql_exe_max_rthd_num` operand in the client definition	None	0
9	`adbreorgsystemdata` command#	None	None	3

#

The `adbreorgsystemdata` command uses three processing real threads. You cannot specify the number of processing real threads in a command option.

In the `adb_sql_exe_max_rthd_num` operand, you specify the maximum number of SQL processing real threads. If a command option is specified, the command option takes precedence over the `adb_sql_exe_max_rthd_num` operand.

When you execute commands concurrently, if the maximum number of processing real threads to be used for command execution as determined in (1) Maximum number of processing real threads that can be used during command execution is exceeded, the process of allocating processing real threads might be placed in wait status. If this occurs, command processing continues after other commands or applications terminate, and after the required number of processing real threads has been allocated.

Therefore, when executing the commands explained in Table 6-28: Operands and command options for specifying the number of processing real threads to be used for command execution concurrently, specify an appropriate number of processing real threads in the operands and command options explained in Table 6-28: Operands and command options for specifying the number of processing real threads to be used for command execution.

> **Note**
>
> To check the commands that are waiting for processing real threads to be available, execute the `adbls -d cnct` command. For details about the `adbls -d cnct` command, see *adbls -d cnct (Display the Connection Status)* in the manual *HADB Command Reference*.

(3) Estimating the number of processing real threads to be used for concurrent command execution

If you do not estimate properly the number of processing real threads to be used for command execution, command execution performance will be adversely affected. The following provides guidelines for estimating the number of processing real threads:

- **For periods during which the number of concurrently executing commands is small**

  Use a number of processing real threads that is approximately 80% of the maximum number of processing real threads used during command execution as determined in (1) Maximum number of processing real threads that can be used during command execution.
- **For periods during which the number of concurrently executing commands is large**

  Use a number of processing real threads that will not cause an operational problem, even if the process of allocating processing real threads is placed in wait status during concurrent command execution.

6.23.3 Points to consider about locking during concurrent command execution

When an HADB server executes the commands listed in 6.23 Points to consider when executing commands concurrently, it reserves locked resources according to 2.10.4 Locked resources that are reserved and their lock modes.

When you execute multiple commands concurrently, make sure that no contention occurs during the process of locked resource reservation. If contention occurs during the process of locked resource reservation, an error normally occurs (the `KFAA50290-E` message is output).

However, there are some exceptions (cases in which the transaction waits until a locked resource can be reserved without causing an error). For details, see 2.10.4 Locked resources that are reserved and their lock modes.

6.23.4 Points to consider about memory requirements during concurrent command execution

The amount of memory used when commands are executed concurrently is the combined total of the memory used by the individual commands.

For details about the amount of memory used by the commands, see 6.3 Estimating the HADB server's memory requirement.

6.23.5 Points to consider about the size of the system log files during concurrent command execution

The size of the system log files used when commands are executed concurrently is the combined total of the sizes of the system log files used by the individual commands.

For details about the size of the system log file used by the commands, see 6.12 Estimating the size of the system log files.

6.24 Points to consider when using the client-group facility

This section explains items that must be taken into consideration when the client-group facility is used.

6.24.1 Points to consider when specifying the number of connections for a group

When you use the client-group facility, you can specify the number of connections that can be made available to HADB clients and commands. The client-group facility supports the following types of connection counts:

- Maximum number of concurrent connections
- Guaranteed minimum number of concurrent connections

For details about the maximum number of concurrent connections and guaranteed minimum number of concurrent connections of the client-group facility, see 2.12.3 Setting the numbers of connections and processing real threads for each group.

(1) Points to consider when specifying the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for a group

Consider the following when you use the client-group facility and you specify a maximum number of concurrent connections and a guaranteed minimum number of concurrent connections for a group:

- What would be the maximum number of concurrent connections to the HADB server?

  Evaluate the value of the `adb_sys_max_users` operand in the server definition. For details about the `adb_sys_max_users` operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).
- What would be the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for a group?

  You can use the `adbcltgrp` operand in the server definition to specify the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for a group. For details about the `adbcltgrp` operand in the server definition, see 7.2.12 Operands and options related to the client-group facility (command format).
- How many connections would be needed for HADB clients and commands that belong to no group?

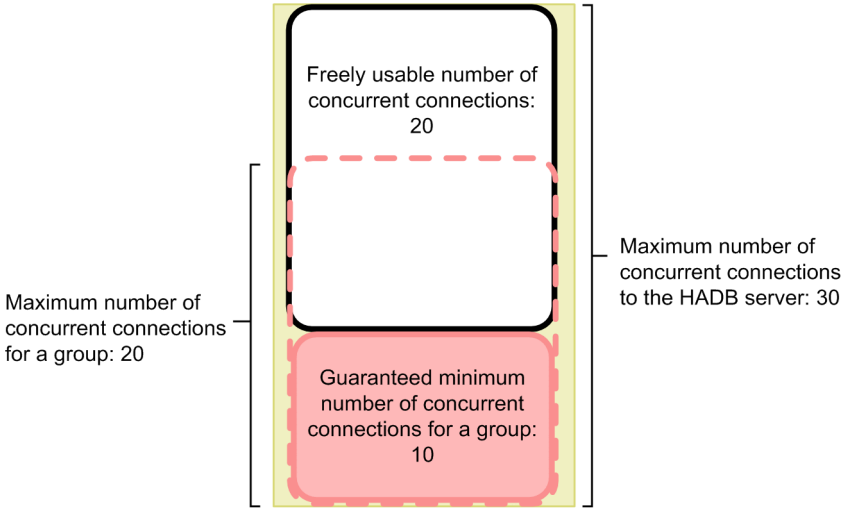
  The number of connections usable by HADB clients and commands that belong to no group is determined by the maximum number of concurrent connections to the HADB server and by the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for each group.

  The number of connections usable by HADB clients and commands that belong to no group is within the range of the *freely usable number of concurrent connections*. The number of freely usable concurrent connections is the maximum number of concurrent connections to the HADB server minus the guaranteed minimum number of concurrent connections for each group.

  Taking this relationship into account, specify the appropriate maximum number of concurrent connections and guaranteed minimum number of concurrent connections for each group.

The following figure provides an overview of the numbers of connections that are allocated when the client-group facility is used.

Figure 6-5: Overview of the numbers of connections that are allocated when the client-group facility is used



(2) How to determine the numbers of connections when the client-group facility is used

This subsection explains how to determine the numbers of connections when the client-group facility is used.

■ How to determine the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for a group

The maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for a group are determined by the values of the `adbcltgrp` and `adb_sys_max_users` operands in the server definition. The HADB clients and commands that belong to a group can use connections within the range of the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections specified for the group. The following table explains the maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for a group.

Table 6-29: Maximum number of concurrent connections and the guaranteed minimum number of concurrent connections for a group

No.	Type of connection count	Value
1	Maximum number of concurrent connections	The value is determined by the following formula: MIN(*A*, (*B* − *C*)) Explanation of variables: *A*: Value specified for the `-m` option of the `adbcltgrp` operand in the server definition in which the group to which clients and commands belong has been set *B*: Value specified for the `adb_sys_max_users` operand in the server definition *C*: Total of the values specified for the `-u` option of the `adbcltgrp` operand in other server definitions
2	Guaranteed minimum number of concurrent connections	The value is as follows: Value of the `-u` option in the `adbcltgrp` operand in the server definition for the target group

■ How to determine the number of freely usable concurrent connections

The freely usable concurrent connections are available to all HADB clients and commands regardless of whether they belong to a group. Use the following formula to determine the number of freely usable concurrent connections.

Formula

A − *B*

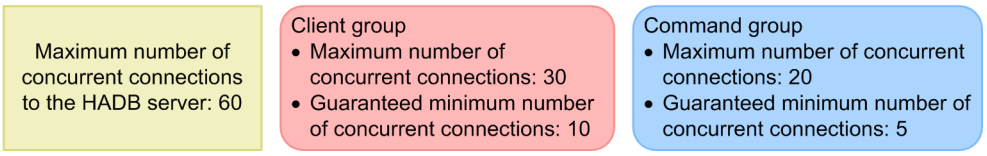
Explanation of variables:
A: Value specified for the `adb_sys_max_users` operand in the server definition
B: Total of the values specified for the `-u` option of the `adbcltgrp` operand in the server definition

(3) Relationships among connection counts when the client-group facility is used

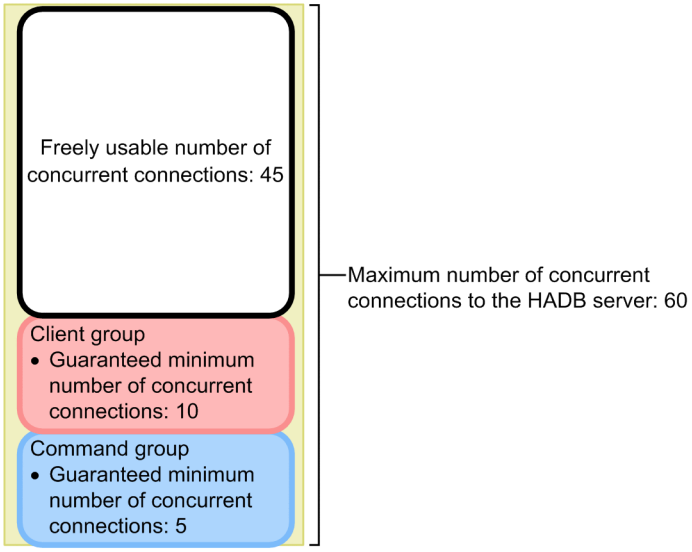
The following figure shows the relationships among the different types of connection counts explained in (2) How to determine the numbers of connections when the client-group facility is used when the client-group facility is used.

Figure 6-6: Relationships among connection counts when the client-group facility is used

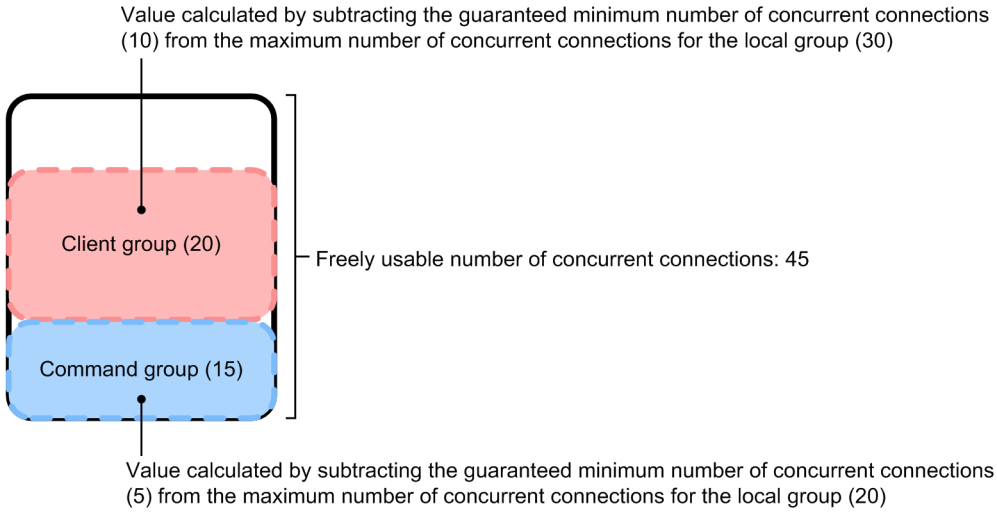
■ Example of settings for *maximum number of concurrent connections to the HADB server* and *maximum number of concurrent connections and guaranteed minimum number of concurrent connections for a group*



■ Relationship between *guaranteed minimum number of concurrent connections for a group* and *freely usable number of concurrent connections*



■ Relationship between *maximum number of concurrent connections for a group* and *freely usable number of concurrent connections*



Explanation:

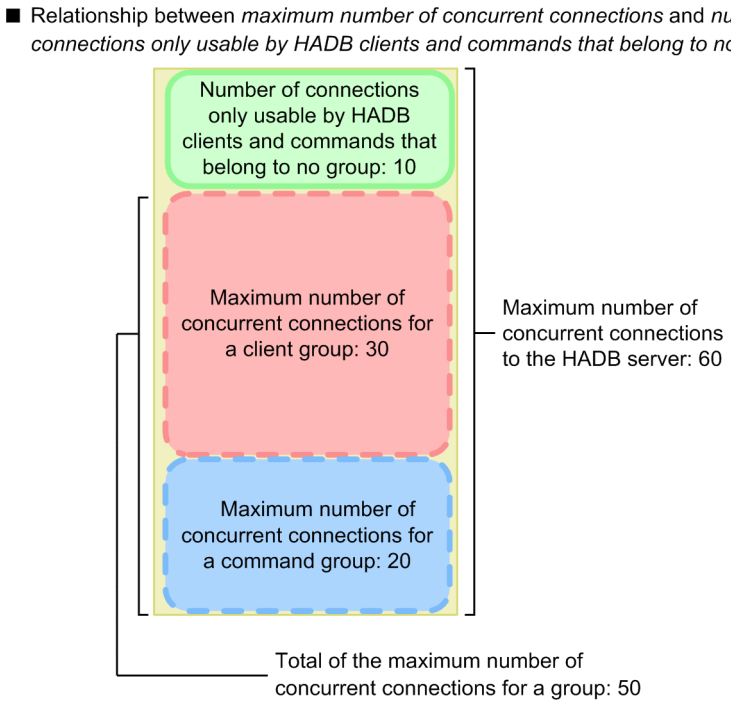
- The guaranteed minimum number of concurrent connections for the client group (10) and the guaranteed minimum number of concurrent connections for the command group (5) are allocated from the maximum number of concurrent connections to the HADB server (60). Therefore, the remaining connections (45) belong to the freely usable concurrent connections. The freely usable concurrent connections are available to all HADB clients and commands regardless of whether they belong to a group.
- Of the freely usable concurrent connections (45), each group can use up to its maximum number of concurrent connections minus its guaranteed minimum number of concurrent connections. In this example, the client group

can use up to its local group's maximum number of concurrent connections (30) minus its guaranteed minimum number of concurrent connections (10) as freely usable concurrent connections (20). The command group can use up to its maximum number of concurrent connections (20) minus its guaranteed minimum number of concurrent connections (5) as freely usable concurrent connections (15).

Note

Depending on the sum of all groups' maximum numbers of concurrent connections, a portion of the freely usable concurrent connections might become available only to HADB clients and commands that belong to no group. The following figure shows the relationship between the number of connections that are available only to HADB clients and commands that do not belong to any group, and the sum of the maximum numbers of concurrent connections for all groups. This figure is based on the settings shown in Figure 6-6: Relationships among connection counts when the client-group facility is used.

Figure 6-7: Relationship between the number of connections that are available only to HADB clients and commands that belong to no group and the sum of the maximum numbers of concurrent connections for all groups



Explanation:

If the sum of the maximum numbers of concurrent connections specified for all groups is smaller than the maximum number of concurrent connections to the HADB server, some of the connections will never be used by any group. These connections become available only to HADB clients and commands that belong to no group.

In this example, the sum of the maximum numbers of concurrent connections specified for all groups (50) is less than the maximum number of concurrent connections to the HADB server (60). As a result, 10 connections will never be used by any group. These 10 connections become available only to HADB clients and commands that belong to no group.

(4) Enabling output of warning messages regarding the maximum number of concurrent connections

By specifying the `-w` option for the `adbcltgrp` operand in the server definition, you can have HADB output the warning message `KFAA40020-W` when the number of concurrent connections approaches the maximum number of concurrent connections available to the group. For details about the `adbcltgrp` operand, see the explanation of the `adbcltgrp` operand in 7.2.12 Operands and options related to the client-group facility (command format).

6.24.2 Points to consider when specifying the number of processing real threads for a group

When you use the client-group facility, you can specify for each group the number of processing real threads that can be used by HADB clients and commands. The client-group facility supports the following types of processing real thread counts:

- Maximum number of processing real threads usable by a group
- Guaranteed minimum number of processing real threads usable by a group

For details about the maximum number and guaranteed minimum number of processing real threads usable by a group, see 2.12.3 Setting the numbers of connections and processing real threads for each group.

(1) Points to consider when specifying the maximum number of processing real threads and the guaranteed minimum number of processing real threads usable by a group

Consider the following when you use the client-group facility and you specify a maximum number of processing real threads and a guaranteed minimum number of processing real threads usable by a group:

- What would be the number of processing real threads on the HADB server?

  Evaluate the value of the `adb_sys_rthd_num` operand in the server definition. For details about the `adb_sys_rthd_num` operand in the server definition, see the description of the `adb_sys_rthd_num` operand in 7.2.2 Operands related to performance (set format).
- What would be the maximum number of processing real threads and the guaranteed minimum number of processing real threads usable by a group?

  You can use the `adbcltgrp` operand in the server definition to specify the maximum number of processing real threads and the guaranteed minimum number of processing real threads usable by a group. For details about the `adbcltgrp` operand in the server definition, see 7.2.12 Operands and options related to the client-group facility (command format).
- How many processing real threads would be needed for HADB clients and commands that belong to no group?

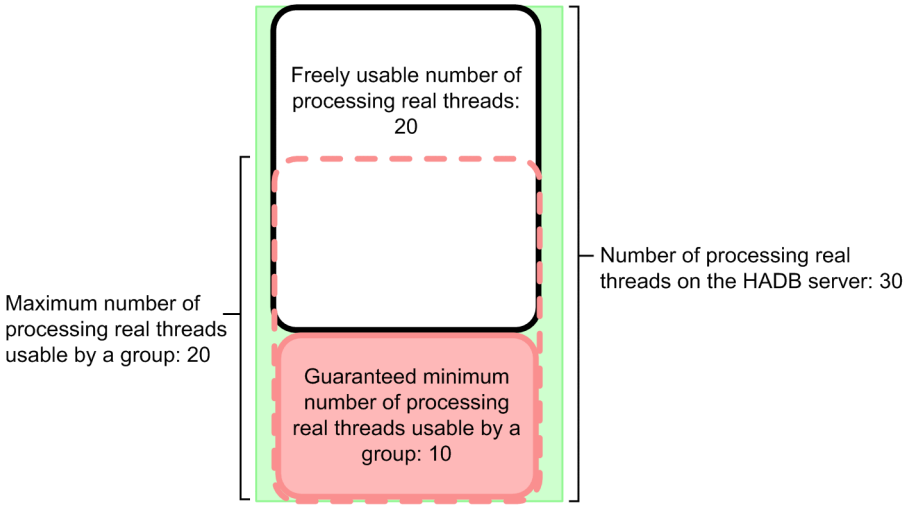
  The number of processing real threads usable by HADB clients and commands that belong to no group is determined by the number of processing real threads on the HADB server and by the maximum number of processing real threads and the guaranteed minimum number of processing real threads that can be used by each group.

  The number of processing real threads usable by HADB clients and commands that belong to no group is within the range of the *freely usable number of processing real threads*. The number of freely usable processing real threads is the number of processing real threads on the HADB server minus the guaranteed minimum number of processing real threads that can be used by each group.

  Taking this relationship into account, specify the appropriate maximum number of processing real threads and guaranteed minimum number of processing real threads usable by a group.

The following figure provides an overview of the range of processing real threads that are used when the client-group facility is used.

Figure 6-8: Overview of the range of processing real threads that are used when the client-group facility is used



(2) How to determine the numbers of processing real threads when the client-group facility is used

This subsection explains how to determine the numbers of processing real threads when the client-group facility is used.

■ How to determine the maximum number of processing real threads and the guaranteed minimum number of processing real threads usable by a group

The maximum number of processing real threads and the guaranteed minimum number of processing real threads usable by a group are determined by the values of the `adbcltgrp` and `adb_sys_rthd_num` operands in the server definition. The HADB clients and commands that belong to a group can use processing real threads within the range of the maximum number of processing real threads and the guaranteed minimum number of processing real threads specified for the group. The following table explains the maximum number of processing real threads and the guaranteed minimum number of processing real threads usable by a group.

Table 6-30: Maximum number of processing real threads and the guaranteed minimum number of processing real threads usable by a group

No.	Type of processing real thread count	Value
1	Maximum number of processing real threads[#]	The value is determined by the following formula: MIN(*A*, (*B* − *C*)) Explanation of variables: *A*: Value specified for the `-r` option of the `adbcltgrp` operand in the server definition in which the group to which clients and commands belong has been set *B*: Value specified for the `adb_sys_rthd_num` operand in the server definition *C*: Total of the values specified for the `-e` option of the `adbcltgrp` operand in other server definitions
2	Guaranteed minimum number of processing real threads that can be used	The value is as follows: Value of the `-e` option in the `adbcltgrp` operand in the server definition for the target group

#

The following cannot be executed unless the *minimum number of processing real threads required* can be allocated. Therefore, the maximum number of processing real threads must be adjusted so that a number of processing real threads at least equal to the *minimum number of processing real threads required* is allocated. If a number of processing real threads at least equal to the *minimum number of processing real threads required* cannot be allocated, execution of the following commands might result in an error:

- `adbimport` command
- `adbidxrebuild` command
- `adbgetcst` command
- `adbexport` command
- `adbmergechunk` command
- `adbarchivechunk` command
- `adbunarchivechunk` command
- `adbreorgsystemdata` command

For details about the *minimum number of processing real threads required* by each command, see Table 6-28: Operands and command options for specifying the number of processing real threads to be used for command execution in (2) Operands and command options for specifying the number of processing real threads to be used for command execution under 6.23.2 Points to consider about the number of processing real threads to be used during command execution.

■ **How to determine the number of freely usable processing real threads**

The freely usable processing real threads are available to all HADB clients and commands regardless of whether they belong to a group. Use the following formula to determine the number of freely usable processing real threads.

Formula

A − *B*

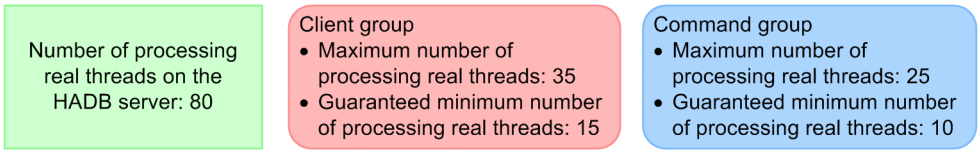
Explanation of variables:
A: Value specified for the `adb_sys_rthd_num` operand in the server definition
B: Total of the values specified for the `-e` option of the `adbcltgrp` operand in the server definition

(3) Relationships among processing real thread counts when the client-group facility is used

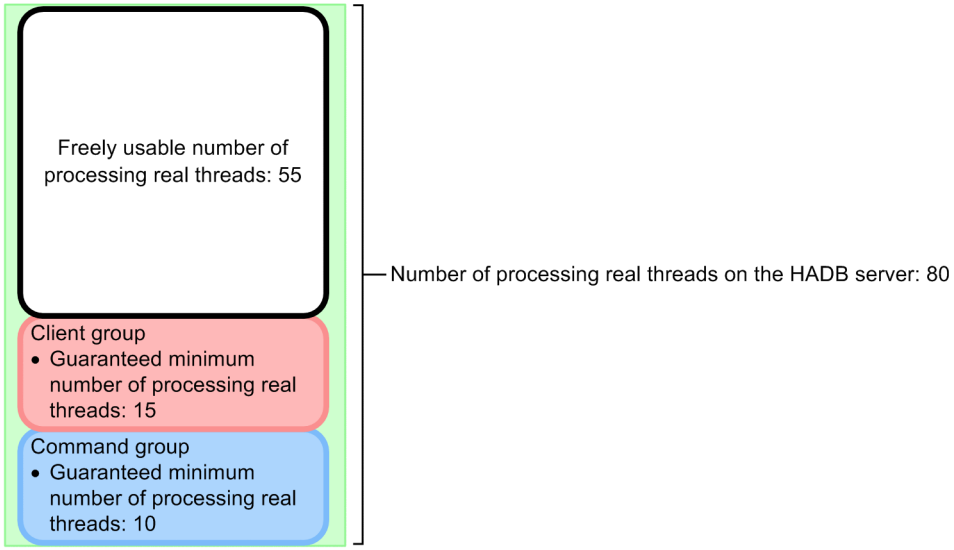
The following figure shows the relationships among the different types of processing real thread counts explained in (2) How to determine the numbers of processing real threads when the client-group facility is used when the client-group facility is used.

Figure 6-9: Relationships among processing real thread counts when the client-group facility is used

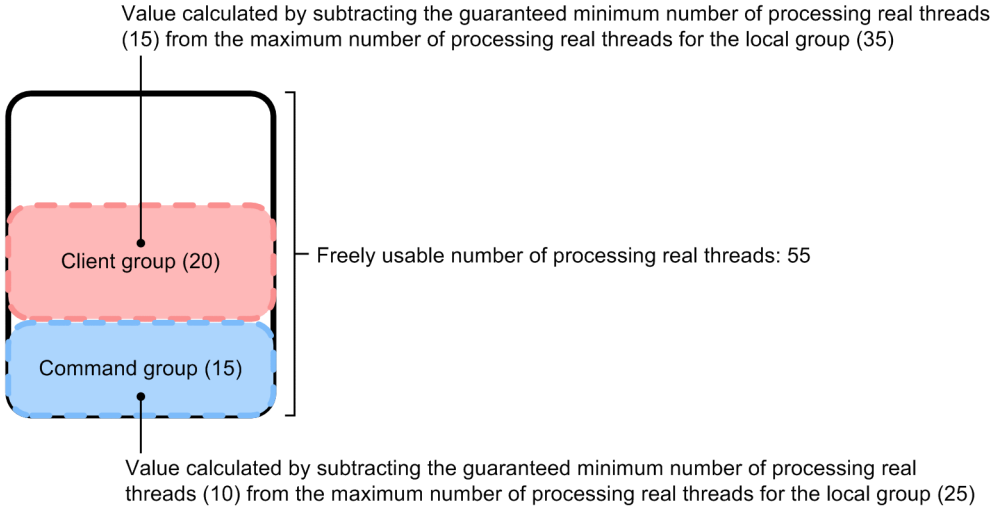
■ Settings for *number of processing real threads on the HADB server* and *maximum number and guaranteed minimum number of processing real threads for a group*



■ Relationship between *guaranteed minimum number of processing real threads usable by a group* and *freely usable number of processing real threads*



■ Relationship between *maximum number of processing real threads usable by a group* and *freely usable number of processing real threads*



Explanation:

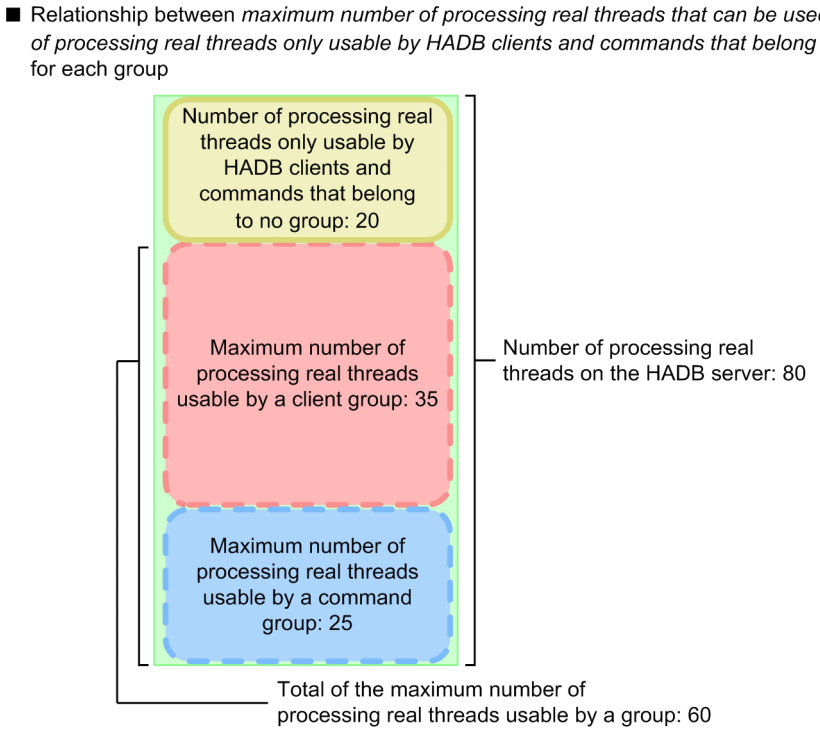
- The guaranteed minimum number of processing real threads that can be used by the client group (15) and the guaranteed minimum number of processing real threads that can be used by the command group (10) are allocated from the number of processing real threads on the HADB server (80). Therefore, the remaining processing real threads (55) belong to the freely usable processing real threads. The freely usable processing real threads are available to all HADB clients and commands regardless of whether they belong to a group.
- Of the freely usable processing real threads (55), each group can use up to its maximum number of processing real threads that can be used minus its guaranteed minimum number of processing real threads that can be used. In this example, the client group can use up to its local group's maximum number of processing real threads that

can be used (35) minus its guaranteed minimum number of processing real threads that can be used (15) as freely usable processing real threads (20). The command group can use up to its maximum number of processing real threads that can be used (25) minus its guaranteed minimum number of processing real threads that can be used (10) as freely usable processing real threads (15).

Note

Depending on the sum of the maximum numbers of processing real threads that can be used by all groups, a portion of the freely usable processing real threads might become available only to HADB clients and commands that belong to no group. The following figure shows the relationship between the number of processing real threads that are available only to HADB clients and commands that do not belong to any group, and the sum of the maximum numbers of processing real threads that can be used by all groups. This figure is based on the settings shown in Figure 6-9: Relationships among processing real thread counts when the client-group facility is used.

Figure 6-10: Relationship between the number of processing real threads that are available only to HADB clients and commands that belong to no group and the sum of the maximum numbers of processing real threads that can be used by all groups



Explanation:

If the sum of the maximum numbers of processing real threads that can be used by all groups is smaller than the number of processing real threads on the HADB server, some of the processing real threads will never be used by any group. These processing real threads become available only to HADB clients and commands that belong to no group.

In this example, the sum of the maximum numbers of processing real threads that can be used by all groups (60) is less than the number of processing real threads on the HADB server (80). As a result, 20 processing real threads will never be used by any group. These processing real threads (20) are made available only to HADB clients and commands that belong to no group.

6.24.3 Relationship between the number of connections and the number of processing real threads that are used by the client-group facility

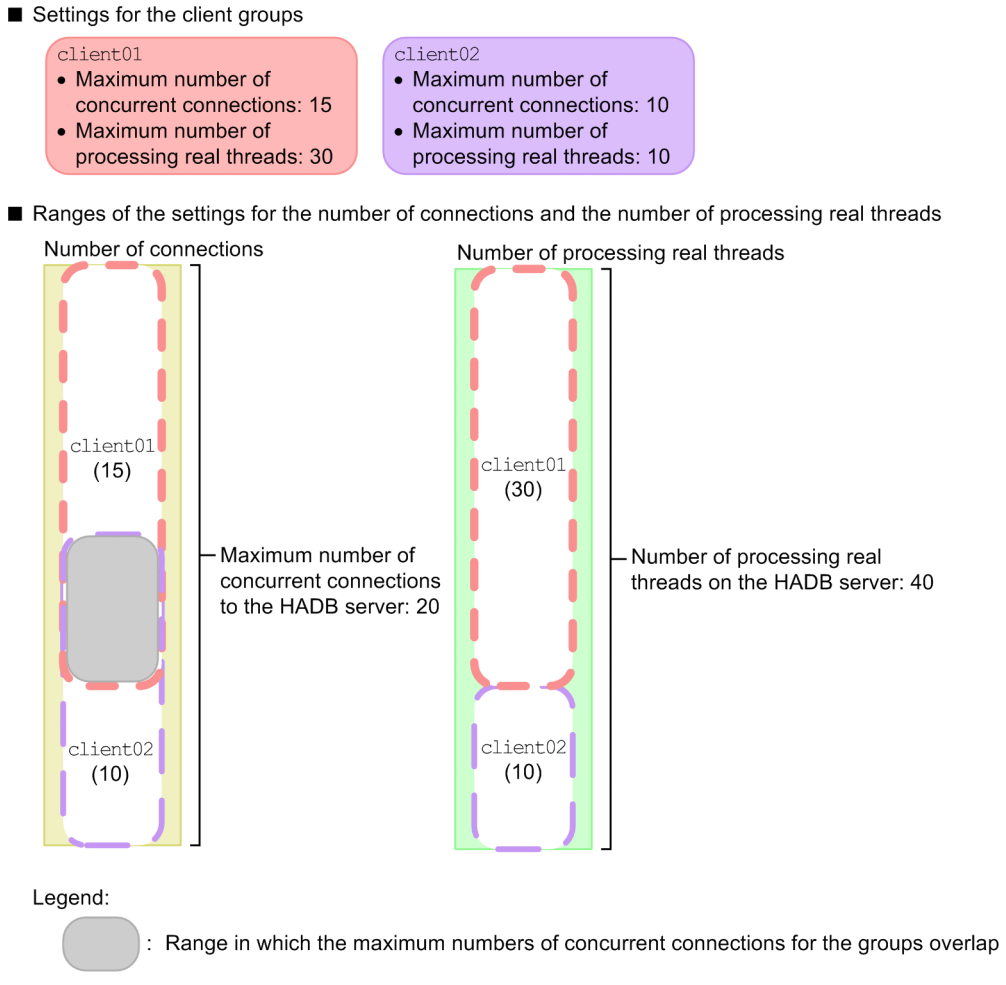
This subsection explains by way of examples the relationship between the number of connections and the number of processing real threads that are used by the client-group facility.

(1) Example 1: Specifying the maximum numbers of connections and processing real threads that can be used

This example uses the client-group facility and specifies the maximum number of concurrent connections for each group and the maximum number of processing real threads usable by each group.

By specifying the maximum number of concurrent connections for each group and the maximum number of processing real threads usable by each group, you can prevent a specific group from monopolizing connections and processing real threads.

Figure 6-11: Example where the maximum numbers of connections and processing real threads that can be used are specified



Explanation:

- Each group uses connections based on the maximum number of concurrent connections specified for that group. No group can use beyond the maximum number of concurrent connections that has been set for it. This prevents any one group from using all connections. The same applies to the number of processing real threads.

In this example, the maximum number of concurrent connections available to `client01` is 15, and the maximum number of processing real threads usable by this group is 30 (`client01` will not use all the available connections and processing real threads).

- As in this example, if one group's maximum number of concurrent connections partially overlaps another group's maximum number of concurrent connections, the actual number of connections in the overlapping part that are available to one group depends on the number of connections being used by the other group. For example, if client group `client01` is using 13 connections, 7 connections are available to client group `client02`.

  The same applies when the maximum number of processing real threads that can be used by one group overlaps the maximum number of processing real threads that can be used by another group.
- This example does not specify a guaranteed minimum number of concurrent connections for each client group (0 is specified as the guaranteed minimum number of concurrent connections). Therefore, while client groups `client01` and `client02` are not using connections, the HADB clients and commands that belong to no group can use the connections. While HADB clients and commands that belong to no group are using the connections, each group might not be able to use the maximum number of concurrent connections allocated to it. The same applies to the number of processing real threads.

The following shows an example specification of the client groups specified in this example.

■ Example specification of the `adbcltgrp` operand in the server definition

```
adbcltgrp -g client01 -m 15 -u 0 -r 30 -e 0
adbcltgrp -g client02 -m 10 -u 0 -r 10 -e 0
```

■ Example specification of the `adb_clt_group_name` operand in the client definition (that will belong to client group `client01`)

```
adb_clt_group_name = client01
```

■ Example specification of the `adb_clt_group_name` operand in the client definition (that will belong to client group `client02`)

```
adb_clt_group_name = client02
```

Note

- For details about the `adbcltgrp` operand in the server definition, see adbcltgrp (for setting a client group) or adbcltgrp (for setting a command group) in 7.2.12 Operands and options related to the client-group facility (command format).
- For details about the `adb_clt_group_name` operand in the client definition, see *Operands related to system configuration* in the *HADB Application Development Guide*.

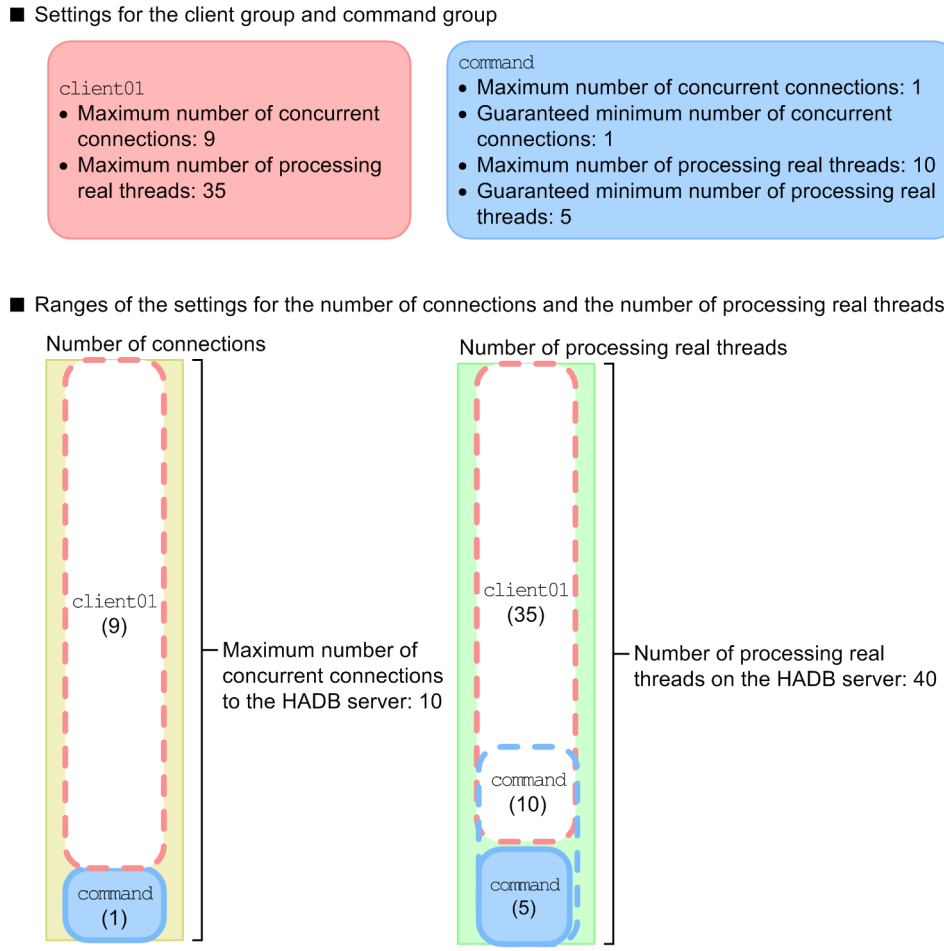
(2) Example 2: Specifying the guaranteed minimum numbers of connections and processing real threads

This example uses the client-group facility and specifies the maximum number of concurrent connections for each group, the guaranteed minimum number of concurrent connections for each group, the maximum number of processing real threads usable by each group, and the guaranteed minimum number of processing real threads usable by each group.

By specifying the maximum number of concurrent connections for each group and the maximum number of processing real threads usable by each group, you can prevent a specific group from monopolizing connections and processing real threads. In addition, if you specify a guaranteed minimum number of concurrent connections for each group and a

guaranteed minimum number of processing real threads usable by each group, the local group will not be affected by other HADB clients.

Figure 6-12: Example where the guaranteed minimum numbers of connections and processing real threads are specified



Explanation:

The `command` command group can always acquire one connection, in accordance with the specified maximum number of concurrent connections and the guaranteed minimum number of concurrent connections. Therefore, the HADB clients belonging to `command` will not be affected by the HADB clients that do not belong to `command`. Client group `client01` uses connections according to a specified maximum number of concurrent connections.

In addition, the `command` command group always allocates five processing real threads in accordance with the specified guaranteed minimum number of processing real threads. This example specifies 10 as the maximum number of processing real threads for `command`. Therefore, a maximum of five processing real threads are available to `command` if there are any processing real threads that are not in use by `client01`.

Note that if connections and processing real threads are not used by `command` and `client01`, the HADB clients that belong to no group can use a maximum of 9 connections and a maximum of 35 processing real threads.

The following shows an example specification of the client groups and command groups specified in this example.

■ Example specification of the `adbcltgrp` operand in the server definition

```
adbcltgrp -g client01 -m 9 -u 0 -r 35 -e 0
adbcltgrp -g command -m 1 -u 1 -r 10 -e 5
```

■ Example specification of the `adb_clt_group_name` operand in the client definition (that will belong to the `client01` client group)

```
adb_clt_group_name = client01
```

■ Example specification of the `adb_clt_group_name` operand in the client definition (that will belong to the `command` command group)

```
adb_clt_group_name = command
```

Note

- For details about the `adbcltgrp` operand in the server definition, see adbcltgrp (for setting a client group) or adbcltgrp (for setting a command group) in 7.2.12 Operands and options related to the client-group facility (command format).
- For details about the `adb_clt_group_name` operand in the client definition, see *Operands related to system configuration* in the *HADB Application Development Guide*.

6.25 Estimates related to work tables

This section explains how to estimate the following items:

- Number of pages in the global buffer for global work tables
- Number of pages in the buffer for local work tables
- Number of work tables created during retrieval using hash tables

6.25.1 Estimating the number of pages in the global buffer for global work tables

This section explains how to estimate the number of pages in the global buffer for global work tables.

You need to specify the value determined here for the `adb_dbbuff_wrktbl_glb_blk_num` operand in the server definition. For details about the `adb_dbbuff_wrktbl_glb_blk_num` operand, see the explanation of the adb_dbbuff_wrktbl_glb_blk_num operand in 7.2.2 Operands related to performance (set format).

The method of estimating the number of pages in the global buffer for global work tables follows.

Estimation method

1. Determine the information related to the columns comprising a work table.

   See *Work tables created when SQL statements are executed* under *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*. When coding SQL statements for which work tables are created, determine for the following cases the data types and data sizes of the columns comprising a work table and the number of rows stored in the work table:

   - When multiple table references are specified in the `FROM` clause
   - When a derived table is specified
   - When a viewed table is specified
   - When the `WITH` clause is specified
   - When a table-function derived table is specified
   - When a joined table is specified
   - When a subquery is specified
   - When a recursive query is specified

2. Determine the number of pages in the global buffer for global work tables.

   Use the formula in 5.9.1 Determining the total number of pages in the work table DB area to determine the number of pages in the global buffer for global work tables.

   Use the values determined in step 1 as the variables to be substituted in the formula.

If the number of pages in the work tables exceeds the size of the work table buffer, those work tables that do not fit will be output to the work table DB area file, resulting in processing performance degradation. For this reason, it is important to estimate accurately the size of the work table buffer.

6.25.2 Estimating the number of pages in the buffer for local work tables

This section explains how to estimate the number of pages in the buffer for local work tables. A local work table is a work table specific to each real thread and is created for each real thread.

Once you have determined the number of pages in the buffer for local work tables, specify a value that is equal to it or greater in the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition. For details, see the explanation of the adb_dbbuff_wrktbl_clt_blk_num operand in 7.2.2 Operands related to performance (set format).

As needed, also specify the following operand and option:

- `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition
  See *Operands related to performance* in the *HADB Application Development Guide*.
- Export option `adb_export_wrktbl_blk_num`
  See *Specification format for the adbexport command* under *adbexport (Export Data)* in the manual *HADB Command Reference*.

The formula for determining the number of pages in the buffer for local work tables follows.

Formula (pages)

$$\text{number of pages in the local work table buffer} = \text{MAX}\left\{ A,\ B \right\}$$

Explanation of variables

A: Number of work tables created when executing an SQL statement × 2

B: Value of the `adb_sys_uthd_num` operand in the server definition

Determine the number of work tables that are created when SQL statements are executed, see *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*. Note the following when determining the number of work tables:

- Determine the number of work tables to be created for each SQL statement based on the explanation in the example explained in *Number of work tables that are created* in the *HADB Application Development Guide*. From the resulting values, use the largest value for calculating the number of tables.
- If there is a subquery that includes an external reference column, determine the number of work tables to be created based on the explanation in the example explained in *Number of work tables that are created* in the *HADB Application Development Guide*. If the resulting value is designated *S*, use the following formula to calculate the number of work tables that will be created ultimately.

  Formula (work tables)

  $$\text{number of work tables to be created} = S \times MIN\left\{ T,\ U \right\}$$

  Explanation of variables

  T: Value of the `adb_sys_uthd_num` operand in the server definition

  U: Number of rows in the result before the evaluation of a subquery of a query specification that searches for columns that are referenced as external reference columns
- When executing a retrieval using a hash table, add the value determined from the following formula to the number of work tables to be created for executing SQL statements:

$2 \times$ *number of work tables created during retrieval using hash tables* $+ 1$

For details about how to determine the number of work tables created during retrieval using hash tables, see 6.25.3 Number of work tables created during retrieval using hash tables.

The following is a calculation example of the number of work tables to be created when executing an SQL statement:

Example

This example determines the number of work tables to be created when executing the SQL statement shown.

```
SELECT "C1","C2","C3" FROM "T1"
  WHERE "C1"=ANY(                                          ...1
                SELECT COUNT(DISTINCT "C3") FROM "T2"
                  WHERE "T1"."C1"="C2" GROUP BY "C1")      ...2
                  ORDER BY "C1","C2","C3"                  ...3
```

Explanation

1. Creates a work table for a quantified predicate. However, because this is a subquery that includes an external reference column (`"T1"."C1"`), use the following formula to determine the number of work tables to be created:

   $1 \times MIN\left\{Y, Z\right\}$

   Explanation of variables:
   Y: Value specified for the `adb_sys_uthd_num` operand in the server definition
   Z: Number of rows in table `T1`

   In this example, the "number of rows in the result before the evaluation of a subquery of a query specification that searches in columns that are referenced as external reference columns" is the number of rows obtained from `SELECT "C1","C2","C3" FROM "T1"`. Therefore, *Z* is the number of rows in table `T1`.

2. Creates a work table to be used for sort processing by the `GROUP BY` clause. However, because this is a subquery that includes an external reference column (`"T1"."C1"`), use the following formula to determine the number of work tables to be created:

   $1 \times MIN\left\{Y, Z\right\}$

   Explanation of variables:
   Y: Value specified for the `adb_sys_uthd_num` operand in the server definition
   Z: Number of rows in table `T1`

   In this example, the "number of rows in the result before the evaluation of a subquery of a query specification that searches in columns that are referenced as external reference columns" is the number of rows obtained from `SELECT "C1","C2","C3" FROM "T1"`. Therefore, *Z* is the number of rows in table `T1`.

3. Creates a work table to be used for sort processing by the `ORDER BY` clause. In this case, a single work table is created.

The number of work tables to be created when executing the above SQL statement is the combined total of steps 1 through 3.

6.25.3 Number of work tables created during retrieval using hash tables

This subsection explains how to determine the number of work tables created during retrieval using hash tables.

Retrieval using hash tables is performed when one of the following conditions is satisfied:

- When hash join is used to join tables
- When hash execution is used to process subqueries
- When global hash grouping is used for processing of grouping or the `DISTINCT` set function
- When hash execution is used as a method for processing `SELECT DISTINCT`
- When hash execution is used as a method for processing the set operation

A hash table area is reserved and hash join processing is performed on the basis of the value specified for the `adb_sql_exe_hashtbl_area_size` operand in the server definition and client definition. A work table is created when a shortage occurs in the reserved hash table area during this processing.

Use the following formula to determine the number of work tables created during retrieval using hash tables for each SQL statement to be executed.

Formula

Number of work tables created during retrieval using hash tables =

$$MAX(\uparrow((A + 32) \times B \times 2) / D\uparrow, 121) \times C$$

Explanation of variables

A: Maximum row length in hash tables (bytes)

Determine the row length in hash tables for each of the elements described later in each query expression body. The maximum value among the determined values becomes the maximum row length in the hash tables.

- Query specification that uses the equal sign (=) to join multiple tables

  For each table joined by the equal sign (=), determine the sum total of the lengths of all columns specified in selection expressions and search conditions.

  Note that if value expressions that include a column specification are specified on the left and right sides of the = join condition, add the data lengths of the results of those value expressions to the sum total.

  The sum total of all values determined for the tables, excluding the minimum value, becomes the row length in the hash tables.

  However, if only two tables are joined using the equal sign (=) and the results of the value expressions specified on the left and right sides have different data types or lengths, determine the total of all values. In this case, do not exclude the minimum value.

  The hash table row length can be determined from the access path display result. For a tree query for which the table joining method is `HASH JOIN`, perform the above to determine the hash table row length.

- Table subquery of a quantified predicate

  The sum total of the following four values becomes the row length in the hash tables:

  · Sum total of the data lengths of the results of subquery selection expressions

  · Sum total of the data lengths of the results of the set functions specified in a subquery

  · Sum total of the lengths of all columns specified in a predicate that includes an external reference column within the search condition

· If the result of the table subquery selection expression on the right side of the quantified predicate or IN predicate and the column specified on the left side of that predicate have different data types or data lengths, the length of the column specified on the left side of the predicate

The hash table row length can be determined from the access path display result. For a tree query for which the subquery processing is SUBQUERY HASH, the sum of the preceding four values determines the hash table row length.

- Table subquery in the IN predicate

  For details about how to determine the row length in hash tables, see *Table subquery of a quantified predicate* above.

- Subquery that uses the equal sign (=) to specify an external reference column within the search condition, other than a table subquery of a quantified predicate or a table subquery in the IN predicate

  For details about how to determine the row length in hash tables, see *Table subquery of a quantified predicate* above.

- Query specification that includes the GROUP BY clause

  Determine the column length of the grouping column and the data length of the result of the set function. The sum total of these results becomes the row length in the hash tables.

  The hash table row length can be determined from the access path display result. Determine the column length of the grouping column and the data length of the set function result for a tree query for which the grouping method is GLOBAL HASH GROUPING. The sum total of these results becomes the row length in the hash tables.

- Query specification that includes the DISTINCT set function

  Determine the following values: The column length of the grouping column, the data length of the result of the value expression specified for the argument of the DISTINCT set function, and the data length of the results of set functions (excluding the DISTINCT set function). The sum total of these results becomes the row length in the hash tables.

- Query specification that includes SELECT DISTINCT

  The total data length of the selection expressions in SELECT DISTINCT becomes the row length of hash tables.

  The hash table row length can be determined from the access path display result. Determine the data length of the selection expression in SELECT DISTINCT for each tree query for which hash execution is the method for processing SELECT DISTINCT. The sum total of these results becomes the row length in the hash tables.

- Query expression body in which UNION or UNION DISTINCT is specified

  Determine the column length of the table that is derived by the query expression body. The sum total of these results becomes the row length in the hash tables.

  The hash table row length can be determined from the access path display result. Determine the column length of the table that is derived by the query expression body for each tree query for which hash execution is the method for processing the set operation. The sum total of these results becomes the row length in the hash tables.

B: Maximum number of intermediate retrieval results

Use the following formula to determine its value:

MAX (*A*, *B*)

A: *Maximum number of provisional intermediate retrieval results*
B: 2,048 × (Value specified for the adb_sql_exe_max_rthd_num operand in the client definition)
× (Value specified for the adb_sql_exe_max_rthd_num operand in the client definition)

Determine the maximum number of provisional intermediate retrieval results for each of the elements described later in each query expression body. The largest number of the determined maximum numbers is the maximum number of provisional intermediate retrieval results.

- Query specification that uses the equal sign (=) to join multiple tables

  For both sides of each equal sign (=), determine the number of retrieval results, and then select the smaller value. Of the values determined for all joins, use the largest as the maximum number of intermediate retrieval results.

  The number of intermediate retrieval results can be determined from the access path display result. Determine the number of results for the outer table for a tree query for which the table joining method is `HASH JOIN`. From the determined numbers of results, the largest number of results is the number of intermediate retrieval results.

- Table subquery of a quantified predicate

  The number of rows of results for the subquery is the number of intermediate retrieval results.

  The number of intermediate retrieval results can be determined from the access path display result. Determine the number of subquery results for a tree query for which subquery processing is `SUBQUERY HASH`. From the determined numbers of results, the largest number of results is the number of intermediate retrieval results.

- Table subquery in the `IN` predicate

  For details about how to determine the number of intermediate retrieval results, see *Table subquery of a quantified predicate*.

- Subquery that uses the equal sign (=) to specify an external reference column inside the search condition, other than a table subquery of a quantified predicate or a table subquery in the `IN` predicate

  For details about how to determine the number of intermediate retrieval results, see *Table subquery of a quantified predicate*.

- Query specification that includes the `GROUP BY` clause

  The number of results after grouping is used becomes the number of intermediate retrieval results.

  The number of intermediate retrieval results can be determined from the access path display result. Determine the number of results after performing grouping for a tree query for which the grouping method is `GLOBAL HASH GROUPING`. From the determined numbers of results, the largest number of results is the number of intermediate retrieval results.

- Query specification that includes the `DISTINCT` set function

  The number of results after duplication is eliminated becomes the number of intermediate retrieval results. If you specify the `GROUP BY` clause in a query specification, assume that the argument of the `DISTINCT` set function is also a grouping column. Then, use the number of results obtained after grouping as the approximate number of intermediate retrieval results.

- Query specification that includes `SELECT DISTINCT`

  The number of results after duplication is eliminated becomes the number of intermediate retrieval results.

- Query expression body in which `UNION` or `UNION DISTINCT` is specified

  The number of results after the set operation is performed becomes the number of intermediate retrieval results.

C: Number of hash tables

Use the following formula to determine its value:

Number of hash tables =

{ (Number of tables joined using equal signs (=)) - (number of query specifications joined using equal signs (=)) }

+ (number of quantified predicates)
+ (number of `IN` predicates for which a subquery is specified on the right side)
+ (number of subqueries for which external reference columns are specified using equal signs (=) in the retrieval conditions, except the subqueries in quantified predicates and in the `IN` predicates for which a subquery is specified on the right side)
+ (number of specified `GROUP BY` clauses)
+ (number of specified `DISTINCT` set functions)
+ (number of `FULL OUTER JOIN` specifications)
+ (number of `SELECT DISTINCT` specifications)
+ (number of `UNION` or `UNION DISTINCT` specifications)

The number of hash tables can also be determined from the access path display result. If you can display the access path, determine the number of hash tables from the access path display result and the following formula:

Number of hash tables =

Number of times `HASH JOIN` (table joining method)# is displayed
+ number of times `SUBQUERY HASH` (table joining method)# is displayed
+ number of times `GLOBAL HASH GROUPING` (table joining method)# is displayed
+ number of times `GLOBAL HASH UNIQUE` (table joining method)# is displayed

#

Determine the number by using the information displayed in the <<Tree View>> of the access path display result. For example, if `HASH JOIN` appears twice in the <<Tree View>>, calculate the number of `HASH JOIN`s (table joining method) displayed as 2.

D: Hash table area

Use the following formula to determine its value:

hash table area =

value specified for the adb_sql_exe_hashtbl_area_size operand × 1,024 × 1,024

If the value specified for the `adb_sql_exe_hashtbl_area_size` operand in the client definition is smaller than the value specified for the `adb_sql_exe_hashtbl_area_size` operand in the server definition, substitute the value specified in the client definition. Otherwise, substitute the value specified in the server definition.

If `0` is specified for the `adb_sql_exe_hashtbl_area_size` operand, retrieval using hash tables is not applied.

7 Designing the Server Definition

This chapter explains the format for specifying operands in a server definition, the values that can be specified, and the syntax rules that apply.

7.1 Specification formats for server definition operands

This section explains the format for specifying each operand in the server definition.

For details about the contents of each operand in the server definition, see 7.2 Detailed descriptions of the server definition operands.

For details about how to create and modify the server definition, see 8.5 Creating and modifying a server definition.

The server definition is written in the set format or the command format. Before specifying each operand in the server definition, identify the format in which the server definition is written. For details about the set and command formats, see (2) Description format in 7.3.1 Details of syntax rules for the server definition.

> **Note**
>
> For details about the operands in the client definition, see *Designing Client Definitions* in the *HADB Application Development Guide*.

The table below lists the formats for specifying server definition operands.

Any operand that is highlighted in color is a mandatory operand. The numbers in the *No.* column in the table correspond to the numbers assigned to the individual operands.

No.	Operand or option specification format	Operand classification
1	**set adb_db_path** = ***DB-directory-name***	System configuration
2	[set adb_rpc_port = *HADB-server-port-number*]	
3	[set adb_sys_max_users = *maximum-number-of-concurrent-connections*]	
4	[set adb_core_path = *name-of-output-destination-directory-for-error-information-(core-files)*]	
5	[set adb_blk_path_wrk = *path-name-of-block-special-file-of-work-table-DB-area-file*]	
6	[set adb_dbarea_wrk_page_size = *page-size-of-work-table-DB-area*]	
7	[set adb_sys_rthd_num = *number-of-processing-real-threads*]	Performance
8	[set adb_sys_uthd_num = *number-of-pseudo-threads-that-can-be-generated-in-a-real-thread*]	
9	[set adb_sys_memory_limit = *maximum-size-of-memory-used-by-the-HADB-server*]	
10	[set adb_sql_exe_max_rthd_num = *maximum-number-of-SQL-processing-real-threads*]	
11	[set adb_sql_exe_hashgrp_area_size = *hash-group-area-size*]	
12	[set adb_sql_exe_hashtbl_area_size = *hash-table-area-size*]	
13	[set adb_sql_exe_hashflt_area_size = *hash-filter-area-size*]	
14	[set adb_sql_opt_drvtbl_grping_prior = {LOCAL|GLOBAL}]	
15	[set adb_sys_rthd_area_max = *maximum-size-of-real-thread-private-memory-that-can-be-allocated*]	
16	[set adb_sys_proc_area_max = *maximum-size-of-process-common-memory-that-can-be-allocated*]	
17	[set adb_sys_shm_huge_page_size = *multiple-of-the-single-page-size-in-HugePages*]	
18	[set adb_dbbuff_wrktbl_glb_blk_num = *number-of-pages-in-global-buffer-for-global-work-tables*]	

No.	Operand or option specification format	Operand classification
19	[set adb_dbbuff_wrktbl_clt_blk_num = *number-of-pages-in-buffer-for-local-work-tables*]	
20	[set adb_sql_tbldef_cache_size = *table-definition-pool-size*]	
21	[set adb_log_usrfile_num = *number-of-user-log-files*]	System logs
22	[set adb_log_usrfile_size = *initial-size-of-user-log-file*[,*threshold-where-user-log-file-size-is-reduced*]]	
23	[set adb_log_usrbuf_num = *number-of-user-log-buffers*]	
24	[set adb_log_rec_msg_interval = *output-interval-of-message-showing-database-recovery-processing-time-elapsed*]	
25	[set adb_sys_max_users_wrn_pnt = *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections*[,*reset-threshold-for-warning-message-output*]]	Status monitoring
26	[set adb_sys_memory_limit_wrn_pnt = *output-threshold-for-warning-message-regarding-HADB-server-memory-usage*[,*reset-threshold-for-warning-message-output*]]	
27	[set adb_rpc_wait_time = *upper-limit-for-wait-time*]	
28	[set adb_rpc_tcp_keepalive_time = *no-communication-time-after-which-KeepAlive-probe-starts*]	
29	[set adb_sql_prep_delrsvd_words = *reserved-word-to-be-unregistered*[,*reserved-word-to-be-unregistered*]...]	SQL statements
30	[set adb_sys_trn_iso_lv = {READ_COMMITTED | REPEATABLE_READ}]	
31	[set adb_sql_text_out = {Y | N}]	
32	[set adb_sql_trc_out = {Y | N}]	
33	[set adb_sql_trc_param = {Y | N}]	
34	[set adb_sql_trc_accesspath = {Y | N}]	
35	[set adb_sql_trc_level = {SQL | CALL}]	
36	[set adb_sql_trc_txtfile_size = *maximum-capacity-of-an-SQL-trace-file*]	
37	[set adb_sql_order_mode = {BYTE | ISO}]	
38	[set adb_sql_prep_dec_div_rs_prior = {INTEGRAL_PART | FRACTIONAL_PART}]	
39	[set adb_sql_default_dbarea_shared = *default-name-of-data-DB-area-that-stores-tables-or-indexes*]	
40	[set adb_sql_rngidx_preread ={ALL|NO|*range-index-name*[,*range-index-name*]...}]	Range indexes
41	[set adb_sta_log_max_size = *maximum-size-of-statistics-log-file*]	Statistical information
42	[set adb_sta_log_path = *name-of-output-destination-directory-for-statistics-log-files*]	
43	[set adb_sta_log_size_unit = {G|M}]	
44	[set adb_syndict_storage_path = *storage-directory-for-synonym-dictionary-files*]	Synonym search
45	[set adb_syndict_node_storage_path = *multi-node-synonym-dictionary-storage-directory*]	
46	[set adb_audit_log_path = *audit-trail-directory*]	Audit trail facility
47	[set adb_audit_log_max_size = *maximum-size-of-audit-trail-file*]	
48	[set adb_audit_log_max_num = *maximum-number-of-audit-trail-file-generations*]	

No.	Operand or option specification format	Operand classification
49	[set adb_sys_multi_node_info = *node-host-name* [:*node-port-number*], *node-host-name* [:*node-port-number*] [,*node-host-name* [:*node-port-number*]] ...]	Multi-node function
50	[set adb_sys_node_start_wait_time = *upper-limit-for-time-to-wait-for-startup-completion-of-other-nodes*]	
51	{{ [adbbuff -g *global-buffer-name* {-n *DB-area-name* [,*DB-area-name*] ... | -o } [-p *number-of-pages-in-global-buffer*] [-a *number-of-pages-in-global-buffer-dedicated-to-range-indexes*] [-v *memory-size-used-for-table-scan-buffer* [,*maximum-memory-size-used-per-real-thread*]] [-k *number-of-sectors-in-table-scan-buffer-for-batch-loading-segments*]] }}	Global buffer
52	{{ [adbcltgrp -g *client-group-name* [-m *maximum-number-of-concurrent-connections-for-the-specified-client-group*] [-u *guaranteed-minimum-number-of-concurrent-connections-for-the-specified-client-group*] [-r *maximum-number-of-processing-real-threads-usable-by-the-specified-client-group*] [-e *guaranteed-minimum-number-of-processing-real-threads-usable-by-the-specified-client-group*] [-w *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-the-specified-client-group* [,*reset-threshold-for-warning-message-output*]]] }}	Client-group facility
53	[adbcltgrp -g command [-m *maximum-number-of-concurrent-connections-for-the-specified-command-group*] [-u *guaranteed-minimum-number-of-concurrent-connections-for-the-specified-command-group*] [-r *maximum-number-of-processing-real-threads-usable-by-the-specified-command-group*] [-e *guaranteed-minimum-number-of-processing-real-threads-usable-by-the-specified-command-group*] [-w *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-the-specified-command-group* [,*reset-threshold-for-warning-message-output*]]]	

7.2 Detailed descriptions of the server definition operands

This section provides detailed descriptions of the server definition operands.

For details about the specification format of each operand in the server definition, see 7.1 Specification formats for server definition operands.

For details about how to create and modify the server definition, see 8.5 Creating and modifying a server definition.

The server definition is written in the set format or the command format. Before specifying each operand in the server definition, identify the format in which the server definition is written. For details about the set and command formats, see (2) Description format in 7.3.1 Details of syntax rules for the server definition.

> **Note**
>
> For details about the operands in the client definition, see *Designing Client Definitions* in the *HADB Application Development Guide*.

See the following subsections for detailed descriptions of each operand in the server definition.

7.2.1 Operands related to system configuration (set format)

[1] `adb_db_path` = *DB-directory-name*

~<path name> ((2 to 510 bytes))

Specify an absolute path name for the DB directory name. This operand must be specified. The DB directory name specified here must match the DB directory name specified using the `adbinit` command.

The following rules apply to the path name specified in this operand:

- The root directory cannot be specified.
- Specify a directory that is different from the server directory (including subdirectories).
- Specify a path name consisting of only alphanumeric characters, underscore (_), hyphen (−), hash mark (#), at sign (@), and forward slash (/).

[2] `adb_rpc_port` = *HADB-server-port-number*

~<integer> ((5,001 to 65,535)) <<23,650>>

Specify the HADB server port number that will be used to communicate between the HADB server and an HADB client.

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.

[3] `adb_sys_max_users` = *maximum-number-of-concurrent-connections*

~<integer> ((1 to 1,024)) <<10>>

Specify the maximum number of HADB clients that can concurrently connect to the HADB server.

If you will be using the client-group facility, set this operand to a value that is greater than the sum of the guaranteed minimum numbers of concurrent connections for all groups. If the sum of the guaranteed minimum numbers of concurrent connections for all groups is greater than the value set for this operand, an error will occur when the HADB server starts.

For details about the guaranteed minimum number of concurrent connections for each group, see the following explanations of the -u option of the adbcltgrp operand in 7.2.12 Operands and options related to the client-group facility (command format):

- -u *guaranteed-minimum-number-of-concurrent-connections-for-the-specified-client-group*
- -u *guaranteed-minimum-number-of-concurrent-connections-for-the-specified-command-group*

Multi-node function

- If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.
- In a multi-node configuration, the value specified for this operand becomes the maximum number of concurrent connections allowed for the HADB servers.

[4] adb_core_path = *name-of-output-destination-directory-for-error-information-(core-files)*

~<path name> ((2 to 510 bytes))

Specify an absolute path for the output destination directory for error information (core files). Error information (core files) is output to the directory specified by this operand.

Normally, this operand does not need to be specified. When this operand is omitted, error information (core files) is output to the $DBDIR/SPOOL directory.

You specify this operand to output error information (core files) to a directory other than the $DBDIR/SPOOL directory. Do not specify the conf directory ($ADBDIR/conf) under the server directory or its subdirectories.

> **Important**
>
> When you specify this operand, specify a directory for which the HADB administrator has access privileges (read, write, and execution privileges).

[5] adb_blk_path_wrk = *path-name-of-block-special-file-of-work-table-DB-area-file*

~<path name> ((2 to 255 bytes))

Specify a path name for the block special file to be allocated to the work table DB area file when the HADB server starts.

The work table DB area file is changed according to the value specified for this operand and the status of the existing work table DB area file (ADBWRK).

If you change the existing work table DB area file, after making the change, initialize the file. If you are not changing the existing work table DB area file, check its content, and initialize it if it has not been initialized. The following table shows the details.

Table 7-1: Relationship between the adb_blk_path_wrk operand and the work table DB area file

Specification of the adb_blk_path_wrk operand in the server definition	Existing work table DB area file		
	None	Regular file	Block special file
Specification is omitted.	A regular file is created.	Unchanged (The existing regular file remains as is.)	Unchanged (The existing block special file remains as is.)
A path name is specified.	Uses the block special file specified in the adb_blk_path_wrk operand.		Uses the block special file specified in the adb_blk_path_wrk operand. If the same name as an existing block special file is specified, no change is made.

Multi-node function

If you are using the multi-node function and if the work table DB area file is not unique to one of the nodes, take one of the following steps:

- For this operand, specify a block special file that is not shared with other nodes.
- Alternatively, delete the existing work table DB area file and omit this operand.

[6] `adb_dbarea_wrk_page_size` = *page-size-of-work-table-DB-area*

~<integer> ((32 to 32,768)) (kilobytes)

When you want to change the page size of the work table DB area that was specified when the `adbinit` command was executed, specify the desired page size. If you specify a value that is not divisible by 32, the specified value is rounded up to the next multiple of 32 kilobytes.

Normally, this operand does not need to be specified. If this operand is omitted, the page size that was specified when the `adbinit` command was executed is applied.

Increasing the page size might cause memory to be insufficient. Therefore, before you increase the page size by using this operand, review the memory requirements. For details about how to estimate the memory requirements, see 6.3.3 Determining the memory requirement for starting the HADB server and 6.3.4 Determining the memory requirement during normal operation.

You can use the following commands to check the page size that is currently applied on the active HADB server:

- `adbls -d gbuf` command

  For details about this command, see *adbls -d gbuf (Display Global Buffer Information)* in the manual *HADB Command Reference*.

- `adbls -d lbuf` command

  For details about this command, see *adbls -d lbuf (Display Local Work Table Buffer Information)* in the manual *HADB Command Reference*.

The page size of the work table DB area that was specified when the `adbinit` command was executed can be checked by searching the dictionary table. For details about searching a dictionary table, see (37) Checking the page size of the work table DB area specified when the adbinit command was executed in B.22 Searching a dictionary table.

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.

7.2.2 Operands related to performance (set format)

[7] `adb_sys_rthd_num` = *number-of-processing-real-threads*

~<integer> ((1 to 4,096)) <<40>>

Specify the number of processing real threads.

A processing real thread is a real thread used by the HADB server when SQL statements and the following targeted commands are executed:

■ Targeted commands

- `adbimport` command
- `adbgetcst` command
- `adbidxrebuild` command
- `adbexport` command

- adbmergechunk command
- adbarchivechunk command
- adbunarchivechunk command
- adbreorgsystemdata command

Specify the value determined using the following formula for this operand. The same applies when the client-group facility is used.

Formula

```
A × B × C
```

Explanation of variables

A: Number of connections that execute SQL statements concurrently

B: Value of the adb_sql_exe_max_rthd_num operand

C: Maximum number of SELECT statements to be concurrently executed in a single connection

When executing these targeted commands without using the client-group facility, specify a value in this operand that is at least the *minimum number of processing real threads required* to execute the command. If you do not specify a value at least equal to the *minimum number of processing real threads required*, execution of the command might result in an error. For details about the *minimum number of processing real threads required* to execute each command, see Table 6-28: Operands and command options for specifying the number of processing real threads to be used for command execution in (2) Operands and command options for specifying the number of processing real threads to be used for command execution under 6.23.2 Points to consider about the number of processing real threads to be used during command execution.

If you will be using the client-group facility, set this operand to a value that is greater than the sum of the guaranteed minimum numbers of processing real threads usable by all groups. If the sum of the guaranteed minimum numbers of processing real threads usable by all groups is greater than the value set for this operand, an error will occur when the HADB server starts.

For details about the guaranteed minimum number of processing real threads usable by each group, see the following explanations of the -e option of the adbcltgrp operand in 7.2.12 Operands and options related to the client-group facility (command format):

- -e *guaranteed-minimum-number-of-processing-real-threads-usable-by-the-specified-client-group*
- -e *guaranteed-minimum-number-of-processing-real-threads-usable-by-the-specified-command-group*

> **Important**
>
> This operand has a major effect on memory requirements. Whenever you change this operand's value, re-estimate the memory requirements. For details about how to estimate the memory requirements, see 6.3 Estimating the HADB server's memory requirement.

[8] adb_sys_uthd_num = *number-of-pseudo-threads-that-can-be-generated-in-a-real-thread*

~<integer> ((0 to 4,096))<<128>>

Specify the number of pseudo threads that can be generated in a real thread.

As a rule, specify the default value for this operand.

> **Important**
>
> When 0 is specified in this operand, the HADB server will execute SQL statements without using out-of-order execution. The HADB server will use only connection threads to execute SQL statements.

[9] `adb_sys_memory_limit` = *maximum-size-of-memory-used-by-the-HADB-server*

~<integer suffixed by the unit> ((5,120MB to 1,099,511,627,776MB))

Specifies the maximum size of the memory that can be used by the HADB server (shared memory and process memory) with an integer suffixed by the unit.

> **Important**
>
> Specify this operand in the case of an HADB server of version **03-01** or later. This operand and the `adb_sys_proc_area_max` and `adb_sys_rthd_area_max` operands in the server definition are mutually exclusive. When you specify this operand, do not specify the `adb_sys_proc_area_max` operand or the `adb_sys_rthd_area_max` operand in the server definition.

■ **Guideline for determining the value to be specified for the adb_sys_memory_limit operand**

Specify in this operand a value that is equal to or greater than the value determined for *Maximum memory capacity used by the HADB server* in 6.3 Estimating the HADB server's memory requirement. Alternatively, specify the maximum size of the memory that can be used by the HADB server.

Note that the value specified for the `adb_sys_memory_limit` operand in the server definition must satisfy the following formula if the `adb_sys_shm_huge_page_size` operand in the server definition is set to a value other than `0`. This is the case where `HugePages` is used for the shared memory used by the HADB server. Specifying a value that does not satisfy the following formula causes an error to occur when the HADB server starts.

```
value-specified-for-adb_sys_memory_limit-operand-in-server-definition
   ≤ kernel-parameter-vm.nr_hugepages × single-page-size-in-HugePages
```

For details about how to check the value of *single-page-size-in-HugePages*, see the explanation of the `adb_sys_shm_huge_page_size` operand in the server definition.

■ **About the unit of the value specified for the adb_sys_memory_limit operand**

- The unit that can be suffixed to the value of this operand is `MB` (megabyte), `GB` (gigabyte), or `TB` (terabyte).

  Examples of specifying a value with a unit

  ```
  set adb_sys_memory_limit = 196608MB
  ```

  ```
  set adb_sys_memory_limit = 128GB
  ```

  ```
  set adb_sys_memory_limit = 64TB
  ```

- If you omit specifying a unit (if you specify only an integer), `MB` (megabyte) is assumed.

  Example of specifying a value without a unit

  In the following case, the system assumes that `196608MB` was specified:

  ```
  set adb_sys_memory_limit = 196608
  ```

- The following shows the range of specifiable values for each unit:

  For MB (megabyte)

  `5,120MB` to `1,099,511,627,776MB`

  For GB (gigabyte)

  `5GB` to `1,073,741,824GB`

  For TB (terabyte)

  `1TB` to `1,048,576TB`

■ **Value assumed if the adb_sys_memory_limit operand is not specified**

If this operand is omitted, the maximum size of process common memory is determined based on the value of the `adb_sys_proc_area_max` operand in the server definition. The maximum size of real thread private memory per real thread is determined based on the value of the `adb_sys_rthd_area_max` operand in the server definition.

When this operand is specified, the maximum amount of memory (shared memory and process memory) that can be used by the HADB server is controlled as shown in the following table.

Table 7-2: List of memory areas controlled by the adb_sys_memory_limit operand

No.	Memory area used by the HADB server			Related server definition operand
1	Shared memory	Shared memory management area		--
2		Process common memory#1	Memory for HADB server control	--
3			Other (including memory used for executing SQL statements and commands)	--
4		Real thread private memory#2	Local work table buffer	`adb_dbbuff_wrktbl_clt_blk_num`
5			Hash grouping area	`adb_sql_exe_hashgrp_area_size`
6			Hash table area	`adb_sql_exe_hashtbl_area_size`
7			Hash filter area	`adb_sql_exe_hashflt_area_size`
8			Other (including work memory used for executing SQL statements and memory used for executing commands)	--
9		Global buffer page	Global buffer	`adbbuff -p`
10			Buffer for range indexes	`adbbuff -a`
11			Buffer for table scan	• `adbbuff -v` • `adbbuff -k`
12			Global buffer for global work tables	`adb_dbbuff_wrktbl_glb_blk_num`
13	Process memory	Heap memory		--

Legend:

--: There is no related server definition operand.

#1

The `adb_sys_proc_area_max` operand in the server definition is related.

#2

The `adb_sys_rthd_area_max` operand in the server definition is related.

[10] `adb_sql_exe_max_rthd_num` = *maximum-number-of-SQL-processing-real-threads*

~<integer> ((0 to 4,096)) <<4>>

Specify the maximum number of SQL processing real threads.

An SQL processing real thread is a processing real thread used when executing SQL statements.

For this operand, specify a value equal to or less than the number of CPU cores in the machine on which the HADB server is installed. When you determine the value to be specified, see the following ▪ *Consideration on specifying the adb_sql_exe_max_rthd_num operand*. Also consider the number of SQL statements that are concurrently executed.

Important

If you change the value specified for this operand, also reassess the value specified for the `adb_sys_rthd_num` operand. If you specify a value greater than the value of the `adb_sys_rthd_num` operand for this operand, the value of the `adb_sys_rthd_num` operand is used.

▪ Consideration on specifying the adb_sql_exe_max_rthd_num operand

If the following condition is satisfied, it might not be possible to use the specified number of processing real threads when executing SQL statements. In such a case, the SQL statements wait until the number of processing real threads specified by this operand can be allocated. The formula to be used depends on whether the client-group facility is used.

Formula (when the client-group facility is not used)

$A < B \times C \times D$

Formula (when the client-group facility is used)

$E < F \times C \times D$

Note: Among the client groups and command groups that were set, groups that satisfy the formula become the targets.

Explanation of variables

A: Value of the `adb_sys_rthd_num` operand

B: Value of the `adb_sys_max_users` operand

C: Value of the `adb_sql_exe_max_rthd_num` operand

D: Maximum number of `SELECT` statements to be concurrently executed in a single connection (if the resulting value is 0, use 1)

E: Guaranteed minimum number of processing real threads usable by the specified client group or command group

F: Maximum number of concurrent connections for the specified client group or command group

When this operand and the client definition `adb_sql_exe_max_rthd_num` operand are both specified, the value specified by the client definition `adb_sql_exe_max_rthd_num` operand is prioritized. However, if the value specified by the client definition `adb_sql_exe_max_rthd_num` operand is greater than the value specified by this operand, the former is not prioritized, and the value specified by this operand is prioritized.

For details about the `adb_sql_exe_max_rthd_num` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

If the value of the `adb_sql_exe_max_rthd_num` operand is greater than the maximum number of processing real threads usable by any of the specified client groups and command groups, the maximum number of processing real threads that can be used is assumed to be the same as the value of the `adb_sql_exe_max_rthd_num` operand for that group.

For details about the maximum number of processing real threads that can be used, see the description of the `-r` option of the `adbcltgrp` operand in 7.2.12 Operands and options related to the client-group facility (command format).

The following table shows the relationship between this operand's value and the number of SQL processing real threads used during execution of SQL statements.

Table 7-3: Number of SQL processing real threads used during execution of SQL statements

No.	SQL statement to be executed		adb_sql_exe_max_rthd_num operand's value		
			0 specified	1 specified	2 or a greater value specified
1	Definition SQL statements	ALTER TABLE statement (for changing an archivable multi-chunk table to a regular multi-chunk table)#1	The number of SQL processing real threads that are used is 0.	The number of SQL processing real threads that are used is 0.	The number of SQL processing real threads that are used is the same as the value of the adb_sql_exe_max_rthd_num operand.
2		DROP INDEX statement#1			
3		DROP SCHEMA statement#1			
4		DROP TABLE statement#1			
5		DROP USER statement#1			
6		SCHEMA of the REVOKE statement (revoking schema operation privileges)#1			
7	Data manipulation SQL statements	PURGE CHUNK statement#1			The number of SQL processing real threads that are used is the value determined by the formula shown below.#2
8		TRUNCATE TABLE statement#1			
9		SELECT statement#3		The number of SQL processing real threads that are used is 1.	The number of SQL processing real threads that are used is the same as the value of the adb_sql_exe_max_rthd_num operand.
10	Other SQL statements#4			The number of SQL processing real threads that are used is 0.	The number of SQL processing real threads that are used is 0.

#1

If a value of 1 or less is specified in this operand, only connection threads are used to execute SQL statements. If a value of 2 or greater is specified in this operand, the number of SQL processing real threads indicated in the table is used to execute SQL statements when segments are released.

#2

The number of SQL processing real threads that are used is determined by the following formula:

MIN(*A*, *B*)

A: Value specified for the adb_sql_exe_max_rthd_num operand
B: MAX(*C*, *D*)

C: Number of data DB area files in the data DB area that stores the tables to be processed
D: Number of data DB area files in the data DB area that stores the indexes defined in the tables to be processed#

#: If multiple indexes are defined, of the data DB areas storing each of the indexes, specify the number of data DB area files in the data DB area that has the largest number of data DB area files.

#3

If a value of 0 is specified in this operand, SQL statements are executed without applying out-of-order execution. If a value of 1 or greater is specified in this operand, the number of SQL processing real threads indicated in the table are used to execute SQL statements with out-of-order execution applied.

#4

Only connection threads are used to execute SQL statements regardless of the value set for this operand.

> **Important**
>
> If 0 is specified in the adb_sys_uthd_num operand in the server definition, SQL statements are executed in the same manner as when 0 is specified in this operand.

[11] adb_sql_exe_hashgrp_area_size = *hash-group-area-size*

~<integer> ((0, 4 to 1,000,000)) <<4,800>> (kilobytes)

Specify a hash group area size in kilobytes.

When you execute an SQL statement for which local hash grouping applies (hashing is used to group the items in the GROUP BY clause), the specified amount of hash group area is allocated in the real thread private memory for each SQL processing real thread.

If 0 is specified, local hash grouping is not executed. If a value smaller than 4 is specified, 4 is assumed.

A separate hash group area is allocated for each SQL statement to which local hash grouping applies. Its size is determined by the values specified in each GROUP BY clause specified in the SQL statement.

If the size of the specified area is insufficient for local hash grouping, as much hash grouping is performed as can be processed in the size that was specified. Rows that could not be processed are stored as work tables in the work table DB area.

Use the value determined from the formula below as a guideline for the hash group area size. If an SQL statement contains multiple query specifications with the GROUP BY clause specified, use the largest of the values determined for the individual queries.

Formula

adb_sql_exe_hashgrp_area_size operand =

$$\uparrow \left\{ 262{,}144 + (LGC + LSF) \times \mathrm{MIN}(NGR,\ 1{,}000{,}000) \right\} \div 1{,}024 \uparrow$$

Explanation of variables

- *LGC*

  Sum total of the data lengths of the grouping columns

  See Table 6-9: Data length of each data type in (c) Determining the variable RTHD_EXESQLSZ under (2) Determining the real thread private memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

- *LSF*

  Data length of the set function result

- *NGR*

  Number of rows in the grouping result

When this operand and the client definition adb_sql_exe_hashgrp_area_size operand are both specified, the value specified by the adb_sql_exe_hashgrp_area_size operand in the client definition is prioritized. For details about the adb_sql_exe_hashgrp_area_size operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

[12] adb_sql_exe_hashtbl_area_size = *hash-table-area-size*

~<integer> ((0 to 4,194,304)) <<2,000>> (megabytes)

Specify a hash table area size in megabytes.

When you execute an SQL statement to which hash retrieval is applied, a hash table area is allocated in the real thread private memory for each SQL processing real thread, according to the following formula:

Formula

```
↑value-specified-for-this-operand ÷ value-specified-for-adb_sql_exe_max_rthd_num
-operand↑
```

If 0 is specified for the adb_sql_exe_max_rthd_num operand, hash retrieval is not applied.

> **Note**
>
> Hash retrieval is processing that uses a hash table area. The following types of processing apply:
>
> - Hash join as a table joining method
> - Global hash grouping as a grouping method
> - Hash execution as a method for processing subqueries
> - Hash execution as a method for processing SELECT DISTINCT
> - Hash execution as a method for processing the set operation

For details about the preceding types of hash retrieval, see the following sections in the *HADB Application Development Guide*:

- *Table joining methods*
- *How to process subqueries*
- *Grouping methods*
- *Method for processing SELECT DISTINCT*
- *Methods for processing set operations*

A hash table area is allocated for each SQL statement to which hash retrieval is applied.

If 0 is specified for this operand, hash retrieval is not applied. Specify 0 for this operand only in the following cases:

Cases in which 0 can be specified for this operand

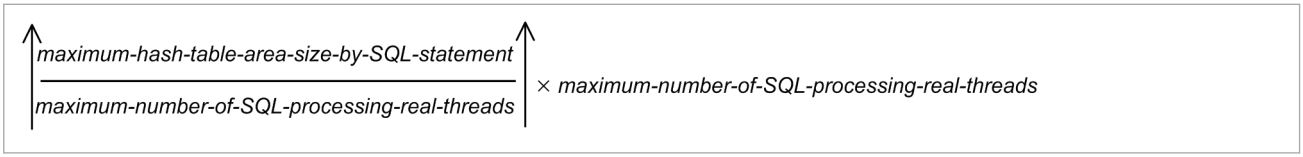
Cases in which a memory shortage error cannot be avoided because the value specified for the following operands cannot be increased (when the KFAA30919-E or KFAA30930-E message is output):

- adb_sys_rthd_area_max operand
- adb_sys_memory_limit operand
- adb_dbbuff_wrktbl_clt_blk_num operand

If the specified area size is insufficient for hash retrieval, a hash table is created that is as large as can be processed with the specified area size. Rows that cannot be processed with hash tables are stored as work tables in the work table DB area.

Specify the value determined using the following formula for the hash table area size.

Formula



Explanation of variables

- *maximum-hash-table-area-size-by-SQL-statement*

  The largest among the hash table area sizes by SQL statement determined for all SQL statements to be executed. Use the following formula to determine the hash table area size by SQL statement:

  number-of-hash-tables × *maximum-row-length-in-hash-tables* × *maximum-number-of-intermediate-retrieval-results*

  For details about the number of hash tables, the maximum row length in the hash tables, and the maximum number of intermediate retrieval results, see 6.25.3 Number of work tables created during retrieval using hash tables.

- *maximum-number-of-SQL-processing-real-threads*

  This is the value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition. However, if `0` is specified for the `adb_sql_exe_max_rthd_num` operand, assume 1 for the calculation.

When this operand and the client definition `adb_sql_exe_hashtbl_area_size` operand are both specified, the value specified by the `adb_sql_exe_hashtbl_area_size` operand in the client definition is prioritized. However, if the value specified by the client definition `adb_sql_exe_hashtbl_area_size` operand is greater than the value specified by this operand, the former is not prioritized, and the value specified by this operand is prioritized. For details about the `adb_sql_exe_hashtbl_area_size` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

[13] `adb_sql_exe_hashflt_area_size` = *hash-filter-area-size*

~<integer> ((0 to 419,430)) <<value of adb_sql_exe_hashtbl_area_size server definition operand ÷ 10>> (megabytes)

Specify the size (in megabytes) of the hash filter area in which a hash filter is to be created. The hash filter is applied when the following types of processing are executed:

- Hash join as a table joining method
- Hash execution as a method for processing subqueries

For details about hash join, hash execution, and hash filters, see *About hash join* and *How to process subqueries* in the *HADB Application Development Guide*.

A hash filter area is allocated for each SQL statement to which hash filters are applied when hash join or hash execution is performed. When you execute an SQL statement to which hash filters are applied, a hash filter area is allocated in the real thread private memory for each SQL processing real thread, according to the following formula:

```
↑value-specified-for-this-operand ÷ value-specified-for-adb_sql_exe_max_rthd_num-operand↑
```

Guideline and adjustment of the value to be specified

- As a guideline, use the value determined from the following formula as the value to be specified for this operand:

```
hash-filter-area-size =
↑value-specified-for-adb_sql_exe_hashtbl_area_size-operand (hash-table-area-size) ÷ 10↑
```

- Specify `0` for this operand only if the processing performance is degraded as a result of applying hash filters when hash join or hash execution is performed.

- If the value of this operand is too small to allocate a sufficient hash filter area, no hash filters are applied when hash join or hash execution is performed. You can check whether hash filters were applied from the values of `Hashtbl_filter_num` and `Hashtbl_sum_filter_check_cnt` in the access path statistical information.

  In a case where the value specified for this operand is not `0`, if the value output to `Hashtbl_filter_num` is not `0` and the value output to `Hashtbl_sum_filter_check_cnt` is `0`, the hash filter area is insufficient. Because no hash filters have been applied, increase the value of this operand. For details about the access path statistical information, see (e) Information related to hash table areas in (2) Items that are output as access path statistical information in 10.11.3 Examples of output of and output items for access path statistical information.

Notes

- If `0` is specified for this operand, no hash filters are applied when hash join or hash execution is performed.
- If `0` is specified for the `adb_sql_exe_max_rthd_num` operand, hash join or hash execution is not applied.
- If this operand is specified when the `adb_sql_exe_hashflt_area_size` operand is specified in the client definition, the value specified by the `adb_sql_exe_hashflt_area_size` operand in the client definition is prioritized. However, if the value specified by the client definition `adb_sql_exe_hashflt_area_size` operand is greater than the value specified by this operand, the former is not prioritized, and the value specified by this operand is prioritized. For details about the `adb_sql_exe_hashflt_area_size` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

[14] `adb_sql_opt_drvtbl_grping_prior = {LOCAL|GLOBAL}`

Specify the grouping method that is to be prioritized when an SQL statement that meets all of the following conditions is executed:

- Only one table reference is specified in the `FROM` clause.
- The preceding table reference is an internal derived table that is not expanded.
- The `GROUP BY` clause is specified in the outermost query specification.

Normally, this operand does not need to be specified.

If there are many (about 30 or more) SQL processing real threads that can be used to process the SQL statement, specifying `GLOBAL` might shorten the time required to execute the SQL statement.

- `LOCAL`

  Specify this to prioritize local hash grouping as the grouping method.

- `GLOBAL`

  Specify this to prioritize global hash grouping as the grouping method.

The following example describes the relationship between the value of this operand and the grouping method:

Example:

```
SELECT "C2"
    FROM (SELECT "C1",COUNT(*) FROM "T1" GROUP BY "C1") AS "DT1" ("C1","C2")
    GROUP BY "C2"
```

If you execute the preceding SQL statement by specifying `LOCAL` for this operand, local hash grouping is used as the method for processing the outer grouping (underlined grouping).

If you execute the preceding SQL statement by specifying `GLOBAL` for this operand, global hash grouping is used as the method for processing the outer grouping (underlined grouping).

Multi-node function

If you are using the multi-node function, we recommend that you specify the same value for this operand and all HADB servers on all nodes. If you specify a different value on each node, the access path might become different on each node where the SQL statement is executed.

[15] adb_sys_rthd_area_max = *maximum-size-of-real-thread-private-memory-that-can-be-allocated*

~<integer> ((1,024 to 2,000,000,000)) <<1,024>> (megabytes)

Specify in megabytes the maximum size of real thread private memory that can be allocated for each real thread.

There is no need to specify this operand for an HADB server of version **03-01** or later. Instead, specify the adb_sys_memory_limit operand in the server definition. When this operand is specified, the adb_sys_memory_limit operand in the server definition cannot be specified.

If you specify this operand, specify the value that is obtained from the following formula:

Formula

adb_sys_rthd_area_max operand =

$$\mathrm{MAX}\left\{1{,}024, \left(MACHINE\_MEMORY\_SIZE - (SHMMAN + SHM\_BUFGLOBAL) \div 1{,}024\right) \div (RTHD\_NUM + 1)\right\}$$

Explanation of variables

- *MACHINE_MEMORY_SIZE*

  Size of the physical memory installed in the server machine on which the HADB server is installed (megabytes)

- *SHMMAN*

  Size of the area (in kilobytes) acquired for starting the HADB server that stores information about the shared memory used by HADB

  For details about the variable *SHMMAN*, see (1) Determining the shared memory management area requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *SHM_BUFGLOBAL*

  Size of the global buffer area acquired for starting the HADB server (kilobytes)

  For details about the variable *SHM_BUFGLOBAL*, see (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *RTHD_NUM*

  Number of real threads used by the HADB server (threads)

  Use the formula below to determine its value. The formula to be used depends on whether the multi-node function is used.

  Formula (when the multi-node function is not used)

  RTHD_NUM =
  value-specified-for-adb_sys_max_users-operand-in-server-definition
  + *value-specified-for-adb_sys_rthd_num-operand-in-server-definition* + 10

  Formula (when the multi-node function is used)

  RTHD_NUM =
  value-specified-for-adb_sys_max_users-operand-in-server-definition
  + *value-specified-for-adb_sys_rthd_num-operand-in-server-definition* + 10
  + (*number-of-host-names-specified-for-adb_sys_multi_node_info-operand-in-server-definition* × 2 + 3)

[16] adb_sys_proc_area_max = *maximum-size-of-process-common-memory-that-can-be-allocated*

~<integer> ((5,120 to 2,000,000,000)) <<8,192>> (megabytes)

Specify in megabytes the maximum size of process common memory that can be allocated.

There is no need to specify this operand for an HADB server of version **03-01** or later. Instead, specify the adb_sys_memory_limit operand in the server definition. When this operand is specified, the adb_sys_memory_limit operand in the server definition cannot be specified.

If you specify this operand, specify the value that is obtained from the following formula:

Formula

$$\texttt{adb\_sys\_proc\_area\_max}\ \text{operand} = \mathrm{MAX}\left\{5{,}120, \left(MACHINE\_MEMORY\_SIZE - (SHMMAN + SHM\_BUFGLOBAL) \div 1{,}024\right) \div (RTHD\_NUM + 1)\right\}$$

Explanation of variables

- *MACHINE_MEMORY_SIZE*

  Size of the physical memory installed in the server machine on which the HADB server is installed (megabytes)

- *SHMMAN*

  Size of the area (in kilobytes) acquired for starting the HADB server that stores information about the shared memory used by HADB

  For details about the variable *SHMMAN*, see (1) Determining the shared memory management area requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *SHM_BUFGLOBAL*

  Size of the global buffer area acquired for starting the HADB server (kilobytes)

  For details about the variable *SHM_BUFGLOBAL*, see (2) Determining the global buffer page requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

- *RTHD_NUM*

  Number of real threads used by the HADB server (threads)

  Use the formula below to determine its value. The formula to be used depends on whether the multi-node function is used.

  Formula (when the multi-node function is not used)

  RTHD_NUM =
  value-specified-for-adb_sys_max_users-operand-in-server-definition
  + *value-specified-for-adb_sys_rthd_num-operand-in-server-definition* + 10

  Formula (when the multi-node function is used)

  RTHD_NUM =
  value-specified-for-adb_sys_max_users-operand-in-server-definition
  + *value-specified-for-adb_sys_rthd_num-operand-in-server-definition* + 10
  + (*number-of-host-names-specified-for-adb_sys_multi_node_info-operand-in-server-definition* × 2 + 3)

[17] `adb_sys_shm_huge_page_size` = *multiple-of-the-single-page-size-in-HugePages*

~<integer> ((0 to 2,147,483,647)) <<0>> (kilobytes)

When `HugePages` is applied to the shared memory used by the HADB server, specify in kilobytes a multiple of the single page size in `HugePages`.

For this operand, we recommend that you specify the value determined by using the command shown below (the single page size in `HugePages`). When using this command to determine the value, pay attention to the units. If you specify a value that is not a multiple of the single page size in `HugePages`, you will not be able to allocate shared memory and thus will not be able to start the HADB server.

Command to be executed

```
grep Hugepagesize /proc/meminfo
```

Execution result example

```
Hugepagesize:       2048 kB
```

In this example, we recommend that you specify `2048`.

If you specify `0` for this operand, `HugePages` is not applied to the shared memory that the HADB server uses.

To apply `HugePages` to the shared memory that the HADB server uses, you must set up kernel parameters. For details about setting up the kernel parameters, see 6.2 Estimating the kernel parameters.

[18] `adb_dbbuff_wrktbl_glb_blk_num` = *number-of-pages-in-global-buffer-for-global-work-tables*

~<integer> ((5 to 100,000,000)) <<128>>

Specify the number of pages in the global buffer for global work tables.

The global buffer allocated by specifying this operand is used only when an SQL statement that creates global work tables is executed.

For details about how to estimate the value to specify for this operand, see 6.25.1 Estimating the number of pages in the global buffer for global work tables.

For details about global work tables, see *Types of work tables* under *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

[19] `adb_dbbuff_wrktbl_clt_blk_num` = *number-of-pages-in-buffer-for-local-work-tables*

~<integer> ((5 to 100,000,000)) <<256>>

Specify the number of pages in the buffer for local work tables.

The buffer for local work tables allocated by specifying this operand is used only when an SQL statement that creates local work tables is executed. A buffer for local work tables containing the number of pages specified by this operand is allocated for each real thread.

For details about how to estimate the value to specify for this operand, see 6.25.2 Estimating the number of pages in the buffer for local work tables.

For details about local work tables, see *Types of work tables* under *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

Note

- If this operand and the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition are both specified, the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition takes precedence. For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.
- If this operand and the export option `adb_export_wrktbl_blk_num` are both specified, the value specified for the export option `adb_export_wrktbl_blk_num` takes precedence. For details about the export option `adb_export_wrktbl_blk_num`, see *Format of export options* in *Specification format for the adbexport command* under *adbexport (Export Data)* in the manual *HADB Command Reference*.
- You can use the `adbls -d lbuf` command to check the number of pages in the buffer for the local work table that is actually allocated. For details about the `adbls -d lbuf` command, see *adbls -d lbuf (Display Local Work Table Buffer Information)* in the manual *HADB Command Reference*.
- While the HADB server is running, you can use the `adbmodbuff` command to change the number of pages in the buffer for the local work table specified in this operand. For details about the

adbmodbuff command, see *adbmodbuff (Change Buffer)* in the manual *HADB Command Reference*.

[20] adb_sql_tbldef_cache_size = *table-definition-pool-size*

~<integer> ((500 to 2,048,000)) <<5,120>> (kilobytes)

Specify in kilobytes the size of the table-definition pool that stores table-definition information.

Normally, this operand does not need to be specified. To use cost information for operation, read the following description, and specify an appropriate value.

Table-definition information is the information that is required to access a table. This information includes not only the information about the table but also the information about the columns and indexes. A dictionary table is accessed when table-definition information is created. Once table-definition information is used, it is cached in the table-definition pool and managed by the LRU algorithm.

Table-definition information that has been cached in the table-definition pool can be fetched from the pool for reuse the next time it becomes necessary. Therefore, it is unnecessary to create table-definition information again by accessing the dictionary table. Therefore, if you use the table-definition pool, you can expect fewer I/O operations and shorter CPU usage time.

When a table is accessed, if the table-definition pool does not contain the necessary table-definition information, the table-definition information is created and cached in the table-definition pool. If the table-definition pool does not have sufficient free space for caching more information, the definition information for other tables that are not recently used is excluded, and then the new table-definition information is cached. If no free space can be reserved in the table-definition pool, the table-definition information is not cached.

To avoid this, specify the table-definition pool size determined from the following formula so that the table-definition information of all tables that are used in a transaction executed by HADB can be cached. If exclusion of table-definition information occurs due to insufficient free space in the table-definition pool, the processing performance of the SQL statement might be degraded.

Formula

$$\text{adb\_sql\_tbldef\_cache\_size operand} = \left\uparrow \frac{\sum_{i=1}^{m} Wi}{1{,}024} \right\uparrow \times 1.2$$

Explanation of variables

- *m*

  Number of tables accessed frequently

- *Wi*

  Table-definition information length (bytes)

  Use the following formula to determine its value:

  Formula

  $$Wi = 1{,}012 + (512 + 32{,}008^{\#1}) \times col\_num + max\_rowsz + 1{,}156 \times idx\_num + 15{,}864^{\#2}$$

  #1

  Add this value if cost information has been collected in the table.

  For details about how to check whether cost information has been collected in the table, see (1) Determining the names of all base tables from which cost information was collected and the collection dates and times in C.9 Searching system tables.

#2

Add this value if the processing-target table is an archivable multi-chunk table.

col_num

Number of columns in the table

max_rowsz

Maximum row length (bytes)

Determine the maximum row length based on the formula for the row length *ROWSZ* in (1) Determining the number of pages for base rows (variable BP(i)) under 5.8.2 Determining the number of pages for storing each type of row.

idx_num

Number of indexes defined for the table

7.2.3 Operands related to system logs (set format)

[21] `adb_log_usrfile_num` = *number-of-user-log-files*

~<integer> ((0 to 10,240)) <<number of real threads that can concurrently update tables>>

Specify the number of user log files required when specific SQL statements and commands are executed.

The number of user log files specified by this value are created when the HADB server is started. For details about the value to specify in this operand, see 6.12.15 Determining the number of user log files.

The following are precautions:

- If specification of this operand is omitted or `0` is specified, the number of real threads that can update tables concurrently is assumed. As a result, the necessary number of user log files of the size specified in the `adb_log_usrfile_size` operand is created. This prevents an error from occurring due to a lack of user log files.
- If you specify `1` in this operand, `2` is automatically assumed. This means that two user log files will be created.
- Even if you specify for this operand a value that is greater than the number of real threads that can concurrently update tables, the number of real threads that can concurrently update tables is assumed as the value specified for this operand.
- The generated user log files are not deleted until the database is re-initialized. For this reason, make sure that the space available in the system directory under the DB directory is more than the value calculated by using the formula below.

  Formula

  A × *B*

  Explanation of variables:
  A: *Number of user log files*
  (Value specified for the `adb_log_usrfile_num` operand)
  B: *User log file initialization size*
  (Value specified for the `adb_log_usrfile_size` operand)

  Database initialization refers to the execution of the `adbinit` command when the multi-node function is not used. When using the multi-node function, database initialization refers to carrying out the tasks described in (2) Initializing the file systems subject to node switchover under 16.3.8 Creating a database when creating the database.

Multi-node function

If you are using the multi-node function, there is no need to specify this operand. Even if you specify a valid value, `0` is assumed as the value specified for this operand. In this case, the number of user log files needed by all nodes is automatically calculated and the largest among the calculated values is used.

[22] `adb_log_usrfile_size` = *initial-size-of-user-log-file*`[,` *threshold-where-user-log-file-size-is-reduced*`]`

Specify the initial size of the user log file, and the threshold at which a reduction in size of the user log file is triggered.

initial-size-of-user-log-file:

~<integer> ((16 to 1,048,576)) <<16>> (megabytes)

Specify the initial size of user log files in megabytes.

For details about the value to specify in this operand, see 6.12.2 Determining the size of the user log files.

Multi-node function

When using the multi-node function, the largest value specified for *initial-size-of-user-log-file* among all nodes takes effect.

threshold-where-user-log-file-size-is-reduced:

~<integer> ((0, 32 to 2,097,152)) <<initial-size-of-user-log-file x 2>> (megabytes)

Specify in megabytes the threshold at which the size of the user log file is to be reduced. If the size of the user log file used by an update transaction exceeds the size specified for this operand when the update transaction is settled, the user log file is reduced to its initial size. Normally, there is no need to specify this operand.

The rules are as follows:

- If you specify `0` for *threshold-where-user-log-file-size-is-reduced*, reduction in size of the user log file will not take place.
- If *threshold-where-user-log-file-size-is-reduced* < *initial-size-of-user-log-file*, then the default value (*initial-size-of-user-log-file* x 2) is assumed.

Multi-node function

- When using the multi-node function, the largest value specified for *threshold-where-user-log-file-size-is-reduced* among all nodes takes effect.
- If the value of *threshold-where-user-log-file-size-is-reduced* that takes effect is less than the value of *initial-size-of-user-log-file* that takes effect, then a value equivalent to twice the initial size of the user log file is assumed for *threshold-where-user-log-file-size-is-reduced*.
- If you specify `0` for *threshold-where-user-log-file-size-is-reduced* on all nodes, the size of the user log file will not be reduced.

[23] `adb_log_usrbuf_num` = *number-of-user-log-buffers*

~<integer> ((64 to 12,582,912)) <<64>>

Specify the number of buffers used for accessing user log files.

Normally, there is no need to specify this operand. Increasing the value might improve the speed of accessing user log files in some cases.

Note that the HADB server allocates a user log buffer to each user log file.

Multi-node function

If you are using the multi-node function, the largest among the numbers of buffers used for accessing user log files specified on the individual nodes is used.

[24] `adb_log_rec_msg_interval` = *output-interval-of-message-showing-database-recovery-processing-time-elapsed*

~<integer> ((0 to 60)) <<5>> (minutes)

Specify the interval at which a message is output indicating how much time (in minutes) has elapsed when performing database recovery processing.

Normally, there is no need to specify this operand.

During database recovery processing, the following two messages are output according to the value specified for this operand:

■ **Cases in which the KFAA81215-I message is output**

When database recovery processing is initiated by processing such as the following, a message KFAA81215-I indicating how long database recovery processing has taken so far is output at a fixed interval determined by the value of this operand.

- Restarting the HADB server
- Rolling back a transaction

For example, if 3 is specified for this operand, the message KFAA81215-I is output every three minutes while database recovery processing is in progress. You can use the contents of these KFAA81215-I messages to follow the progress of database recovery processing.

■ **Cases in which the KFAA81211-I message is output**

The message KFAA81211-I is also output during database recovery processing when the recovery processing for a particular transaction is taking a long time.

For example, if 3 is specified for this operand, the message KFAA81211-I is output when a single transaction has taken three minutes or longer to recover, and when recovery of that transaction has completed. The KFAA81211-I message contains information related to the recovery processing for the transaction in question. You can use this output information as part of the resources you use to identify the cause of a delay in transaction recovery processing.

If you specify 0 for this operand, neither message KFAA81211-I nor KFAA81215-I is output.

7.2.4 Operands related to status monitoring (set format)

[25] adb_sys_max_users_wrn_pnt = *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections*[**,** *reset-threshold-for-warning-message-output*]

Specify this operand to output the warning message KFAA40020-W when the number of concurrent connections approaches the maximum number of concurrent connections specified in the adb_sys_max_users operand.

*output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections***:**

~<integer> ((0 to 100)) <<0>> (percent)

Specify the output threshold for the warning message KFAA40020-W as a proportion of the value specified for the adb_sys_max_users operand (the maximum number of concurrent connections). For example, if you specify 200 for the adb_sys_max_users operand and 90 as the value of *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections*, a warning message is output when the number of connections to the HADB server reaches 180 (which is 90% of 200).

If you specify 0 for *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections*, the warning message will not be output.

reset-threshold-for-warning-message-output:

~<integer> ((0 to 99)) (percent)

Specify the threshold for resetting the output state of the warning message KFAA40020-W as a proportion of the value specified for the adb_sys_max_users operand (the maximum number of concurrent connections). When the warning message has been output once, HADB registers a state in which the warning message has

been output. In this state, the warning message is not output again until the number of connections to the HADB server has fallen below a specific value.

Note

The relationship between the value specified for the `adb_sys_max_users_wrn_pnt` operand and output of the warning message is as follows:

```
adb_sys_max_users = 200
adb_sys_max_users_wrn_pnt = 90,70
```

When the operands are specified in this way, the warning message is output when the number of connections to the HADB server reaches 180 (which is 90% of 200). If the number of connections to the HADB server then falls to 170 before rising again to 180 or more, HADB will not output the warning message again.

However, if the number of connections to the HADB server falls to 140 (70% of 200) or below, the state in which the warning message has been output is reset. This means that subsequently, the warning message will be output again when the number of connections to the HADB server reaches 180 or more.

Notes regarding the threshold for resetting the warning message output state are as follows:

- If you do not specify a reset threshold, a value equivalent to *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections* - 30 is assumed. If this results in a negative value, `0` is assumed.
- Specify the operand so that *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections* > *reset-threshold-for-warning-message-output*. If this condition is not satisfied, the reset threshold specification is invalid (the warning message `KFAA40022-W` is output). In this case, a value of *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections* - 30 is assumed.
- If you specify `0` for *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections*, the reset threshold specification is invalid.

Multi-node function

- If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.
- The warning message `KFAA40020-W` is output on the master node.

[26] `adb_sys_memory_limit_wrn_pnt =` *output-threshold-for-warning-message-regarding-HADB-server-memory-usage* `[,` *reset-threshold-for-warning-message-output* `]`

Specify this operand to output the warning message `KFAA40021-W` when the remaining amount of memory available to the HADB server becomes low.

The value specified for the `adb_sys_memory_limit_wrn_pnt` operand is only valid if the `adb_sys_memory_limit` operand is also specified.

output-threshold-for-warning-message-regarding-HADB-server-memory-usage:

~<integer> ((0 to 100)) <<0>> (percent)

Specify the output threshold for the warning message `KFAA40021-W` as a proportion of the value specified for the `adb_sys_memory_limit` operand (the maximum amount of memory available to the HADB server). For example, if you specify `64,000` MB for the `adb_sys_memory_limit` operand and `90` as the value of

output-threshold-for-warning-message-regarding-HADB-server-memory-usage, a warning message is output when the memory used by the HADB server reaches 57,600 MB (which is 90% of 64,000).

Important

Immediately after starting the HADB server, execute the `adbls -d mem -a` command and check the value of `TOTAL_USE_PER` in the command output. Specify a value for this operand that exceeds the value of `TOTAL_USE_PER`. `TOTAL_USE_PER` is the ratio (in percent) of the current memory usage of the HADB server to the value specified in the `adb_sys_memory_limit` operand.

If you specify a value less than `TOTAL_USE_PER`, the conditions for outputting the warning message will be met as soon as the HADB server starts. In this case, the warning message is output when the HADB server starts, and no further warning messages are output during HADB operation.

If you specify `0` for *output-threshold-for-warning-message-regarding-HADB-server-memory-usage*, the warning message will not be output.

reset-threshold-for-warning-message-output:

~<integer> ((0 to 99)) (percent)

Specify the threshold for resetting the output status of the warning message `KFAA40021-W` as a proportion of the value specified for the `adb_sys_memory_limit` operand (the maximum amount of memory available to the HADB server). When the warning message has been output once, HADB registers a state in which the warning message has been output. In this state, the warning message is not output again until the memory usage of the HADB server has fallen below a specific value.

Note

The relationship between the value specified for the `adb_sys_memory_limit_wrn_pnt` operand and output of the warning message is as follows:

```
adb_sys_memory_limit = 64,000
adb_sys_memory_limit_wrn_pnt = 90,70
```

When the operands are specified in this way, the warning message is output when the memory usage of the HADB server reaches 57,600 MB (which is 90% of 64,000). Subsequently, if the memory usage of the HADB server falls to 51,200 MB (which is 80% of 64,000) before rising again to 57,600 MB or more, HADB will not output the warning message again.

However, if the memory usage of the HADB server falls to 44,800 MB (70% of 64,000) or less, the state in which the warning message has been output is reset. This means that the warning message will be output again when the memory usage of the HADB server reaches 57,600 MB.

Notes regarding the threshold for resetting the warning message output state are as follows:

- If you do not specify a reset threshold, a value equivalent to *output-threshold-for-warning-message-regarding-HADB-server-memory-usage* - 30 is assumed. If this results in a negative value, `0` is assumed.
- Specify the operand so that *output-threshold-for-warning-message-regarding-HADB-server-memory-usage* > *reset-threshold-for-warning-message-output*. If this condition is not satisfied, the reset threshold specification is invalid (the warning message `KFAA40022-W` is output). In this case, a value of *output-threshold-for-warning-message-regarding-HADB-server-memory-usage* - 30 is assumed.
- If you specify `0` for *output-threshold-for-warning-message-regarding-HADB-server-memory-usage*, the reset threshold specification is invalid.

Multi-node function

The warning message KFAA40021-W might be output on all nodes.

[27] adb_rpc_wait_time = *upper-limit-for-wait-time*

~<integer> ((0 to 86,400)) <<1,800>> (seconds)

Specify, in seconds, the maximum amount of time to wait for the next processing request after having responded to a processing request from an HADB client.

If no processing request is received from the HADB client within this wait time limit, the connection to the HADB client is terminated. If a transaction is being processed, that transaction is rolled back before disconnecting the HADB client.

If 0 is specified for this operand, no upper limit is set for the wait time.

Multi-node function

When master node switchover occurs, the application programs that are connected to the HADB server but that are not performing transactions are disconnected from the HADB server.

[28] adb_rpc_tcp_keepalive_time = *no-communication-time-after-which-KeepAlive-probe-starts*

~<integer> ((0 to 120)) <<10>> (minutes)

Specify in minutes the no-communication time after which an alive check using the KeepAlive probe starts for the connection established with an HADB client.

Normally, there is no need to specify this operand. By specifying this operand, you can change the OS's KeepAlive control (the no-communication time after which the KeepAlive probe starts).

If there is no response to the KeepAlive probe, the alive check is repeated four times at 30-second intervals. If there is still no response, the connection to the HADB client is cut off.

If 0 is specified for this operand, the alive check using the KeepAlive probe is not performed.

Note

KeepAlive is a function that checks whether a connection is alive based on a response or non-response to a probe packet that is sent at regular intervals. A *probe packe*t is a packet that is sent to check whether a communication line is alive.

7.2.5 Operands related to SQL statements (set format)

[29] adb_sql_prep_delrsvd_words = *reserved-word-to-be-unregistered*[,*reserved-word-to-be-unregistered*]...

~<character string> ((1 to 100 bytes))

Specify the words to be unregistered as HADB reserved words. You can specify up to 999 reserved words for unregistration.

To unregister a reserved word, specify it in all uppercase letters enclosed in the backslash and double quotation mark (\"), or specify it in all uppercase or all lowercase letters not enclosed in the backslash and double quotation mark (\"). The following are specification examples in which the specified reserved word will be unregistered correctly.

Example: Specification examples to unregister the reserved word ABS

- adb_sql_prep_delrsvd_words = \"ABS\"
- adb_sql_prep_delrsvd_words = ABS
- adb_sql_prep_delrsvd_words = abs

> **Note**
>
> A character string that is enclosed in the backslash and double quotation mark (\") is treated as being case-sensitive. Therefore, if you specify a character string consisting entirely of lowercase letters enclosed in the backslash and double quotation mark, such as \"abs\", the character string is regarded as lowercase only. However, because there are no lowercase reserved words in HADB, you cannot unregister a reserved word by enclosing a character string consisting of all lowercase letters in the backslash and double quotation mark.

Note that the reserved words that can be unregistered with the adb_sql_prep_delrsvd_words operand are predetermined. If a specified word is not a reserved word or is a reserved word that cannot be unregistered, the KFAA51111-W warning message will be issued.

For details about reserved words that cannot be unregistered, see *List of reserved words* in the manual *HADB SQL Reference*.

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes. When you do so, also specify the reserved words in the same order.

[30] adb_sys_trn_iso_lv = {READ_COMMITTED|REPEATABLE_READ}

Specify the transaction isolation level to be applied. For details about transaction isolation levels, see 2.9.2 Transaction isolation levels supported by HADB.

- READ_COMMITTED

  Applies READ COMMITTED as the transaction isolation level.

- REPEATABLE_READ

  Applies REPEATABLE READ as the transaction isolation level.

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.

[31] adb_sql_text_out = {Y|N}

Specify whether to output all SQL statements accepted by the HADB server to the server message log file.

When Y is specified, the first 2,048 bytes of each SQL statement are output.

- Y

  Output SQL statements to the server message log file.

- N

  Do not output SQL statements to the server message log file.

Even when N is specified for this operand or this operand is omitted, SQL statements are output to the server message log file if Y is specified for the adb_clt_sql_text_out operand in the client definition.

For details about the adb_clt_sql_text_out operand, see *Operands related to SQL* in the *HADB Application Development Guide*.

[32] adb_sql_trc_out = {Y|N}

Specify whether SQL trace information is to be output when SQL statements are executed.

- Y

  Output SQL trace information to SQL trace files.

- N

Do not output SQL trace information to SQL trace files.

When N is specified in this operand, SQL trace information is not output regardless of the values specified in the following operands:

- adb_sql_trc_param operand in the server definition
- adb_sql_trc_accesspath operand in the server definition
- adb_sql_trc_level operand in the server definition
- adb_sql_trc_txtfile_size operand in the server definition

[33] adb_sql_trc_param = {Y|N}

Specify whether information about dynamic parameters is to be output as part of the SQL trace information.

- Y

  Output information about dynamic parameters as part of the SQL trace information.

- N

  Do not output information about dynamic parameters as part of the SQL trace information.

When Y is specified in this operand but N is specified in the adb_sql_trc_out operand in the server definition, information about dynamic parameters is not output.

[34] adb_sql_trc_accesspath = {Y|N}

Specify whether access path information and access path statistical information are to be output as part of the SQL trace information.

- Y

  Output access path information and access path statistical information as part of the SQL trace information.

- N

  Do not output access path information and access path statistical information as part of the SQL trace information.

When Y is specified in this operand but N is specified in the server definition's adb_sql_trc_out operand, access path information and access path statistical information are not output.

[35] adb_sql_trc_level = {SQL|CALL}

Specify the unit for output of SQL trace information.

- SQL

  Output SQL trace information for each SQL statement.

- CALL

  Output SQL trace information for each call.

[36] adb_sql_trc_txtfile_size = *maximum-capacity-of-an-SQL-trace-file*

~<integer> ((32 to 1,024)) <<256>> (megabytes)

Specify in megabytes the maximum size of each SQL trace file. Eight SQL trace files are created.

Normally, there is no need to specify this operand. Specify this operand when you want to change the maximum size of the individual SQL trace files.

The maximum size of an SQL trace files specified in this operand takes effect when the HADB server starts. If the value of this operand is reduced, SQL trace files whose size is greater than the specified value might still exist immediately after the HADB server has started. In such a case, the maximum size of SQL trace files specified in this operand takes effect when the corresponding SQL trace files are swapped out to output SQL trace information.

[37] adb_sql_order_mode = {BYTE|ISO}

Specify the sort order mode for character string data when the search results of the SELECT statement with the ORDER BY clause specified are sorted. If this operand is omitted, BYTE is assumed.

- BYTE

  Character string data is sorted by bytecode.

- ISO

  Character string data is sorted by sort code (ISO/IEC 14651:2011 compliance).

  However, if SJIS is specified in the environment variable ADBLANG, you cannot specify ISO for this operand.

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.

You can also specify the sort order mode for character string data in the adb_clt_sql_order_mode operand in the client definition and in the connection attribute. If the sort order mode for character string data is specified in these locations, the sort order is determined according to the priority listed in the following table (the smaller the number, the higher the priority).

Table 7-4: Priority of the sort order mode for character string data

Priority	Location where the sort order mode for character string data is specified
1	Connection attribute
2	adb_clt_sql_order_mode operand in the client definition
3	adb_sql_order_mode operand in the server definition

Explanation

For example, if ISO is specified in the adb_sql_order_mode operand in the server definition and BYTE is specified in the adb_clt_sql_order_mode operand in the client definition, BYTE is applied to the SELECT statement (with the ORDER BY clause specified) executed from an application program.

If you omit specification in all locations listed in the above table, BYTE is assumed.

[38] adb_sql_prep_dec_div_rs_prior = {INTEGRAL_PART|FRACTIONAL_PART}

If the data type of the result of a division (arithmetic operation) specified in an SQL statement is DECIMAL, specify the minimum value of the scaling value for the division result.

INTEGRAL_PART:

The minimum value of the scaling for a division result is 0. If INTEGRAL_PART is specified, the number of digits in the integer part takes precedence. Therefore, when you execute an SQL statement that performs division whose result might become a large value, you can avoid occurrence of an overflow error wherever possible by specifying INTEGRAL_PART.

FRACTIONAL_PART:

The scaling of the first operand (dividend) is used as the minimum value for a division result. If FRACTIONAL_PART is specified, the number of digits in the fractional part of the first operand is the minimum number of digits in the fractional part of the division result.

If the first operand is DECIMAL(*p1*,*s1*) and the second operand is DECIMAL(*p2*,*s2*), the division result is DECIMAL(38,*s*).

- If INTEGRAL_PART is specified

  $s = \mathrm{MAX}\ \{0, 38 - (p1 - s1 + s2)\}$

- If FRACTIONAL_PART is specified

$s = \mathrm{MAX}\ \{s1, 38 - (p1 - s1 + s2)\}$

The following shows an example of a difference in division results depending on the value specified for this operand.

Example:

Table T1

Column C1 DECIMAL(38,4)	Column C2 DECIMAL(10,5)
30.5256	0.05223

Suppose that the following `SELECT` statement is executed.

```
SELECT "C1"/"C2" AS "division result" FROM "T1"
```

- The division result if `INTEGRAL_PART` is specified

  `584.`

- The division result if `FRACTIONAL_PART` is specified

  `584.4457`

For details about an example of the scaling for division results when a viewed table is searched, see *Notes applying when the data type of the division result is DECIMAL* in the manual *HADB SQL Reference*.

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.

[39] `adb_sql_default_dbarea_shared =` *name-of-default-data-DB-area-that-stores-tables-or-indexes*

~<character string> ((1 to 30 bytes))

Specify the name of a data DB area. The data DB area you specify in this operand is used as the default DB area for storing tables or indexes when no DB area is specified for storing the table or index in the following SQL statements:

- `CREATE TABLE` statement
- `CREATE INDEX` statement
- `ALTER TABLE` statement

For details about the naming rules that apply to DB area names, see *Specifying names* in the manual *HADB SQL Reference*.

If either of the following applies, a warning message is output when the HADB server starts. The value specified for this operand takes effect after you have resolved the cause of the warning.

- The DB area specified in the operand is not a data DB area
- The DB area specified in the operand does not exist

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.

7.2.6 Operands related to range indexes (set format)

[40] `adb_sql_rngidx_preread ={ALL|NO|`*range-index-name*`[,`*range-index-name*`]...}`

Specify the range indexes that are to be pre-read when the HADB server starts. For details about pre-reading of range indexes, see 5.5.3 Pre-reading of range indexes.

ALL:

Pre-reads all range indexes.

NO:

Does not pre-read any range indexes.

range-index-name:

~<character string> ((1 to 205 bytes))

Specify the names of range indexes that are to be pre-read. You can specify up to 8,192 range index names. Range indexes are pre-read in the order in which they are specified here.

> **Important**
>
> You must specify a range index name in the format *schema-name* **.** *index-identifier*. You cannot omit the schema name.
>
> Example: \"ADBUSER01\".\"RINDX01\"

Notes:

- To pre-read range indexes, you must specify the adbbuff operand's -a option (number of pages in the global buffer dedicated to range indexes). Therefore, when allocating a global buffer to the DB area that stores range indexes to be pre-read, you must specify the -a option.
- When range indexes are pre-read, multiple pages are read based on a single page request. Consequently, in the global buffer statistical information that is output when the adbstat command is executed, the following two types of information might not match each other in some cases:
  - DBbuff_page_rng_request_cnt (number of requests for pages in the range index buffer)
  - DBbuff_page_rng_pagein_cnt (number of times pagein occurs in the range index buffer)

  For details about the adbstat command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.
- If the number of pages specified by the adbbuff operand's -a option become full while range indexes are being pre-read, only some of the range indexes will be in pre-read status.
- Range indexes in unfinished status cannot be pre-read.
- If a specified range index name contains a syntax error, that range index is not pre-read. Other range indexes are pre-read.

7.2.7 Operands related to statistical information (set format)

[41] adb_sta_log_max_size = *maximum-size-of-statistics-log-file*

~<integer> ((1 to 16)) <<4>> (gigabytes#)

Specify in gigabytes the maximum size of each statistics log file. Four statistics log files are created.

Normally, there is no need to specify this operand. Specify this operand if you want to reduce the maximum size of statistics log files.

Note that the maximum size of statistics log files, as specified for this operand, takes effect when a statistics log file is switched. Therefore, even if you specify this operand, the maximum size of the statistics log files differ from the value specified for this operand until switching of a statistics log file occurs.

#

To test a swap between small statistics log files, you can use the adb_sta_log_size_unit operand to change the unit of the maximum size of statistics log files to a smaller size (MB).

[42] adb_sta_log_path = *name-of-output-destination-directory-for-statistics-log-files*

~<path name> ((2 to 510 bytes))

Specify an absolute path for the name of the output destination directory for statistics log files.

Statistics log files are output to the directory specified by this operand.

Normally, this operand does not need to be specified. If this operand is omitted, statistics log files are output to the $ADBDIR/spool directory.

Specify this operand if you want to output statistics log files to a directory other than the $ADBDIR/spool directory.

Note the following when specifying this operand.

- Specify a dedicated directory in which only statistics log files are stored.
- Specify a directory for which the HADB administrator has access privileges (read, write, and execution privileges).
- Specify a directory on the disk where enough free space is allocated.

  If the adb_sta_log_max_size and adb_sta_log_size_unit operands are omitted, the maximum size of each statistics log file is 4 gigabytes. Four statistics log files are created. Therefore, the minimum required size is 16 gigabytes.

  > **Note**
  >
  > You can change the maximum size for a statistics log file by using the adb_sta_log_max_size and adb_sta_log_size_unit operands in the server definition.

- If you change the output destination directory for statistics log files, statistical information stored in the statistics log files before change can no longer be referenced. Therefore, before changing the output destination directory for statistics log files, output the statistical information by using the adbstat command.

[43] adb_sta_log_size_unit = {G|M}

Specify this to change the unit of the maximum size of statistics log files that is specified for the adb_sta_log_max_size operand.

In normal operation, omit this operand. Specify this operand if you want to change the unit of the maximum size to MB when, for example, you test a swap between statistics log files. If this operand is specified, a warning message (KFAA52201-W) is output.

- G

  Specifying this value sets the unit of the maximum size of statistics log files to GB.

- M

  Specifying this value sets the unit of the maximum size of statistics log files to MB.

7.2.8 Operands related to synonym search (set format)

[44] adb_syndict_storage_path = *storage-directory-for-synonym-dictionary-files*

~<path name> ((2 to 510 bytes))

Specify an absolute path name of the storage directory for synonym dictionary files. Specify this operand to perform a synonym search.

When specifying this operand, see (3) Estimating the size required for the directory for storing synonym dictionary files to (6) Modifying the server definition in 11.16.1 Preparing for synonym search operations.

Multi-node function

If you are using the multi-node function, specify the same value for this operand for all of the HADB servers on all nodes.

Note

- If the `adbsyndict` command is executed with this operand omitted, the `adbsyndict` command results in an error.
- If the information specified for this operand is invalid, an error occurs when the HADB server starts.

[45] `adb_syndict_node_storage_path =` *multi-node-synonym-dictionary-storage-directory*

~<path name> ((2 to 510 bytes))

Specify the multi-node synonym dictionary storage directory by absolute path. Specify this operand if you intend to use synonym search when using the multi-node function. When you specify this operand, see (6) Creating the multi-node synonym dictionary storage directory in 16.24.1 Preparing for synonym search operations.

Important

If you change the value specified for this operand, you need to synchronize the synonym dictionary files by executing the `adbsyndict` command. For details, see 16.24.5 Synchronizing synonym dictionary files.

Note

- If you execute the `adbsyndict` command (to register or update a synonym dictionary or synchronize synonym dictionary files) and this operand has been omitted, the `adbsyndict` command results in an error.
- If you specify this operand incorrectly, an error occurs when the HADB server starts.
- The warning message `KFAA51525-W` is output if all of the following conditions are met:
  - The multi-node function is enabled
  - The `adb_syndict_storage_path` operand is specified
  - The `adb_syndict_node_storage_path` operand is not specified

7.2.9 Operands related to audit trail facility (set format)

[46] `adb_audit_log_path =` *audit-trail-directory*

~<path name> ((2 to 510 bytes))

Specify an absolute path name for the audit trail directory. You must specify this operand when using the audit trail facility.

[47] `adb_audit_log_max_size =` *maximum-size-of-audit-trail-file*

~<integer> ((256 to 1,024)) <<256>> (megabytes)

Specify in megabytes the maximum size of each audit trail file.

Normally, there is no need to specify this operand. Specify this operand when you want to increase the maximum size of audit trail files.

To manage audit trail information by linking with JP1/Audit, specify the default value for this operand. Note that if you specify a large number as the maximum size of each audit trail file, JP1/Audit might fail to collect all audit trails.

[48] `adb_audit_log_max_num =` *maximum-number-of-audit-trail-file-generations*

~<integer> ((0, 4 to 10,000)) <<0>>

Specify the maximum number of generations of audit trail files that can be created in the audit trail directory.

The value specified for this operand becomes the maximum number of audit trail files created in the audit trail directory.

Normally, there is no need to specify this operand. Specify this operand when you want to limit the number of audit trail file that can be created. If the number of audit trail file exceeds the maximum, the oldest audit trail file is deleted.

If this operand is omitted or `0` is specified for this operand, there is no limit to the number of audit trail files that can be created.

If you specify a value from `1` to `3` in this operand, `4` is assumed. In this case, the `KFAA51413-W` message is output.

The value specified in this operand takes effect when the HADB server starts.

> **Important**
>
> When changing the value specified for this operand, first check how many audit trail files are stored in the audit trail directory. Then, specify in this operand a value that is greater than the number of audit trail files that you checked.
>
> If you specify a smaller value in this operand than the number of audit trail files stored in the audit trail directory, the HADB server cannot start (an error will occur).

7.2.10 Operands related to the multi-node function (set format)

[49] `adb_sys_multi_node_info =` *node-host-name*`[:`*node-port-number*`],` *node-host-name*`[:`*node-port-number*`][,`*node-host-name*`[:`*node-port-number*`]] ...`

~<character string> ((1 to 261 bytes))

Specify, in the *host-name*`:`*port-number* format, the host name of the HADB server and port number to be used for communication among HADB servers on the individual nodes used by the multi-node function. Specifying this operand allows you to use the multi-node function.

node-host-name`[:`*node-port-number*`]` can be specified up to four times.

> **Important**
>
> The network to be used among HADB servers must be the same as the network for the HA Monitor monitoring path. For details, see (2) Inter-node network in 16.2.3 Network configuration.

You must also specify for this operand the host name and port number of the local node (the HADB server that is started with this server definition specified).

Specify the same value for this operand for all nodes that are used by the multi-node function. Also, use the same specification order.

Note that you can omit the port numbers to be used for communication among HADB servers. If you omit the specification, `23651` is assumed.

For details about the multi-node function, see 16. Operations When Using the Multi-Node Function.

[50] `adb_sys_node_start_wait_time` = *upper-limit-for-time-to-wait-for-startup-completion-of-other-nodes*

~<integer> ((1 to 7,200)) <<300>> (seconds)

If you are using the multi-node function, specify in seconds the upper limit for the time to wait for startup completion of all nodes.

As a rule, specify the default value for this operand.

7.2.11 Operands and options related to global buffers (command format)

[51] `adbbuff`

Defines the content of the global buffers to be allocated to a data DB area. You can specify this operand multiple times.

When you specify this operand, use a value such that the combined total value of the following two variables will be 30 to 40% of the maximum size of the memory used by the HADB server, as determined with reference to 6.3 Estimating the HADB server's memory requirement.

- The variable *SHM_BUFGLOBAL* explained in (2) Determining the global buffer page requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server
- The variable *BUFGLOBAL* explained in (d) Determining the variable BUFGLOBAL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.

Note that because global buffers are allocated automatically to the master directory DB area, dictionary DB area, system-table DB area, and work table DB area, you do not need to define them in this operand.

You can use the `adbls -d gbuf` command to check the contents of the global buffers allocated by this operand. For details about the `adbls -d gbuf` command, see *adbls -d gbuf (Display Global Buffer Information)* in the manual *HADB Command Reference*.

Note

A global buffer for global work tables and a buffer for local work tables are allocated to the work table DB area. The following describe their specification methods:

- **How to specify a global buffer for global work tables**

  You can specify the number of pages in the global buffer for global work tables in the adb_dbbuff_wrktbl_glb_blk_num server definition operand explained in 7.2.2 Operands related to performance (set format). The name of the global buffer for global work tables is `ADBWRK`.

- **Specifying a buffer for local work tables**

  You can specify the number of pages in the buffer for local work tables in the adb_dbbuff_wrktbl_clt_blk_num server definition operand explained in 7.2.2 Operands related to performance (set format).

  You can also specify the number of pages in the buffer for local work tables in the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition and the export option `adb_export_wrktbl_blk_num`.

  For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*. For details about the export option `adb_export_wrktbl_blk_num`, see *Format of export options* in

Specification format for the adbexport command under *adbexport (Export Data)* in the manual *HADB Command Reference*.

`-g` *global-buffer-name*

~<global buffer name> ((1 to 30 bytes))

Specify a global buffer name that is unique within the HADB server.

The naming rules for global buffer names are as follows:

- The permitted characters include single-byte numeric characters, single-byte uppercase letters, single-byte lowercase letters, half-width katakana characters, single-byte spaces, single-byte underscores (_), and single-byte hyphens (-). Double-byte characters are also permitted. Double-byte spaces are not permitted.
- Single-byte characters can be intermixed with double-byte characters.
- Single-byte uppercase letters are always distinguished from single-byte lowercase letters.
- The first character must be a single-byte uppercase letter, a single-byte lowercase letter, or a half-width uppercase katakana character (ｱ to ﾝ, ﾜ). Alternatively, specify a double-byte character.
- The last character cannot be a single-byte space.
- To include a single-byte space or a single-byte hyphen (-) in the a global buffer name, enclose the entire global buffer name in `\"` (backslash followed by a double quotation mark). Then, enclose the entire name in double quotation marks (").

  Example

```
adbbuff -g "\"Sample Buffer01\"" -n AREA01 -p 1024
adbbuff -g "\"Sample-Buffer02\"" -n AREA02 -p 1024
```

Do not specify in this option a name that includes `##ADBOTHER`. Names that include `##ADBOTHER` are sometimes used by the HADB server for allocating global buffers automatically. If you specify a name that includes `##ADBOTHER`, it might not be possible to differentiate global buffers based on their names.

Tip

- **Choice of the global buffer allocation method (`-n` or `-o` option)**

  To specify the DB area to which you want to allocate the global buffer, specify the `-n` option in normal cases. To allocate the global buffer to all DB areas that are not specified in the `-n` option, specify the `-o` option.

`-n` *DB-area-name*`[,`*DB-area-name*`]...`

~<character string> ((1 to 30 bytes))

Specify the names of the DB areas to which the global buffer are to be allocated.

For details about the naming rules for DB area names, see *Specifying names* in the manual *HADB SQL Reference*.

Note the following when specifying this option:

- **When specifying multiple DB area names**

  By specifying multiple DB area names in this option, you can allocate multiple DB areas to a single global buffer. When specifying multiple DB area names, specify only DB areas having the same page size. If you specify DB areas having different page sizes, an error occurs.
- **When specifying a data DB area that stores a column store table**

When specifying a data DB area that stores a column store table, we recommend that you define an adbbuff operand that specifies only the name of this data DB area in this option. That is, specify this option in such a way that a dedicated global buffer is assigned to data DB areas that store column store tables.

Assigning a data DB area that stores a row store table or index to the same global buffer as a data DB area that stores a column store table might impact retrieval performance. That is, retrieval from the row store table or retrieval using the index might be slower.

- **When specifying a dictionary DB area**

  If a message is output asking you to allocate a global buffer to a dictionary DB area, define a new adbbuff operand with only ADBDIC specified for this option. Normally, there is no need to specify a dictionary DB area in this option.

- **When specifying a system-table DB area**

  If a message is output asking you to allocate a global buffer to the system-table DB area, define a new adbbuff operand with only ADBSTBL specified for this option. Normally, there is no need to specify a system-table DB area in this option.

If no allocation of global buffers is made to a DB area by means of the -n option, the same global buffer as when the -o option is specified is allocated automatically to that DB area. In this case, ##ADBOTHER is specified for the -g option.

-o

Specifies that global buffers are to be allocated to all DB areas that are not specified in the -n option.

You can only have one adbbuff operand that specifies this option.

When this option is specified, DB areas that have the same page size are grouped together and a global buffer is allocated to each such group. The name of each global buffer is -g *global-buffer-name* + #*nnnnnnnnnn* (where *nnnnnnnnnn* is the page size expressed as a 10-digit decimal number). An example follows.

Example:

- Specification of adbbuff operand

```
adbbuff -g gbuf01 -o
```

- DB areas that are not allocated to global buffers in the -n option.

  - DB areas with page sizes of 4 KB: ADBDIC, ADBSTBL, ADBU00011, and ADBU00012

  - DB area with page sizes of 32 KB: ADBU00013

Explanation

In this case, the HADB server creates the following two global buffers:

- gbuf01#0000004096

  ADBDIC, ADBSTBL, ADBU00011, and ADBU00012 are allocated to this global buffer.

- gbuf01#0000032768

  ADBU00013 is allocated to this global buffer.

If global buffers are allocated to all DB areas by using the -n option, specification of the -o option is ignored.

-p *number-of-pages-in-global-buffer*

~<integer> ((0 to 2,147,483,647)) <<0>>

Specify the number of pages in the global buffers to be allocated to DB areas.

If you omit this option or specify 0 for it, the value calculated using the following formula is assumed:

Formula

number-of-pages-in-global-buffer =
(*value-specified-for-adb_sys_max_users-operand* + *value-specified-for-adb_sys_rthd_num-operand* +
10 + *multi_num*#) × (*value-specified-for-adb_sys_uthd_num-operand* + 2)

Explanation of variables:
multi_num: *number-of-host-names-specified-for-adb_sys_multi_node_info-operand-in-server-definition* × 2 + 3

#
Add *multi_num* when you use the multi-node function.

`-a` *number-of-pages-in-global-buffer-dedicated-to-range-indexes*

~<integer> ((0 to 2,147,483,647))

This option is related to range indexes.

Specify the number of pages in the global buffer to be used exclusively for range indexes.

We recommend that you specify the value determined from the following formula. For a multi-chunk table, use the following formula to determine the value for each chunk. Then, specify in this option the sum of the values you determined.

Formula

$$\sum_{i=1}^{rngidx\_num} \left(SGRI(i) \times SEGSIZE(i) + \mathrm{MIN}\left(SAUSEDPAGENUM(i), SEGSIZE(i) \right) + \left\downarrow \frac{SAUSEDPAGENUM(i)}{SEGSIZE(i)} \right\downarrow \times SEGSIZE(i) \right)$$

rngidx_num

Number of range indexes that are stored in the DB area specified in this `adbbuff` operand

SGRI(i)

See (1) Determining the SGRI variable in 5.8.6 Determining the number of segments for storing each range index.

SEGSIZE(i)

Segment size in the DB area that stores range indexes (pages)

Use the following formula to determine its value:

$$SEGSIZE = 4{,}194{,}304 \div page\_size_{(i)}$$

page_size(i)

Page size of the DB area that stores range indexes (bytes)

SAUSEDPAGENUM(i)

Number of pages used by each range index (pages)

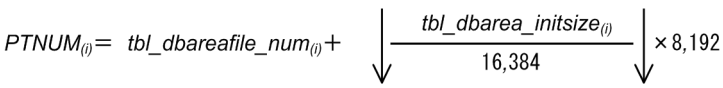
Use the following formula to determine its value:

$$SAUSEDPAGENUM_{(i)} = \left\uparrow \frac{PTNUM_{(i)} \times 65}{(page\_size_{(i)} - 64) \times 8} \right\uparrow + 1$$

PTNUM(i)

Number of pointers used by each range index (pointers)

Use the following formula to determine its value:



tbl_dbareafile_num(i)

Number of DB area files that store tables for which range indexes are defined (files)

tbl_dbarea_initsize(i)

Initial allocation size of the DB area that stores tables for which range indexes are defined (gigabytes)

Substitute the value that was specified for the initial allocation size option when one of the following commands was executed for the target DB area.

- The value specified for the `-i` option, which is an initialization option of the `adbinitdbarea` operand of the `adbinit` command
- The value specified for the `-i` option, which is a DB area addition and modification option of the `adbaddarea` operand of the `adbmodarea` command
- The value specified for the `-i` option, which is a DB area addition and modification option of the `adbexpandarea` operand of the `adbmodarea` command

Note

- For details about the initialization options of the `adbinit` command, see *Format of initialization options* in *Specification format for the adbinit command* under *adbinit (Initialize the Database)* in the manual *HADB Command Reference*.
- For details about the DB area addition and modification options of the `adbmodarea` command, see *Format of DB area addition and modification options* in *Specification format for the adbmodarea command* under *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

Note the following points:

- The global buffer pages specified by the `-a` option are allocated separately from the global buffer pages specified by the `-p` option.
- If you specify the `-a` option, range indexes do not use the global buffer pages specified by the `-p` option.
- If you omit the `-a` option, range indexes use the global buffer pages specified by the `-p` option.

`-v` *memory-size-used-for-table-scan-buffer* `[,` *maximum-memory-size-used-per-real-thread* `]`

Specifying this option allocates a table scan buffer to improve the processing speed of table scans.

The allocated table scan buffer is applied to the following segments:

- Basic row segment (for a table in row store format)
- Column-data segment (for a table in column store format)

Application targets are only the preceding segments.

Important

Specify this option when you want to allocate a table scan buffer in HADB server version **04-01** or later. Do not specify the `-k` option.

If you omit both this option and the `-k`, the system does not allocate a table scan buffer.

memory-size-used-for-table-scan-buffer:

~<integer> ((50 to 100,000,000)) (megabytes)

Specify, in megabytes, the amount of memory to assign to the table scan buffer used during execution of a table scan.

The memory to be assigned to the table scan buffer is allocated according to the value specified in this option when the HADB server starts.

The table scan buffer is designed to improve the table scan processing speed. Batch-reading multiple pages from a database into the table scan buffer can reduce the number of I/O operations. The HADB server accesses the pages in the table scan buffer at the page level, in the same manner as those in the global buffer.

When you specify this option, a table scan buffer is allocated. This means that the table scan buffer rather than the global buffer is used when performing a table scan. Use of a table scan buffer might improve the execution speed of the following SQL statements:

- An SQL statement that becomes a simple table scan
- An SQL statement in which the outermost query specification becomes a table scan

If the table scan buffer allocated for table scanning is insufficient, global buffers are used.

If multiple table scans take place during execution of an SQL statement, the memory allocated to the real threads used to execute that SQL statement are re-used. For example, this applies to a situation when a table scan is executed for multiple tables. The memory allocated to the real threads that execute each SQL statement are deallocated when the cursor is closed.

When you specify this option, specify a value that is between the minimum memory size and the maximum memory size determined from following two formulas:

Formula (value specified for *memory-size-used-for-table-scan-buffer* in `-v` option)

Minimum memory size ≤ A ≤ maximum memory size

A = Value specified for *size-of-memory-used-for-table-scan-buffer* in the `-v` option
Minimum memory size = MAX(*sql_rthd_num* × *uthd_num* , 512)
Maximum memory size = MAX(*rthd_num* × *uthd_num* × *table_num* × 12 , 512)

Explanation of variables

sql_rthd_num
Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

uthd_num
Value specified for the `adb_sys_uthd_num` operand in the server definition

rthd_num
Value specified for the `adb_sys_rthd_num` operand in the server definition

table_num
Total number of tables defined in the DB area whose name is specified for the `-n` option of the `adbbuff` operand

Note

The memory size specified in this option includes the following two types of memory:

- Global buffer pages used to read data from the database
- Process common memory used to manage the table scan buffer

For details about the page size of the global buffer pages used to read data from the database, see the description of the variable *SCANPAGES* in (2) Determining the global buffer page requirement (for starting the HADB server) under 6.3.3 Determining the memory requirement for starting the HADB server.

maximum-memory-size-used-per-real-thread:

~<integer> ((50 to 100,000,000)) (megabytes)

Specify, in megabytes, the maximum memory size of the table scan buffer available to each real thread.

If you do not specify a value, no maximum memory size is set.

When you specify this option, when the memory allocated to the real threads that execute each SQL statement reaches the maximum memory size, the global buffer is used instead of the table scan buffer to fulfill further memory requirements.

If the number of concurrently executed SQL statements that include table scans is unknown, we recommend that you do not specify a value. This is to ensure that the table scan buffer can be used in the most efficient way.

If the number of concurrently executed SQL statements that include table scans is known and you want the table scan buffer to be split evenly among them, specify the value determined by the following formula:

Formula (value to specify for *maximum-memory-size-used-per-real-thread* in -v option)

```
Value specified for maximum-size-of-memory-used-per-real-thread in the -v option
 =
    value specified for size-of-memory-used-for-table-scan-buffer in the -v opti
on ÷
    (number of concurrently executed SQL statements that include table scans
      × value specified for the adb_sql_exe_max_rthd_num operand in the server d
efinition)
```

`-k` *number-of-sectors-in-table-scan-buffer-for-batch-loading-segments*

~<integer> ((1 to 20,000,000))

> **Important**
>
> There is no need to specify this operand in HADB server version **04-01** or later. Specify the `-v` option when you want to allocate a table scan buffer.
>
> You cannot specify this option and the `-v` option together. If you specify both options, the HADB server cannot start (the message `KFAA51223-E` is output).
>
> If you omit both this option and the `-v` option, the system does not allocate a table scan buffer.
>
> If you specify a value of 12 or lower for this option, the system does not allocate a table scan buffer (the message `KFAA51204-W` is output).

Specify the number of buffer sectors in the table scan buffer for batch-reading segments when a table scan is performed. The size of a sector in the table scan buffer is the same as the segment size, which is 4 megabytes.

The table scan buffer is designed to improve the table scan processing speed. Batch-reading multiple pages from a database into the table scan buffer at the segment level can reduce the number of I/O operations. This might improve the processing speed.

The HADB server accesses the pages in the table scan buffer, which were read in units of segments, on the same page-by-page basis as it accesses the pages in the global buffer.

Specifying this option allocates the table scan buffer. Therefore, the table scan buffer, rather than the global buffer, is used when a table scan is performed. Using the table scan buffer might improve the execution speed of the following SQL statements:

- An SQL statement that becomes a simple table scan
- An SQL statement in which the outermost query specification becomes a table scan

When you specify this option, specify a value that is between the minimum number of buffer sectors and the maximum number of buffer sectors determined from the two formulas shown in the following.

Formula (value specified in the -k option)

Minimum number of buffer sectors ≤ value specified for the -k option ≤ maximum number of buffer sectors

Minimum number of buffer sectors = MAX(↑*sql_rthd_num* × *uthd_num* ÷ 4↑ , 128)
Maximum number of buffer sectors = MAX(*rthd_num* × *uthd_num* × *table_num* × 3 , 128)

Explanation of variables

sql_rthd_num
Value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition

uthd_num
Value specified for the `adb_sys_uthd_num` operand in the server definition

rthd_num
Value specified for the `adb_sys_rthd_num` operand in the server definition

table_num
Total number of tables targeted for table scanning, among the tables defined in the DB area whose name is specified for the `-n` option of the `adbbuff` operand

When you specify this option, we recommend that you specify the server definition so that the formula shown below is satisfied. Make adjustments so that the values on the left and right sides of the equation are as close to each other as possible.

Formula

RTHD ≤ *MAXUSER* × *SQLTHD*

Explanation of variables:
RTHD: Value specified for the `adb_sys_rthd_num` operand
MAXUSER: Value specified for the `adb_sys_max_users` operand
SQLTHD: Value specified for the `adb_sql_exe_max_rthd_num` operand

If the table scan buffer allocated for table scanning is insufficient, global buffers are used. You can use the `adbstat` command to check whether a table scan buffer of sufficient size has been allocated. For details, see 13.2.5 Reducing the execution time of SQL statements that perform table scans.

For details about table scanning, see *How to retrieve tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

7.2.12 Operands and options related to the client-group facility (command format)

This subsection explains the operands and options that are related to the client-group facility.

The client-group facility enables you to set the following groups:

- Client groups
- Command groups

To set each group, specify the `adbcltgrp` operand in the server definition. You can specify multiple `adbcltgrp` operands that set client groups. Note that you can specify only one `adbcltgrp` operand that sets a command group.

Note

You can use the `adbls -d cltgrp` command to check the groups that have been set with the `adbcltgrp` operand. For details about the `adbls -d cltgrp` command, see *adbls -d cltgrp (Display Information on Client Groups and Command Groups)* in the manual *HADB Command Reference*.

The maximum number of `adbcltgrp` operands that can be specified is determined by the following formula.

Formula (count)

Maximum number of adbcltgrp operands# that can be specified in the server definition =
value specified for the adb_sys_max_users operand in the server definition + 1

#: This is the number of `adbcltgrp` operands in which a client group is specified plus the `adbcltgrp` operand in which the command group is specified.

Multi-node function

If you will be using the multi-node function, specify the same value in this operand for all nodes. However, you can specify different values for the `-r` and `-e` options. If you specify different values for nodes, determine the values of the `-r` and `-e` options according to the `adb_sys_rthd_num` operand value in the server definition.

Note

- For details about client groups and command groups, see 2.12.1 Overview of the client-group facility.
- For details about the `adb_sys_max_users` operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).
- For details about the `adb_sys_rthd_num` operand in the server definition, see the description of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

The following subsections explain the `adbcltgrp` operand and options in the server definition for each group that is set.

[52] `adbcltgrp` **(for setting a client group)**

Sets a client group. You can specify for the client group the numbers of connections and processing real threads that can be used by the HADB clients that belong to that group.

`-g` *client-group-name*

~<character string> ((1 to 30 bytes))

Specify a name for the client group that is unique within the HADB server.

The permitted characters include single-byte numeric characters, single-byte uppercase letters, single-byte lowercase letters, and single-byte underscores (_). The first character must be a single-byte uppercase letter, a single-byte lowercase letter, or a single-byte underscore (_).

The client group name specified in this option must be specified in the `adb_clt_group_name` operand in the client definition.

Note

For details about the `adb_clt_group_name` operand in the client definition, see *Operands related to system configuration* in the *HADB Application Development Guide*.

-m *maximum-number-of-concurrent-connections-for-the-specified-client-group*

~<integer> ((0 to 1,024)) <<value of adb_sys_max_users - sum of the -u option values in all other adbcltgrp operands>>

Specify the maximum number of concurrent connections for the specified client group.

The value specified in this option is the maximum number of concurrent connections usable by this group. This group will not use more connections than specified in this option.

If this option is omitted, the value determined by the following formula is assumed.

Formula

$A - B$

Explanation of variables:
A: Value specified for the `adb_sys_max_users` operand in the server definition
B: Total of the values specified for the `-u` option of the `adbcltgrp` operand in the server definition for groups other than the local group#

#: The value is 0 if the `adbcltgrp` operand in the server definition is not specified for groups other than the local group.

The value from this formula is also assumed if the specified value is greater than the value obtained from the formula.

For details about the `adb_sys_max_users` operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).

Important

If `0` is specified in this option, the HADB clients belonging to this group will not be able to connect to the HADB server.

-u *guaranteed-minimum-number-of-concurrent-connections-for-the-specified-client-group*

~<integer> ((0 to 1,024)) <<0>>

Specify the guaranteed minimum number of concurrent connections for the specified client group.

This option determines the minimum number of concurrent connections that will always be made available to this group. The specified number of concurrent connections are usable by the local group regardless of the connection status of other HADB clients and commands.

Specify in this option a value that does not exceed the value of the `-m` option. Note also that the specified value must satisfy the condition shown below (if these conditions are not satisfied, an error will occur when the HADB server starts).

Formula

$A + B \leq C$

Explanation of variables:
A: Value specified for this option
B: Total of the values specified for the `-u` option of the `adbcltgrp` operand in the server definition for groups other than the local group#
C: Value specified for the `adb_sys_max_users` operand in the server definition

#: The value is 0 if the `adbcltgrp` operand in the server definition is not specified for groups other than the local group.

For details about the `adb_sys_max_users` operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).

Important

If the *total-of-the-values-specified-for-the--u-option-of-the-adbcltgrp-operand-in-all-server-definitions* is the same as the *value-specified-for-the-adb_sys_max_users-operand-in-the-server-definition*, the following HADB clients will not be able to connect to the HADB server:

- HADB clients belonging to a group for which `0` is specified in the `-u` option of the `adbcltgrp` operand in the server definition
- HADB clients that belong to no group

`-r` *maximum-number-of-processing-real-threads-usable-by-the-specified-client-group*

~<integer> ((0 to 4,096) <<value of adb_sys_rthd_num - sum of the -e option values in all other adbcltgrp operands>>

Specify the maximum number of processing real threads usable by the specified client group.

The value specified in this option is the maximum number of processing real threads usable by this group. This group will not use more processing real threads than specified in this option.

If this option is omitted, the value determined by the following formula is assumed.

Formula

$A - B$

Explanation of variables:
A: Value specified for the `adb_sys_rthd_num` operand in the server definition
B: Total of the values specified for the `-e` option of the `adbcltgrp` operand in the server definition for groups other than the local group#

#: The value is 0 if the `adbcltgrp` operand in the server definition is not specified for other than the local group.

The value from this formula is also assumed if the specified value is greater than the value obtained from the formula.

For details about the `adb_sys_rthd_num` operand in the server definition, see the description of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

Important

If `0` is specified in this option, `0` is assumed for the value of the `adb_sql_exe_max_rthd_num` operand that is applied to the HADB clients belonging to this group. This means that those HADB clients will be able to use only connection threads to execute SQL statements.

For details about the `adb_sql_exe_max_rthd_num` operand in the server definition, see the description of the adb_sql_exe_max_rthd_num operand in 7.2.2 Operands related to performance (set format).

For details about the `adb_sql_exe_max_rthd_num` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

`-e` *guaranteed-minimum-number-of-processing-real-threads-usable-by-the-specified-client-group*

~<integer> ((0 to 4,096)) <<0>>

Specify the guaranteed minimum number of processing real threads usable by the specified client group.

This option determines the minimum number of processing real threads that will always be made available to this group. The specified number of processing real threads are always usable by the local group regardless of other HADB clients' and commands' processing real thread usage status.

Specify in this option a value that does not exceed the value of the -r option. Note also that the specified value must satisfy the condition shown below (if these conditions are not satisfied, an error will occur when the HADB server starts).

Formula

$A + B \leq C$

Explanation of variables:
A: Value specified for this option
B: Total of the values specified for the -e option of the adbcltgrp operand in the server definition for groups other than the local group#
C: Value specified for the adb_sys_rthd_num operand in the server definition

#: The value is 0 if the adbcltgrp operand in the server definition is not specified for groups other than the local group.

For details about the adb_sys_rthd_num operand in the server definition, see the description of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

Important

If the *total-of-the-values-specified-for-the--e-option-of-the-adbcltgrp-operand-in-all-server-definitions* is the same as the *value-specified-for-the-adb_sys_rthd_num-operand-value-in-the-server-definition*, 0 is assumed as the value of the adb_sql_exe_max_rthd_num operand for the following HADB clients:

- HADB clients belonging to a group for which 0 is specified in the -e option of the adbcltgrp operand in the server definition
- HADB clients that belong to no group

These HADB clients will always use only connection threads to execute SQL statements.

For details about the adb_sql_exe_max_rthd_num operand in the server definition, see the description of the adb_sql_exe_max_rthd_num operand in 7.2.2 Operands related to performance (set format).

For details about the adb_sql_exe_max_rthd_num operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

-w *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-the-specified-client-group*[,*reset-threshold-for-warning-message-output*]

Specify this option to output the warning message KFAA40020-W when the number of concurrent connections approaches the maximum number of concurrent connections specified in the -m option.

The value specified for the -w option is invalid if you specify 0 in the -m option (this includes situations in which 0 is assumed).

output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-client-group:

~<integer> ((0 to 100)) <<0>> (percent)

Specify the output threshold for the warning message KFAA40020-W as a proportion of the value specified for the -m option (the maximum number of concurrent connections applied to the client group). For example, if you specify 20 for the -m option and 90 as the output threshold, a warning message is output when the number of connections to the HADB server reaches 18 (which is 90% of 20).

If you specify 0 as the output threshold, the warning message will not be output.

Important

When groups share their freely usable number of connections with other groups (when different values are specified for the -m option and the -u option), those connections might be in use by the other groups. In this situation, an error resulting from the maximum number of concurrent connections being exceeded might occur before the warning message is output.

reset-threshold-for-warning-message-output:

~<integer> ((0 to 99)) (percent)

Specify the threshold for resetting the output state of the warning message KFAA40020-W as a proportion of the value specified for the -m option (the maximum number of concurrent connections applied to the client group). When the warning message has been output once, HADB registers a state in which the warning message has been output. In this state, the warning message is not output again until the number of connections to the HADB server has fallen below a specific value.

Note

The relationship between the value specified for the -w option and output of the warning message is as follows:

```
adbcltgrp -g ...
          -m 20
          -w 90,70
```

When the -m option and the -w option are specified in this way, the warning message is output when the number of connections to the HADB server reaches 18 (90% of 20). Subsequently, if the number of connections to the HADB server falls to 17 before rising again to 18, HADB will not output the warning message again.

However, if the number of connections to the HADB server falls to 14 (70% of 20) or below, the state in which the warning message has been output is reset. This means that subsequently, the warning message will be output again when the number of connections to the HADB server reaches 18 or higher.

Notes regarding the threshold for resetting the warning message output state are as follows:

- If you do not specify a reset threshold, a value equivalent to *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-client-group* - 30 is assumed. If this results in a negative value, 0 is assumed.
- Specify the operand so that *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-client-group* > *reset-threshold-for-warning-message-output*. If this condition is not satisfied, the reset threshold specification is invalid (the warning message KFAA41000-W is output). In this case, a value equivalent to *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-client-group* - 30 is assumed.
- If you specify 0 for *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-client-group*, the reset threshold specification is invalid.
- HADB clients that do not belong to a client group are subject to the value specified for the adb_sys_max_users_wrn_pnt operand.
- If you define client groups, a warning message is not output even if the system as a whole exceeds the ratio specified in the adb_sys_max_users_wrn_pnt operand. The warning message is only output when a client group exceeds the ratio specified for that client group.

Multi-node function

The warning message KFAA40020-W is output on the master node.

[53] adbcltgrp (for setting a command group)

Sets a command group. You can specify for the command group the numbers of connections and processing real threads that can be used by the commands belonging to that group. This operand can be specified only once (only one command group can be set).

-g command

Sets a command group.

When an adbcltgrp operand that includes this option is specified, all *commands that connect to the HADB server* will belong to the command group that is set up. For details about the commands that connect to the HADB server, see *List of commands* in the manual *HADB Command Reference*.

> **Note**
>
> In addition to the *commands that connect to the HADB server*, HADB clients can also belong to the command group. To specify an HADB client as a member of the command group, specify command in the adb_clt_group_name operand in the client definition.
>
> For details about the adb_clt_group_name operand in the client definition, see *Operands related to system configuration* in the *HADB Application Development Guide*.

-m *maximum-number-of-concurrent-connections-for-the-specified-command-group*

~<integer> ((0 to 1,024)) <<value of adb_sys_max_users - sum of the -u option values in all other adbcltgrp operands>>

Specify the maximum number of concurrent connections for the specified group.

The value specified in this option is the maximum number of concurrent connections usable by this group. This group will not use more connections than specified in this option.

If this option is omitted, the value determined by the following formula is assumed.

Formula

$A - B$

Explanation of variables:
A: Value specified for the adb_sys_max_users operand in the server definition
B: Total of the values specified for the -u option of the adbcltgrp operand in the server definition for groups other than the local group#

#: The value is 0 if the adbcltgrp operand in the server definition is not specified for groups other than the local group.

The value from this formula is also assumed if the specified value is greater than the value obtained from the formula.

For details about the adb_sys_max_users operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).

> **Important**
>
> If 0 is specified in this option, the commands belonging to this group will not be able to connect to the HADB server.

-u *guaranteed-minimum-number-of-concurrent-connections-for-the-specified-command-group*

~<integer> ((0 to 1,024)) <<0>>

Specify the guaranteed minimum number of concurrent connections for the specified command group.

This option determines the minimum number of concurrent connections that will always be made available to this command group. The specified number of concurrent connections are usable by the command group regardless of the connection status of other HADB clients and commands.

Specify in this option a value that does not exceed the value of the `-m` option. Note also that the specified value must satisfy the condition shown below (if these conditions are not satisfied, an error will occur when the HADB server starts).

$A + B \leq C$

Explanation of variables:
A: Value specified for this option
B: Total of the values specified for the `-u` option of the `adbcltgrp` operand in the server definition for groups other than the local group#
C: Value specified for the `adb_sys_max_users` operand in the server definition group.

#: The value is 0 if the `adbcltgrp` operand in the server definition is not specified for groups other than the local group.

For details about the `adb_sys_max_users` operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).

Important

If the *total of the values specified for the -u option of the adbcltgrp operand in all server definitions* is the same as the *value specified for the adb_sys_max_users operand in the server definition*, the following commands will not be able to connect to the HADB server:

- Commands belonging to a group for which `0` is specified in the `-u` option of the `adbcltgrp` operand in the server definition
- Commands that belong to no group

`-r` *maximum-number-of-processing-real-threads-usable-by-the-specified-command-group*

~<integer> ((0 to 4,096) <<value of adb_sys_rthd_num - sum of the -e option values in all other adbcltgrp operands>>

Specify the maximum number of processing real threads usable by the specified command group.

The value specified in this option is the maximum number of processing real threads usable by this group. This group will not use more processing real threads than specified in this option.

If this option is omitted, the value determined by the following formula is assumed.

Formula

$A - B$

Explanation of variables:
A: Value specified for the `adb_sys_rthd_num` operand in the server definition
B: Total of the values specified for the `-e` option of the `adbcltgrp` operand in the server definition for groups other than the local group#

#: The value is 0 if the `adbcltgrp` operand in the server definition is not specified for groups other than the local group.

The value from this formula is also assumed if the specified value is greater than the value obtained from the formula.

For details about the `adb_sys_rthd_num` operand in the server definition, see the description of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

When you set a command group and you will be executing commands listed under *Targeted commands* in the explanation of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format), specify a value in this option that is at least the *minimum number of processing real threads required*. If you do not specify a

value at least equal to the *minimum number of processing real threads required*, execution of the command might result in an error. For details about the *minimum number of processing real threads required* to execute each command, see Table 6-28: Operands and command options for specifying the number of processing real threads to be used for command execution in (2) Operands and command options for specifying the number of processing real threads to be used for command execution under 6.23.2 Points to consider about the number of processing real threads to be used during command execution.

Important

If `0` is specified in this option, `0` is assumed for the value of the `adb_sys_rthd_num` operand that is applied to the commands belonging to this command group. This means that an error will result when those commands are executed, due to insufficient processing real threads.

`-e` *guaranteed-minimum-number-of-processing-real-threads-usable-by-the-specified-command-group*

~<integer> ((0 to 4,096)) <<0>>

Specify the guaranteed minimum number of processing real threads usable by the specified command group.

This option determines the minimum number of processing real threads that will always be made available to this group. The specified number of processing real threads are always usable by the local group regardless of other HADB clients' and commands' processing real thread usage status.

Specify in this option a value that does not exceed the value of the `-r` option. Note also that the specified value must satisfy the condition shown below (if these conditions are not satisfied, an error will occur when the HADB server starts).

Formula

$A + B \leq C$

Explanation of variables:
A: Value specified for this option
B: Total of the values specified for the `-e` option of the `adbcltgrp` operand in the server definition for groups other than the local group#
C: Value specified for the `adb_sys_rthd_num` operand in the server definition

#: The value is 0 if the `adbcltgrp` operand in the server definition is not specified for groups other than the local group.

For details about the `adb_sys_rthd_num` operand in the server definition, see the description of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

Important

If the *total of the values specified for the -e option of the adbcltgrp operand in all server definitions* is the same as the *value specified for the adb_sys_rthd_num operand value in the server definition*, `0` is assumed as the value of the `adb_sys_rthd_num` operand for the following commands:

- Commands belonging to a group for which `0` is specified in the `-e` option of the `adbcltgrp` operand in the server definition
- Commands that belong to no group

An error will result if these commands are executed, due to insufficient processing real threads.

`-w` *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-the-specified-command-group* `[,` *reset-threshold-for-warning-message-output* `]`

Specify this option to output the warning message `KFAA40020-W` when the number of concurrent connections approaches the maximum number of concurrent connections specified in the `-m` option.

The value specified for the -w option is invalid if you specify 0 in the -m option (this includes situations in which 0 is assumed).

output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-command-group:

~<integer> ((0 to 100)) <<0>> (percent)

Specify the output threshold for the warning message KFAA40020-W as a proportion of the value specified for the -m option (the maximum number of concurrent connections applied to the command group). For example, if you specify 20 for the -m option and 90 as the output threshold, a warning message is output when the number of connections to the HADB server reaches 18 (which is 90% of 20).

If you specify 0 as the output threshold, the warning message will not be output.

Important

When groups share their freely usable number of connections with other groups (when different values are specified for the -m option and the -u option), those connections might be in use by the other groups. In this situation, an error resulting from the maximum number of concurrent connections being exceeded might occur before the warning message is output.

reset-threshold-for-warning-message-output:

~<integer> ((0 to 99)) (percent)

Specify the threshold for resetting the output state of the warning message KFAA40020-W as a proportion of the value specified for the -m option (the maximum number of concurrent connections applied to the command group). When the warning message has been output once, HADB registers a state in which the warning message has been output. In this state, the warning message is not output again until the number of connections to the HADB server has fallen below a specific value.

Note

The relationship between the value specified for the -w option and output of the warning message is as follows:

```
adbcltgrp -g ...
          -m 20
          -w 90,70
```

When the -m option and the -w option are specified in this way, the warning message is output when the number of connections to the HADB server reaches 18 (90% of 20). Subsequently, if the number of connections to the HADB server falls to 17 before rising again to 18, HADB will not output the warning message again.

However, if the number of connections to the HADB server falls to 14 (70% of 20) or below, the state in which the warning message has been output is reset. This means that subsequently, the warning message will be output again when the number of connections to the HADB server reaches 18 or higher.

Notes regarding the threshold for resetting the warning message output state are as follows:

- If you do not specify a reset threshold, a value equivalent to *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-command-group* - 30 is assumed. If this results in a negative value, 0 is assumed.
- Specify the operand so that *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-command-group* > *reset-threshold-for-warning-message-output*. If

this condition is not satisfied, the reset threshold specification is invalid (the warning message KFAA41000-W is output). In this case, a value equivalent to *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-command-group* - 30 is assumed.

- If you specify 0 for *output-threshold-for-warning-message-regarding-maximum-number-of-concurrent-connections-for-specified-command-group*, the reset threshold specification is invalid.
- Commands that do not belong to a command group are subject to the value specified for the adb_sys_max_users_wrn_pnt operand.
- If you define a command group, a warning message is not output even if the system as a whole exceeds the ratio specified in the adb_sys_max_users_wrn_pnt operand. The warning message is only output when the command group or a client group exceeds the specified ratio.

7.3 Syntax rules for the server definition

This section explains the syntax rules that apply when creating or modifying a server definition.

7.3.1 Details of syntax rules for the server definition

The following explains in detail the syntax rules that apply when creating or modifying the server definition.

For details about how to create and modify the server definition, see 8.5 Creating and modifying a server definition.

For details about the specification format of the operands in the server definition and the values to be specified, see the following topics:

- 7.1 Specification formats for server definition operands
- 7.2 Detailed descriptions of the server definition operands

(1) Description sequence

Definitions in the server definition can be entered in any order.

(2) Description format

The server definition can be entered in either of the following two formats:

- Set format
- Command format

Explanations of these formats follow:

(a) Set format

In the set format, you specify a *value* for an *operand*.

Format

```
set operand = value
```

operand:

Operand name in the set format

value:

Specified value that corresponds to the operand name. This consists of an alphanumeric character string.

A specification example follows.

Specification example

```
set adb_sys_max_users = 1024
```

If the same operand is specified more than once, the value specified for the first one takes effect, and all subsequent values are ignored.

(b) Command format

The specification format for the command format follows.

Format

```
operand-name option
```

operand-name:

Operand name in the command format

option:

A character string that begins with a hyphen (−). There are two formats, Format 1 and Format 2.

Format 1

```
option
```

Format 2

```
option option-argument
```

option:

Consists of a hyphen (−) and a letter. It is case-sensitive.

option-argument:

Argument for the option. This consists of an alphanumeric character string.

A specification example follows.

Specification example

```
adbbuff -g BUFFER -n ADBUTBL01 -p 1024
```

(3) Line-ending codes

The following line-ending codes can be used in server definitions:

- `LF` (line feed)
- `NL` (newline)
- `CR` + `LF` (carriage return + line feed)

Note that you cannot use CR (carriage return) alone. If CR is used alone, an error will occur when the HADB server is started.

(4) Line continuation

The maximum length of each line in the definition is 4,000 bytes. Enter any line that exceeds 4,000 bytes over multiple lines.

To indicate that multiple lines are continuations of a single entry, enter a backslash (\) at the end of each line to be continued, followed immediately by a line-ending code. A single-byte space at the beginning of a continuation line is considered to be a significant character and part of the definition.

However, if a backslash (\) is not followed by a line-ending code, the following line is not recognized as being a continuation. Specification examples follow.

Example in which lines are recognized as being continuous

```
adbbuff -g BUFFER \↓
        -n ADBU01
```

Example in which lines are not recognized as being continuous

```
adbbuff -g BUFFER \Δ ↓
        -n ADBU01
```

Legend:

↓: End of line

Δ: Single-byte space

(5) Character string recognition rule

To specify a double quotation mark (") inside a character string, specify \". In the following example, the character string is recognized as "AREA01".

Specification example

```
adbbuff -g BUFFER -n \"AREA01\" -p 1024
```

In some operands (such as the adbbuff operand), you must enclose a character string inside double quotation marks (") for HADB to differentiate between uppercase and lowercase letters. For details, see the description on the operand of interest.

(6) Comment

To enter a comment in the server definition, start the comment with a hash mark (#).

If a line begins with a hash mark (#) as shown in Format 1, the entire line is recognized as a comment.

Format 1

```
# comment...
```

To enter a comment following a definition, specify a single-byte space, and then a hash mark (#) as shown in Format 2. If you do not enter a single-byte space, the text following the hash mark will not be recognized as a comment.

Format 2

```
definitionΔ#comment...
```

Legend:

Δ: Single-byte space

If the hash mark (#) is located inside a character string enclosed in double quotation marks ("), the character string is not recognized as a comment. The hash mark (#) and the characters that follow it are treated as letters in the character string.

Example in which the hash mark (#) and what follows it are both treated as letters

```
set adb_db_path = "TESTDBΔ#abc"
```

Legend:

Δ: Single-byte space

If the hash mark (#) is placed in the middle of a line, the entire character string following the hash mark (#) is treated as a comment. Therefore, be careful about where you place comments if you want to continue an entry across multiple lines. In the following example, the continuation line is treated as part of the comment.

Example in which the continuation line is treated as part of the comment

```
adbbuff -g BUFFERΔ#comment \↓
        -n DBAREA01
```

Legend:

↓: End of line

Δ: Single-byte space

In the above case, enter Δ#*comment* after `-n DBAREA01`.

(7) BOM

You cannot use BOM (Byte Order Mark) in the server definition.

(8) Commas

When you specify multiple character strings for a set format value or command format option argument, use a comma (`,`) to separate the character strings. Do not enter a single-byte space or line-ending code between a leading character string and a comma. Otherwise, an error occurs during HADB server startup.

Comma specification examples follow.

Correct specification example

```
set adb_sql_prep_delrsvd_words = ABS,BEFORE,CALL,DATA

set adb_sql_prep_delrsvd_words = ABS,ΔBEFORE,ΔΔCALL,ΔΔΔDATA
```

When you specify multiple character strings, use a comma (`,`) to separate the character strings. A single-byte space can be entered between a leading comma and a character string.

Specification example 1 that will cause an error

```
set adb_sql_prep_delrsvd_words = ABSΔ,BEFOREΔ,CALLΔ,DATA
```

A single-byte space cannot be entered between a leading character string and a comma. Because this was done in the above example, the specification value is recognized as invalid and an error occurs.

Specification example 2 that will cause an error

```
set adb_sql_prep_delrsvd_words = ABS,BEFORE↓
                                 ,CALL,DATA
```

A line-ending code cannot be entered between a leading character string and a comma. An error occurs.

When entering a line-ending code, you must indicate that multiple lines are continuations of a single entry. Enter a backslash (\) at the end of each line to be continued, followed immediately by a line-ending code. Specification examples follow.

Specification examples (to indicate that multiple lines are continuations of a single entry)

```
set adb_sql_prep_delrsvd_words = ABS,BEFORE\↓
                                 ,CALL,DATA

set adb_sql_prep_delrsvd_words = ABS,BEFORE,\↓
                                 CALL,DATA
```

Enter a backslash (\) as the continuation character between a leading comma and a character string, followed immediately by a line-ending code. In both cases, the lines are recognized as continuations of a single entry.

Legend:

Δ: Single-byte space

↓: End of line

8 Building a System

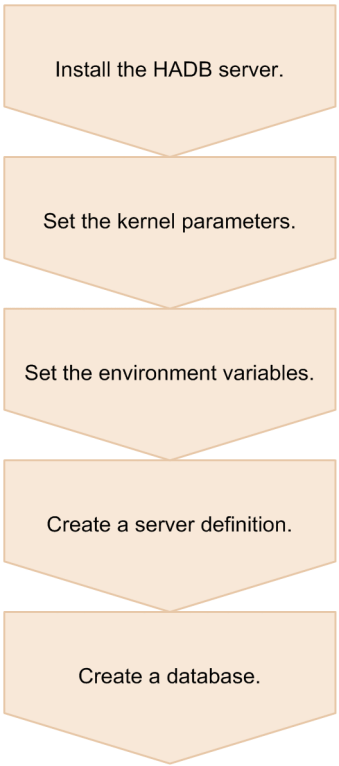
This chapter explains how to build an HADB system.

The procedures for building a hands-on environment are explained in 3. Guide for Building a Hands-on Environment. We recommend that you also read this chapter.

8.1 System construction procedure

This section explains how to build an HADB server system. For details about building an HADB client system, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*. The following figure shows the general procedure for building an HADB server system.

Figure 8-1: General procedure for building an HADB server system



Explanation

1. **Install the HADB server.**
   Install the HADB server. For details about how to install the HADB server, see 8.2 Installing the HADB server.
2. **Set the kernel parameters.**
   Set the kernel parameters. For details about setting up the kernel parameters, see 8.3 Setting kernel parameters.
3. **Set the environment variables.**
   Set the environment variables. For details about how to set the environment variables, see 8.4 Setting environment variables.
4. **Create a server definition.**
   Create a server definition. For details about creating a server definition, see 7. Designing the Server Definition and 8.5 Creating and modifying a server definition.
5. **Create a database.**
   Create a database. For details about the procedure for creating a database, see 9.1 Steps for creating a database.

8.2 Installing the HADB server

This section explains how to install the HADB server.

For details about how to install an HADB client, see *Installing and uninstalling an HADB client* in the *HADB Application Development Guide*.

8.2.1 Tasks that must be performed before installation

Before installing the HADB server, execute the following tasks as a superuser:

- Check the prerequisite libraries and user commands#
- Set up an HADB administrators group.
- Set up the HADB administrator.
- Set up an OS user who belongs to the HADB administrators group.
- Create a directory for storing the server directory.
- Change the settings of the directory for storing the server directory.
- Create the directories for storing communication-information files.
- Modify the access privileges to syslog.

#

This can be performed by a general user (an OS user without administrator privileges) instead of a superuser.

(1) Checking the prerequisite libraries and user commands

Make sure that the libraries and user commands necessary for the HADB server to operate correctly are installed in the OS.

▪ Check method

Execute the OS's `yum` command to check the packages installed in the OS. The following shows the command to be executed.

```
yum list installed
```

After checking the execution results of the preceding command, if all the packages listed in the following two tables are installed, the necessary libraries and user commands have been installed in the OS.

Table 8-1: List of packages to check (checking the prerequisite libraries)

No.	Package name and version to check	Prerequisite library for the HADB server included in the package
1	`glibc` (2.12 or later)	`libc.so.6`
2		`librt.so.1`
3		`libm.so.6`
4		`libpthread.so.0`
5		`ld-linux-x86-64.so.2` (loader for execution)
6		`libdl.so.2`

No.	Package name and version to check	Prerequisite library for the HADB server included in the package
7	`libaio` (0.3.107 or later)	`libaio.so.1`
8	`openssl` (1.0.0 or later)	`libcrypto.so.10`
9	`zlib` (1.2.3 or later)	`libz.so.1`
10	`libuuid` (2.17.2 or later)	`libuuid.so.1`
11	None	`linux-vdso.so.1#`

#

Because this library is a virtual shared library provided by kernels, you do not need to check whether the package exists.

Table 8-2: List of packages to check (checking the prerequisite user commands)

No.	Package name and version to check	User commands prerequisite for the HADB server contained in packages
1	`gdb` (7.2 or later)	`gcore`
2		`gstack`

If any package is not installed, install it in the OS. For details about how to install packages, see the OS documentation. You need to install the packages as a superuser.

Note

The following shows an example of executing the command to check whether a specific package has been installed.

Example

Check whether package `libaio` has been installed.

```
yum list installed | grep libaio
```

If package `libaio` is displayed in the execution result, package `libaio` has been installed. If package `libaio` is not displayed in the execution result, package `libaio` has not been installed.

(2) Setting up an HADB administrators group

Set up an HADB-only group (HADB administrators group) under the server machine's OS. Use the operating system's `groupadd` command to set up an HADB administrators group.

By setting up an HADB administrators group, you can restrict access to the files in the directories used to run the HADB server (server directory and DB directory) so that only OS users in the HADB administrators group can access them. This enhances the level of security.

■ HADB administrators group setup example

Set up `adbgroup` as an HADB administrators group under the server machine's OS.

```
groupadd adbgroup
```

You can also use an existing group that has been set up under the OS as an HADB administrators group.

(3) Setting up the HADB administrator

Once you have set up an HADB administrators group, set up the OS user who will manage the HADB server (HADB administrator) under the server machine's OS. Use the operating system's `useradd` command to set up the HADB administrator. Use a maximum of 32 bytes for the name of the HADB administrator.

An HADB administrator is a special user who is set up in the OS. The HADB administrator has the privileges needed to execute all HADB commands, and is the owner of files in the server directory and the DB directory. The OS user who is set up as the HADB administrator will be recognized as such by the HADB server when the HADB server is installed.

When you set the user name of the HADB administrator in the OS, you can also set up an HADB administrators group in the primary group to which the HADB administrator belongs. This gives the HADB administrator owner-level access privileges to the various types of files and directories on the HADB server, and prevents other OS users from being able to overwrite data.

Once you have set up the user name of the HADB administrator in the OS, you must set up the administrator's password. Use the operating system's `passwd` command to set up a password.

■ HADB administrator setup example

Set up `adbmanager` as the HADB administrator under the server machine's OS. When you do this, also set the HADB administrators group (`adbgroup`) as the primary group.

```
useradd -g adbgroup adbmanager
```

(4) Setting up the OS user who belongs to the HADB administrators group

Once you have set up the HADB administrator, we recommend that you set up the OS user who will manage the HADB server (OS user who belongs to the HADB administrators group). You set up this user separately from the HADB administrator under the server machine's OS.

The OS user who belongs to the HADB administrators group is given the privileges needed to execute HADB commands (excluding some commands). This OS user can also access the files in the server directory and the DB directory owned by the HADB administrator. Setting up the OS user who belongs to the HADB administrators group allows an OS user other than the HADB administrator to execute HADB commands, making HADB server operations more convenient.

Use the operating system's `useradd` command to set up the OS user who belongs to the HADB administrators group. Use a maximum of 32 bytes for the name of this OS user. Also, when you set up this user's user name under the OS, set the HADB administrators group as the primary group.

Once you have set up the user name of the OS user who belongs to the HADB administrators group under the OS, you must set a password. To do so, use the operating system's `passwd` command.

■ Example of setting an OS user who belongs to the HADB administrators group

Set up `adbgroupuser01` as the OS user who belongs to the HADB administrators group under the server machine's OS. When you do so, also set the HADB administrators group (`adbgroup`) as the primary group.

```
useradd -g adbgroup adbgroupuser01
```

Note

For details about the HADB commands that an OS user who belongs to the HADB administrators group can execute, see *List of commands* under *List of Commands and Common Rules* in the manual *HADB Command Reference*.

(5) Creating a directory for storing the server directory

Create a *directory for storing the server directory*, which is required to install the HADB server, as a superuser. Use the `mkdir` OS command to create a directory for storing the server directory.

Specify `/HADB` as the path name of the directory for storing the server directory.

■ Example of creating a directory for storing the server directory

The following shows how to create `/HADB` as a directory for storing the server directory.

```
mkdir /HADB
```

(6) Changing the settings of the directory for storing the server directory

After creating a directory for storing the server directory (`/HADB`), make the following changes to the settings as a superuser:

- Changing the owner and group of the directory for storing the server directory (`/HADB`)

  Change the owner of the directory to the HADB administrator (`adbmanager`). Also, change the group of the directory to the HADB administrators group (`adbgroup`). To change the owner and group of the directory, use the `chown` OS command.

- Assigning write permission to the directory for storing the server directory (`/HADB`)

  Assign write permission to the directory so that the HADB administrator (`adbmanager`) can write data to the directory. To assign write permission to the directory, use the `chmod` OS command.

■ Example of changing the settings of the directory for storing the server directory

The following shows how to change the owner and group of the directory for storing the server directory (`/HADB`) and assign write permission to the directory.

```
chown adbmanager:adbgroup /HADB
chmod 755 /HADB
```

(7) Creating the directories for storing communication-information files

Create the directories for storing communication-information files needed to operate the HADB server. Use the operating system's `mkdir` command to create these directories.

The directories for storing communication-information files vary depending on the version of the OS of the server machine on which the HADB server is installed.

- If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)

  See (a) Red Hat Enterprise Linux Server 6.

- If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)

  See (b) Red Hat Enterprise Linux Server 7.

(a) Red Hat Enterprise Linux Server 6

The following table shows the path names of the directories for storing communication-information files and the access privileges that are assigned.

Table 8-3: Information about the directories for storing communication-information files (for Red Hat Enterprise Linux Server 6)

No.	Path name of the directory to be created	Owner	Access privilege
1	/dev/HADB/pth	Superuser	777
2	/lib/udev/devices/HADB/pth		

▪ **Example of creating the directories for storing communication-information files**

Create directories (/dev/HADB/pth and /lib/udev/devices/HADB/pth) for storing communication-information files. When you do so, assign 777 as the access privilege.

```
mkdir -p -v -m 777 /dev/HADB/pth

mkdir -p -v -m 777 /lib/udev/devices/HADB/pth
```

(b) Red Hat Enterprise Linux Server 7

The following table shows the path names of the directories for storing communication-information files and the access privileges that are assigned.

Table 8-4: Information about the directories for storing communication-information files (for Red Hat Enterprise Linux Server 7)

No.	Path name of the directory to be created	Owner	Access privilege
1	/dev/HADB/pth	Superuser	777

In addition, you must create the following settings file.

Table 8-5: Information about the settings file (for Red Hat Enterprise Linux Server 7)

No.	Settings file to be created	Owner	Access privilege
1	/etc/tmpfiles.d/dev-HADB-pth.conf	Superuser	644 (default)

The following shows the information to be entered in the settings file (dev-HADB-pth.conf).

```
# Type Path Mode UID  GID  Age Argument
d /dev/HADB/pth 0777 root root - -
```

▪ **Example of creating a directory for storing communication-information files**

Create the directory (/dev/HADB/pth) for storing communication-information files. When you do so, assign 777 as the access privilege.

```
mkdir -p -v -m 777 /dev/HADB/pth
```

▪ **Example of creating a settings file**

Create the settings file (dev-HADB-pth.conf).

Execute the following OS command.

```
vi /etc/tmpfiles.d/dev-HADB-pth.conf
```

After executing the `vi` command, press the **I** key. Then, enter the following information.

```
# Type Path Mode UID  GID  Age Argument
d /dev/HADB/pth 0777 root root - -
```

When you finish entering the information, press the **Esc** key. Then, enter the following command, and then press the **Enter** key.

```
:wq
```

(8) Modifying the syslog access privilege

Modify the access privilege to syslog, to which HADB server messages are output. You can use the operating system's `chmod` command to modify this privilege.

Modifying the access privilege to syslog allows you to acquire syslog when you use the `adbinfoget` command to acquire troubleshooting information.

The following table shows the default path name of syslog and the access privilege that is assigned.

Table 8-6: Information about syslog whose access privilege is to be modified

No.	Default path name for syslog	Owner	Access privilege
1	`/var/log/messages`#	Superuser	`604`

#

If you have changed syslog, to which HADB server messages are output, to a destination other than `/var/log/messages`, assign the access privilege to syslog at the modified destination.

■ Example of modifying the syslog access privilege

This example changes the syslog access privilege (`/var/log/messages`) to `604`.

```
chmod 604 /var/log/messages
```

8.2.2 Installation procedure

This subsection explains how to install the HADB server. You install the HADB server as the OS user who was set up as the HADB administrator.

(1) Creating a directory for storing the installation data

Create a directory for storing the HADB server's installation data. To do this, enter the following OS command:

```
mkdir path-name-of-desired-directory
```

An example follows in which `/home/adbmanager/install` is selected as the path name for the desired directory.

• Command execution example

```
mkdir /home/adbmanager/install
```

(2) Granting write privileges for the directory that stores the installation data

Grant write privileges for the directory that will store the installation data so that the HADB administrator can perform write operations. To do this, enter the following OS command:

```
chmod 755 path-name-of-desired-directory
```

An example follows in which `/home/adbmanager/install` is selected as the path name for the desired directory.

• Command execution example

```
chmod 755 /home/adbmanager/install
```

(3) Mounting the file system CD-ROM

Allow the file system CD-ROM containing the installation command (`adbinstall` command) and the installation data (`tar.gz` file) for the HADB server to mount automatically.

If the file system CD-ROM does not mount automatically, you must mount it manually. To do so, enter the following command and press **Enter**:

```
mount /dev/cdrom /media
```

The underlined portion is the name of the mount directory for the file system CD-ROM. It will vary depending on the environment.

Important

Depending on the machine being used, the directory names and file names used on the CD-ROM might be different from those shown above. Enter the directory name that is displayed exactly as shown when you execute the `ls` operating system command.

(4) Copying the installation command (adbinstall command)

Copy the HADB server installation command (`adbinstall` command) that is stored in the mounted file system CD-ROM to a directory of your choice.

Enter the following operating system command to copy the `adbinstall` command to the desired directory:

```
cp /media/adbinstall path-name-of-desired-directory
```

The underlined portion is the name of the mount directory for the file system CD-ROM. It varies depending on the environment.

An example follows in which `/home/adbmanager/install` is selected as the directory path name.

• Command execution example

```
cp /media/adbinstall /home/adbmanager/install
```

(5) Copying the installation data (tar.gz file)

Copy to a directory of your choice the HADB server installation data (`tar.gz` file) that is stored on the mounted file system CD-ROM.

Enter the following operating system command to copy the `tar.gz` file to the directory of your choice:

```
cp /media/hitachi_advanced_data_binder_server-$VR.tar.gz
   path-name-of-desired-directory
```

The underlined portion is the name of the mount directory for the file system CD-ROM. It varies depending on the environment. `$VR` indicates the HADB version and release number.

> **Important**
>
> Copy the `tar.gz` file to the directory to which the `adbinstall` command was copied in (4) Copying the installation command (adbinstall command). Unless you store the `adbinstall` command and the `tar.gz` file in the same directory, you cannot install the HADB server.

The following provides an example in which `/home/adbmanager/install` is selected as the directory path name.

• Command execution example

```
cp /media/hitachi_advanced_data_binder_server-$VR.tar.gz
   /home/adbmanager/install
```

(6) Assigning execution privileges to the installation command (adbinstall command)

Assign execution privileges to the HADB server installation command (`adbinstall` command) that was copied to the directory of your choice, so that the HADB administrator has execution privileges.

An example follows in which the installation command (`adbinstall` command) is stored in `/home/adbmanager/install`.

• Command execution example

Execute the following operating system command to assign execution privileges to the installation command (`adbinstall` command):

```
chmod 777 /home/adbmanager/install/adbinstall
```

Execution privileges have now been assigned to the installation command (`adbinstall` command).

(7) Installing the HADB server

Install the HADB server by executing the HADB server installation command (`adbinstall` command), to which execution privileges were assigned.

Enter the HADB command shown below to install the HADB server. The directory under which the HADB server is installed becomes the *server directory*. The server directory stores a group of files related to server processes.

```
/install/adbinstall -s server directory path name
```

Note

The underlined portion is the path name of the directory that was selected to copy and store the `adbinstall` command.

The HADB server is installed under the directory specified in the `-s` option. This directory becomes the server directory. For the path name of the server directory, use a character string of no more than 118 bytes that begins with a forward slash (/).

For information about the characters that can be used in a server directory path, see <*path name*> under ■ Conventions: Syntax elements in the *Preface*.

In the `-s` option, you must specify a directory for which the HADB administrator has write privileges. If you specify a directory for which the HADB administrator does not have write privileges, an error occurs and the `KFAA91553-E` message is output.

The `KFAA91553-E` message is also output when the HADB administrator does not have write privileges for the installation command (`adbinstall` command) and the installation data (`tar.gz` file).

If the `KFAA91553-E` message is output, grant write privileges for the target directory.

If the directory specified in the `-s` option does not exist, execute the `adbinstall` command to automatically generate the directory.

Note

The OS user who executes the `adbinstall` command to install the HADB server is recognized as the HADB administrator by the HADB server.

If you execute the `adbinstall` command as `root`, not as the OS user specified as the HADB administrator, a warning message (`KFAA91558-W`) is output. Normally, the OS user specified as the HADB administrator executes the `adbinstall` command. Therefore, if the `KFAA91558-W` message is output, make sure that executing the `adbinstall` command by `root` does not cause any problems.

If a program exists, enter `n` or `N` in response to the input request in the `KFAA91559-Q` message that is output after the `KFAA91558-W` message. Then, execute the `adbinstall` command as the OS user specified as the HADB administrator.

If you execute the `adbinstall` command as a superuser other than `root`, the `KFAA91558-W` message is not output.

Note

`root` indicates a user for which `0` is displayed by executing the OS's `id -u` command. This includes a case where a value of `0` is displayed by executing the OS's `id -u` command after another OS user was changed to `root` by using the OS's `su` command.

> **Note**
>
> For details about the `adbinstall` command, see *adbinstall (Install HADB Server or Client)* in the manual *HADB Command Reference*.

An execution example follows in which `/home/adbmanager/install` is selected as the directory path name and `/HADB/server` is selected as the server directory path name.

• Command execution example

The HADB server is installed under the `/HADB/server` directory. The `/HADB/server` directory becomes the server directory.

```
/home/adbmanager/install/adbinstall -s /HADB/server
```

> **Note**
>
> For details about the configuration of the server directory when the HADB server is installed using the `adbinstall` command, see A.1 Configuration of the server directory (for installation).

8.2.3 Tasks that must be performed after installation

After installing the HADB server, perform the following tasks.

(1) Revising the scan target by using antivirus software

If antivirus software is installed on the server machine on which the HADB server is installed, revise the scan target.

If directories and files for use by the HADB server are specified for the scan target of antivirus software, the HADB server might not operate correctly or other problems might occur. Therefore, remove the following directories and files from the scan target of the antivirus software.

▪ Directories and files to be removed from the scan target

- Server directory
- DB directory
- Directory for storing communication-information files (`/dev/HADB/pth`)
- Directory to be specified for the `adb_core_path` operand in the server definition
- All directories that the commands of the HADB server access
- All archive directories
- Storage directory for synonym dictionary files (the directory to be specified for the `adb_syndict_storage_path` operand in the server definition)
- Audit trail directory (the directory to be specified for the `adb_audit_log_path` operand in the server definition)
- Multi-node synonym dictionary storage directory (the directory to be specified for the `adb_syndict_node_storage_path` operand in the server definition)
- Client directory

- Output-directory for common format audit trails (directory to be specified for the -d option of the adbconvertaudittrailfile command)
- Files that the ADB_CSVREAD function accesses
- Files that the ADB_AUDITREAD function accesses
- Conversion-target audit trail file to be specified in the adbconvertaudittrailfile command

(2) Setting IT Report Utility

If IT Report Utility (ITRU) is installed on the machine on which the HADB server is installed, specify settings so as to collect troubleshooting information via ITRU.

Important

The ITRU version supported by the HADB server depends on the OS version of the server machine.

- If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  The HADB server supports ITRU version 02-00 or later.
- If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  The HADB server supports ITRU version 03-00 or later.

The following describes the procedure for setting ITRU.

Procedure:

1. Log on to the server machine as root.
   If you are logged on as the HADB administrator, log off and then log on as root.
2. Copy the ITRU collection pattern definition file.
   From the CD-ROM file system containing the HADB server installation data, copy the ITRU collection pattern definition file (!8A9_HADB).
   Use the OS command to copy the ITRU collection pattern definition file (!8A9_HADB), and then store the file in the storage directory for ITRU collection pattern definition files (/etc/opt/hitachi/systoru/pattern).
   Command execution example

```
cp /media/!8A9_HADB /etc/opt/hitachi/systoru/pattern
```

   The underlined portion shows the name of the mount directory for the CD-ROM file system. It will vary depending on the environment.
3. Correct the contents of the ITRU collection pattern definition file.
   Use a text editor to open the copied ITRU collection pattern definition file (!8A9_HADB), and then correct the following locations.
   ▪ **Contents of the ITRU collection pattern definition file (before correction)**

```
FORMAT_VERSION=1.0

PP_NAME="HADB"
PP_GROUP="HADB"

# HADB
#
BLOCK {
  TARGET="config"
```

```
  TARGET="failure"
  TOOL_BLOCK {
    TIMEOUT=300
    COMMAND_LINE="XXX/adbsystoru -s XXX -l YYY -o \"%d%\""
    REDIRECT_PATH="%d%/HADB.txt"
  }
  DATA_BLOCK {
    DESCRIPTION="HADB"
    FILE_TYPE="directory"
    PATH_NAME="%d%"
  }
}
```

Correct the preceding information for XXX and YYY.

▪ Contents of the ITRU collection pattern definition file (after correction)

```
   :
COMMAND_LINE="/HADB/server/adbsystoru -s /HADB/server -l UTF8 -o \"%d%\""
   :
```

- **Information for XXX**

  Specify the absolute path to the server directory. Specify the same information as that specified for the ADBDIR environment variable. In this example, /HADB/server is specified.

- **Information for YYY**

  Specify the character encoding to be used by the HADB server. Specify the same information as that specified for the ADBLANG environment variable. In this example, UTF8 is specified.

For details about the environment variables ADBDIR and ADBLANG, see 8.4 Setting environment variables.

4. Save the results of editing the ITRU collection pattern definition file.

   Save the information edited in step 3, and then close the ITRU collection pattern definition file (!8A9_HADB).

This completes the setting of the ITRU collection pattern definition file.

8.3 Setting kernel parameters

This section explains how to set the kernel parameters.

You set the kernel parameters on the server machine on which the HADB server has been installed. The following describes the general procedure for setting the kernel parameters.

General procedure for setting the kernel parameters

1. Determine the value to be specified for each kernel parameter.
2. Specify those values in the settings file.

For details about the settings file and how to determine the values to specify for each kernel parameter, see 6.2 Estimating the kernel parameters. For details about how to set the kernel parameters, see the OS documentation.

8.4 Setting environment variables

This section explains the environment variables that will be set by HADB administrators and the OS users that belong to the HADB administrators group.

Make sure that the values that you set are valid in the shell when the HADB server is used. For details about how to set the environment variables, see the documentation for the operating system you are using.

If you are using the multi-node function, set the environment variables on all nodes.

Important

The HADB server must be stopped before you set environment variables. You can stop the HADB server by executing the `adbstop` command.

The following table shows the values to be specified in the environment variables.

Table 8-7: Values to be specified in the environment variables

No.	Environment variable	Value to be specified
1	`LANG`	Specify the character encoding for the operating system.# **When the HADB server is used in a Japanese language environment:** Select either of the following character encodings: • Unicode (UTF-8) Specify `ja_JP.UTF-8` in this environment variable. • Shift-JIS Specify `ja_JP.SJIS` in this environment variable. **When the HADB server is used in an English language environment:** Unicode (UTF-8) is used as the character encoding. Specify `en_US.UTF-8` in this environment variable.
2	`LD_LIBRARY_PATH`	Add the following directories to this environment variable: • `$ADBDIR/lib` When either of the following conditions are satisfied, also add `$ADBDIR/client/lib` to this environment variable: • The `adbsql` command will be executed on the HADB server • Applications that use the CLI function will be developed or executed on the HADB server
3	`PATH`	Add the following directories to this environment variable: • `$ADBDIR/bin` • `$ADBDIR/client/bin`
4	`TZ`	Specify the time zone of the machine on which the HADB server is to be installed. Do not specify a time zone that supports leap seconds. Multi-node function If you are using the multi-node function, specify the same value for this environment variable on all nodes.
5	`ADBDIR`	This environment variable is used to specify the absolute path to the server directory.
6	`ADBLANG`	Specify the character encoding to be used by the HADB server.# **When the HADB server is used in a Japanese language environment:** Select either of the following character encodings: • Unicode (UTF-8)

No.	Environment variable	Value to be specified
		Specify UTF8 in this environment variable. • Shift-JIS Specify SJIS in this environment variable. If you specify SJIS, take one of the following actions to prevent multi-byte Shift-JIS characters from being output to syslog: • Apply the extended syslog function to the HADB server and then convert character encoding for syslog (convert from Shift-JIS to Unicode). For details about the extended syslog function, see 10.4.6 Converting character encoding and improving reliability for syslog (Applying the extended syslog function). • Do not use multi-byte characters when you specify the following items: • Names specified in definition SQL statements (such as table names, index names, and column names) • Directory names and file names that will be used by HADB • Specify 0 in the ADBSYSLOGLV environment variable. When 0 is specified in ADBSYSLOGLV, no HADB messages will be output to syslog. Note that all HADB messages are output to the message log files. **When the HADB server is used in an English language environment:** Unicode (UTF-8) is used as the character encoding. Specify UTF8 in this environment variable. Multi-node function If you are using the multi-node function, specify the same value for this environment variable on all nodes.
7	ADBMSGLOGSIZE	Specify, in megabytes, the maximum size of each server message log file. A value between 1 and 2,000 can be specified. The HADB server creates four server message log files. If this environment variable is omitted, the maximum size of each server message log file is 16 megabytes.
8	ADBSYSLOGLV	If this environment variable is specified, you can suppress the messages that are output to syslog based on the output level. To suppress the messages that are output to syslog, specify this environment variable. Specify a value between 0 and 6 for the output level. For a list of the output levels of messages output to syslog, see 10.4.5 Suppressing message output to syslog. An error occurs if a value outside the range of 0 to 6 is specified for the output level. If this environment variable is not specified, 6 is assumed for the output level.
9	ADBCLTDIR	Specify the directory in which the HADB client's various types of files are to be stored. Specify the absolute path to the server directory for this environment variable. Set this environment variable when either of the following conditions are satisfied: • The adbsql command will be executed on the HADB server • Applications that use the CLI function will be developed or executed on the HADB server
10	ADBCLTLANG	Specify the character encoding to be used by HADB clients.# **When the HADB server is used in a Japanese language environment:** Select either of the following character encodings: • Unicode (UTF-8) Specify UTF8 in this environment variable. • Shift-JIS Specify SJIS in this environment variable. **When the HADB server is used in an English language environment:** Unicode (UTF-8) is used as the character encoding. Specify UTF8 in this environment variable. Set this environment variable when either of the following conditions are satisfied: • The adbsql command will be executed on the HADB server

No.	Environment variable	Value to be specified
		• Applications that use the CLI function will be developed or executed on the HADB server
11	ADBSQLNULLCHAR	If the adbsql command's retrieval results might contain a null value, specify in this environment variable the character string to be used to indicate the null value (character string displaying a null value). A character string consisting of 0 to 32 bytes is permitted. If a zero-byte character string is specified, the space is displayed as the null value. If this environment variable is omitted, the asterisk (*) is displayed as the null value. Specify this environment variable when retrieval data might contain asterisks or if you want to display a desired character as the null value. Note that if a multi-byte character is specified, the retrieval results might not be displayed correctly. Consider setting this environment variable when the adbsql command will be executed on the HADB server.
12	CLASSPATH	Specify this environment variable if you plan to develop or execute Java applications on the HADB server (when using the JDBC driver). Specify the absolute path to the JAR file for this environment variable. Specify the following path: • $ADBDIR/client/lib/adbjdbc8.jar If you change the preceding storage location for the JAR file, specify the absolute path of the new storage location. To work directly with classes provided by the JDBC driver, such as with non-standard JDBC methods provided by the JDBC driver (for example, connection information setup and acquisition interface methods), specify this environment variable before compiling the application.

#

Select the same character encoding for the LANG, ADBLANG, and ADBCLTLANG environment variables.

Example 1: Unicode (UTF-8) is used in a Japanese language environment

```
LANG="ja_JP.UTF-8"
ADBLANG=UTF8
ADBCLTLANG=UTF8
```

Example 2: Unicode (UTF-8) is used in an English language environment

```
LANG="en_US.UTF-8"
ADBLANG=UTF8
ADBCLTLANG=UTF8
```

8.5 Creating and modifying a server definition

This section explains how to create and modify a server definition.

> **Note**
>
> For details about how to create and modify a client definition, see *Creating a client definition* in *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

8.5.1 Server definition creation method and storage destination

You specify the execution environment for the HADB server in a server definition.

The following procedure describes how to create a server definition and where to store it.

Procedure

1. Copy the server definition template.
   Copy the server definition template file (`server.def`) stored under the server directory (`$ADBDIR/sample/conf/`), and store it under the server directory (`$ADBDIR/conf`).
2. Edit the copied server definition template file.
   Open the server definition template file (`server.def`) that you copied to the `$ADBDIR/conf` directory in step 1 in a text editor, and edit the file.
   For details about the format and contents of the operands specified in the server definition, and the syntax rules for the server definition, see 7. Designing the Server Definition.
3. Save the server definition by overwriting the existing one.
   When you have finished editing the server definition file (`$ADBDIR/conf/server.def`), save it by overwriting the existing file. Saving the file as `server.def` under the `$ADBDIR/conf` directory allows server definitions to be loaded when the HADB server starts.

8.5.2 Modifying the server definition

This subsection explains how to modify the contents defined in the server definition.

Before modifying the contents defined in the server definition, carefully consider the impact on the HADB server.

Storage location

Server definition file: `$ADBDIR/conf/server.def`

Procedure

1. Terminate the HADB server.
   Terminate the HADB server by executing the `adbstop` command.
2. Modify the defined content of the server definition.
   Open the server definition file (`$ADBDIR/conf/server.def`) in a text editor and modify the defined content.

For details about the format and contents of the operands specified in the server definition, and the syntax rules for the server definition, see 7. Designing the Server Definition.

3. Start the HADB server.

   After you have modified the contents of the server definition, restart the HADB server by executing the `adbstart` command.

> **Important**
>
> The changes to the server definition are not applied before the HADB server is restarted.

8.6 Upgrading the HADB server version

This section explains how to upgrade the HADB server version.

To replace the HADB server with a revised version rather than upgrading the HADB server version, see 8.8 Swapping the HADB server with its revised version. 8.8 Swapping the HADB server with its revised version also explains the nature of revised versions of the HADB server.

If you temporarily upgrade the version of the HADB server for test purposes, such as checking operational behavior of applications, and then restore the previous version, see 8.9 Temporarily upgrading the HADB server version and then downgrading it (test for checking operational behavior).

8.6.1 Steps to take before upgrading the server version

The following describes the steps you must take before upgrading the HADB server version.

Important

Before upgrading the HADB server version, do not change the values specified in the environment variables ADBLANG and ADBCLTLANG. If they are changed, you might not be able to correctly upgrade the HADB server version.

The language code and nation code specified in the LANG environment variable can be changed as follows:

- Change from ja_JP.UTF-8 to en_US.UTF-8
- Change from en_US.UTF-8 to ja_JP.UTF-8

The encoding set (type of character encoding) cannot be changed.

Note that if you change the value of the LANG environment variable, you also need to change the value of the LANG environment variable for the HADB client.

(1) Checking the size of the dictionary DB area and system-table DB area

Upgrading the HADB server version might increase the size of the dictionary DB area and system-table DB area.

Therefore, check the following two points for each DB area:

- Size of the DB area that will increase during version upgrade
- Amount of free space on the disk containing the DB area files of which the DB area consists

Compare the check results and make sure that an increase in the size of the DB area will not cause a problem. The following describes the check method for each value.

▪ Checking the size of the DB area that will increase during version upgrade

Re-estimate the size of the dictionary DB area as explained in 5.11 Estimating the size of the dictionary DB area.

Re-estimate the size of the system-table DB area as explained in 5.12 Estimating the size of the system-table DB area.

▪ **Checking the amount of free space on the disk containing the DB area files of which the DB area consists**

How to check the amount of free space on the disk containing the DB area files depends on the type of files that are used as DB area files.

For regular files

Execute the `df` command of the OS to check the amount of free space on the disk containing the DB area files. The following shows execution examples.

Execution example 1 (for dictionary DB area files)

```
df $DBDIR/ADBDIC
```

Execution example 2 (for system-table DB area files)

```
df $DBDIR/ADBSTBL
```

The amount of free space on the disk containing the DB area files specified in the `df` command is displayed in kilobytes.

For block special files

Execute the `adbdbstatus` command to output summary information for DB areas. For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*. The following shows execution examples.

Execution example 1 (for dictionary DB areas)

```
adbdbstatus -c dbarea -n ADBDIC -S K
```

Execution example 2 (for system-table DB areas)

```
adbdbstatus -c dbarea -n ADBSTBL -S K
```

Check the following two items output as the summary information for DB areas:

- `KB_Total_segments`
- `KB_Used_segments`

Based on the values you checked, use the following formula to determine the amount of free space on the disk.

Formula (kilobytes)

Free space on the disk containing the DB area files = KB_Total_segments - KB_Used_segments

If the check results indicate that the free disk space will be insufficient due to the increase in the DB area size, see 15.3.1 When a free space shortage is caused by an increase in the size of the DB area files and take the appropriate action.

(2) Checking the memory requirement

Upgrading the HADB server version might increase the HADB server's memory requirement. Re-estimate the memory requirements as explained in 6.3 Estimating the HADB server's memory requirement.

(3) Checking the environment variables

When upgrading the HADB server version, you might need to change the values assigned to environment variables. Check whether the environment variables need to be changed based on the explanation in 8.4 Setting environment variables.

(4) Checking the kernel parameters

When upgrading the HADB server version, you might need to change the kernel parameters. Check whether the kernel parameters need to be changed based on the explanation in 6.2 Estimating the kernel parameters.

(5) Checking the server definition

- If you are upgrading from HADB server version **02-00**, check the value specified for the `adb_log_usrfile_num` operand in the server definition. If `1` is specified for the `adb_log_usrfile_num` operand, change it to `2` or greater after the HADB server has terminated normally. Then, use the `adbstart` command to start the HADB server and use the `adbstop` command to terminate the HADB server normally.
- If you are upgrading the HADB server from a version earlier than **04-03**, make sure that the value specified for the `adb_sys_memory_limit` operand in the server definition satisfies the following formula. If you specify a value that does not satisfy the following formula, upgrading the HADB server will fail. In this case, the `KFAA40002-E` message is output.

```
value-specified-for-adb_sys_memory_limit-operand-in-server-definition
   ≤ kernel-parameter-vm.nr_hugepages × single-page-size-in-HugePages
```

(6) Checking the reserved words to be added

Upgrading the HADB server version might add reserved words.

The following table shows the reserved word that will be added.

Table 8-8: Reserved word to be added

HADB server version	Reserved word that will be added
02-01	`CURRENT_USER_IS_DBA`
03-00	`MILLISECOND`, `MILLISECONDS`, `NANOSECOND`, `NANOSECONDS`, `PICOSECOND`, `PICOSECONDS`

You cannot specify a reserved word as a table name or column name in SQL statements. If the same character string as the reserved word that will be added is used in an SQL statement, the SQL statement will cause an error after the HADB server version is upgraded.

Therefore, check whether a character string identical to the reserved word that will be added is specified in an SQL statement. If one is specified, enclose the character string that is the same as the reserved word in double quotation marks (`"`). For details, see *Reserved words* under *SQL Basics* in the manual *HADB SQL Reference*.

(7) Checking the number of schemas

If you try to upgrade the HADB server version when a database created by the HADB server version 02-00 contains no schema that was defined by an HADB user, upgrading will fail. Therefore, make sure that there is at least one schema defined by an HADB user.

If you try to upgrade the HADB server version when a database created by the HADB server version 02-00 contains 30,001 or more schemas that were defined by an HADB user, upgrading will fail. Therefore, use the `DROP SCHEMA` statement to delete unnecessary schemas so that the number of schemas does not exceed 30,000.

(8) Checking the schema name

When the HADB server is upgraded from version 02-00, an HADB user is created based on the schema name stored in the SQL_SCHEMATA dictionary table (base table). The authorization identifier and password of the created HADB user are the same as the schema name.

To connect to the upgraded HADB server, check the schema name that has been defined. A specification example follows of an SQL statement for checking the schema name.

Specification example

```
SELECT "SCHEMA_NAME" FROM "MASTER"."SQL_SCHEMATA"
```

After you finish upgrading the HADB server version, you must re-examine the password and privileges of the HADB user. For instructions, see (3) Re-examining the HADB user password and privileges (version 02-00) in 8.6.3 Steps to take after version upgrading.

(9) Stopping application programs and commands

Before you upgrade the HADB server version, terminate any application programs and commands that are being executed.

(10) Verifying that there are no non-updatable base tables

Before you upgrade your HADB server version, you need to confirm that there are no non-updatable base tables. If there are any non-updatable base tables, release them from non-updatable status before upgrading the HADB server version.

For details about how to check whether a base table is non-updatable, see (1) Checking whether a base table is non-updatable in 10.9.2 Checking the status and usage of a base table.

For details about how to release a base table from non-updatable status, see 15.8.1 Steps to take when a base table becomes non-updatable.

Important

If you upgrade the HADB server version without releasing base tables from non-updatable status, an error will occur when commands are re-executed after the upgrade. Because interrupted commands cannot be re-executed, the base table cannot be released from non-updatable status. At this time, the KFAA50243-E message is output. Take action as explained in the KFAA50243-E message.

Note that if an index is defined for a base table that entered non-updatable status with the following commands interrupted, and you take the action described in message KFAA50243-E, all indexes defined for the base table being released from non-updatable status are rebuilt. This means that depending on the amount of index data, removing the base table from non-updatable status might take a long time.

- adbimport command
- adbidxrebuild command

(11) Disabling the updated-row columnizing facility

If only system data will be backed up in (14) Acquiring a database backup, disable the updated-row columnizing facility now.

The following shows the procedure for disabling the updated-row columnizing facility.

1. Check the status of the updated-row columnizing facility.
   Execute the `adbcolumnize` command with the `-d` option specified.
   If `ACTIVE` (enabled) has been output in the `STATUS` column of the output result, go to step 2.
2. Disable the updated-row columnizing facility.
   Execute the `adbcolumnize` command with the `--stop` option specified.

For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).

> **Note**
>
> For details about the `adbcolumnize` command, see *adbcolumnize (Manage the Updated-Row Columnizing Facility)* in the manual *HADB Command Reference*.

(12) Terminating the HADB server normally

Before upgrading the HADB server version, use the `adbstop` command to terminate the HADB server normally.

If the HADB server has already stopped, use the following procedure to confirm that it terminated normally:

1. Execute the `adbls -d srv` command.
   Confirm that the HADB server status, which is output to the output item `STATUS`, is `STOP`.
   If it is `STOP`, proceed to step 2 or 3.
2. Check the server message log file.
   Open the server message log file (`$ADBDIR/spool/adbmessage`*XX*`.log`[#]) and confirm that the termination mode, output in the `KFAA91154-I` message, is `normally`.
   #
   XX is a sequential number between `01` and `04`.
3. Check syslog.
   Open syslog and confirm that the termination mode, output in the `KFAA91154-I` message, is `normally`.

If the result in step 1 is `STOP` and the result in step 2 or 3 is `normally`, the HADB server has terminated normally.

If the HADB server did not terminate normally, use the `adbstart` command to start it, and then use the `adbstop` command to terminate it normally.

Multi-node function
: If the HADB server in a multi-node configuration has not terminated normally, start the HADB servers on all nodes and then terminate normally the HADB servers on all nodes. In this case, there is no need to terminate HA Monitor (no need to execute the `monstop` command).

For details about how to start a HADB server in a multi-node configuration and how to terminate it normally, see 16.4.1 Starting the HADB servers in the multi-node configuration and 16.4.2 Terminating HADB servers in the multi-node configuration.

(13) Acquiring a server directory backup

Before upgrading the HADB server version, you also need to back up the server directory.

If you inadvertently upgrade the HADB server version after an abnormal termination, you will not be able to start the HADB server. Therefore, make a backup of the server directory.

Delete the backup that was made after you finish upgrading the HADB server version.

For details about the procedure for acquiring a backup, see (2) Backup procedure in 10.3.1 Backup acquisition method.

Multi-node function

If you are using the multi-node function, make a backup of the server directory on the HADB servers on all nodes.

(14) Acquiring a database backup

Before you upgrade the HADB server, make a full backup of the database. For details about the procedure for acquiring a full backup, see (2) Backup procedure in 10.3.1 Backup acquisition method.

Delete the backup that was made after you finish upgrading the HADB server version, if it is no longer needed.

Multi-node function

If you are using the multi-node function, make a backup as described in 16.8.1 Backup acquisition method.

In some cases, a backup of system data only rather than a full backup will suffice. A backup of system data only backs up the minimum files necessary, such as the files containing management information for the HADB server. It does not back up the data DB area files.

■ Cases in which a backup of system data only suffices

If you are restoring the HADB server to the previous version immediately after a version upgrade, a backup of system data only will suffice. For example, one such situation is when you intend to restore the HADB server to its previous version immediately after it was upgraded to a later version for the purpose of a version-upgrade test.

Note that if you perform an operation that updates the database after the upgrade, you will be unable to restore the HADB server to its previous version. For this reason, do not perform any operations other than the following after an upgrade:

- Execution of a `SELECT` statement, `COMMIT` statement, or `ROLLBACK` statement
- Execution of the `adbls` command

  Some commands other than the `adbls` command can also be executed. For details about the commands that can be executed, see the table *List of commands (database recovery by using a backup of system data only)* in *List of commands* in the manual *HADB Command Reference*.

If you perform an operation other than the preceding operations after an upgrade, database consistency will not be maintained if you use the backup of system data only to restore the HADB server to the previous version. This might prevent the HADB server from starting or cause the database to become corrupted.

To back up the system data only, back up the files numbered 1 to 6 in Table 10-6: List of files that needed to be backed up.

Multi-node function

To back up the system data only, back up the files numbered 1 to 6 in Table 16-9: List of files and directories that need to be backed up.

Note

- The backup procedure is the same for a full backup and a backup of system data only. The only difference is what is backed up.
- The procedure for restoring the database from a full backup and the procedure for restoring the database from a backup of system data only are the same.

8.6.2 Upgrading the server version

After you have taken the steps described in 8.6.1 Steps to take before upgrading the server version, upgrade the HADB server version using the following procedure.

Multi-node function

If you are using the multi-node function, perform steps *1. Log on to the OS from the HADB administrator's OS account* through *9. Confirm that the HADB server has been swapped* on the HADB servers on all nodes. After that, perform step *10. Upgrade the database version*.

Procedure

1. Log on to the OS from the HADB administrator's OS account.

   Log on to the server machine's OS from the HADB administrator's OS account that you have been using.

   Important

   Do not modify the HADB administrator's OS account. If you do, you might not be able to upgrade the HADB server version correctly.

2. Create the directory for storing the installation data.

   Create a directory to store the HADB server installation data. To do this, enter the following OS command:

   ```
   mkdir path-name-of-desired-directory
   ```

   An example follows in which `/home/adbmanager/vup` is selected as the directory path name.

   ▪ **Command execution example**

   ```
   mkdir /home/adbmanager/vup
   ```

3. Grant write privileges to the directory for storing the installation data.

   Grant write privileges to the directory for storing the installation data so that the HADB administrator can access the directory for write operations. To do this, enter the following OS command:

   ```
   chmod 755 path-name-of-desired-directory
   ```

   An example follows in which `/home/adbmanager/vup` is selected as the directory path name.

   ▪ **Command execution example**

   ```
   chmod 755 /home/adbmanager/vup
   ```

4. Mount the file system CD-ROM.

   Allow the file system CD-ROM containing the installation data (`tar.gz` file) for the HADB server to mount automatically.

   If the file system CD-ROM does not mount automatically, you must mount it manually. To do so, enter the following operating system command:

```
mount /dev/cdrom /media
```

   The underlined portion is the name of the mount directory for the file system CD-ROM. It varies depending on the environment.

> **Important**
>
> Depending on the machine being used, the directory and file names used on the CD-ROM might be different from those shown above. Enter the directory name that is displayed exactly as it is shown when you execute the `ls` operating system command.

5. Copy the installation command (`adbinstall` command).

   Copy the HADB server installation command (`adbinstall` command), which is stored in the mounted file system CD-ROM, to a directory of your choice.

   Enter the following operating system command to copy the `adbinstall` command to the desired directory:

```
cp /media/adbinstall path-name-of-desired-directory
```

   The underlined portion is the name of the mount directory for the file system CD-ROM. It varies depending on the environment.

   An example follows in which `/home/adbmanager/vup` is selected as the directory path name.

   • Command execution example

```
cp /media/adbinstall /home/adbmanager/vup
```

6. Copy the installation data (`tar.gz` file).

   Copy to a directory of your choice the HADB server installation data (`tar.gz` file) that is stored on the mounted file system CD-ROM.

   Enter the following operating system command to copy the `tar.gz` file to the directory of your choice:

```
cp /media/hitachi_advanced_data_binder_server-$VR.tar.gz
   path-name-of-directory-of-your-choice
```

   The underlined portion is the name of the mount directory for the file system CD-ROM. It varies depending on the environment. `$VR` indicates the HADB version and release number.

> **Important**
>
> Copy the `tar.gz` file to the directory in which the `adbinstall` command was stored in step *5. Copy the installation command (adbinstall command)*. You can upgrade the HADB server only if the `adbinstall` command and the `tar.gz` file are stored in the same directory.

   An example follows in which `/home/adbmanager/vup` is selected as the directory path name.

   • Command execution example

```
cp /media/hitachi_advanced_data_binder_server-$VR.tar.gz
   /home/adbmanager/vup
```

7. Assign execution privileges to the installation command (`adbinstall` command).

Assign execution privileges to the HADB server installation command (adbinstall command), which you copied to the directory of your choice, so that the HADB administrator has execution privileges.

An example follows in which the installation command (adbinstall command) is stored in /home/adbmanager/vup.

▪ Command execution example

```
chmod 777 /home/adbmanager/vup/adbinstall
```

Execution privileges have now been assigned to the installation command (adbinstall command).

8. Swap the HADB server.

   To swap the HADB server, execute the installation command (adbinstall command) that you copied to the directory of your choice.

   Execute the following HADB command to swap the HADB server:

```
/vup/adbinstall -s server-directory-path-name
```

   For the -s option, specify the absolute path for the server directory being used. If you specify for the -s option a directory that cannot be written by the HADB administrator, an error occurs and the KFAA91553-E message is output.

   The KFAA91553-E message is also output if the HADB administrator does not have write privileges for the installation command (adbinstall command) and the installation data (tar.gz file).

   If the KFAA91553-E message is output, grant write privileges for the target directory.

   Note

   The underlined portion is the path name of the directory that was selected to copy and store the adbinstall command.

   When you execute the adbinstall command, the KFAA91554-Q message is output, asking whether you want to overwrite the directory. Press **Y** to overwrite it.

   In the following execution example, /home/adbmanager/vup is selected as the directory path name and /HADB/server is selected as the server directory path name.

   ■ Command execution example

```
/home/adbmanager/vup/adbinstall -s /HADB/server
```

   If you execute the adbinstall command as root rather than using the HADB administrator's OS account, a warning message (KFAA91558-W message) is output. Normally, the HADB administrator's OS account is used to execute the adbinstall command. Therefore, if the KFAA91558-W message is output, make sure that executing the adbinstall command by root does not cause any problems.

   If a program exists, enter n or N in response to the input request in the KFAA91559-Q message that is output after the KFAA91558-W message. Then, execute the adbinstall command by using the HADB administrator's OS account.

   If you execute the adbinstall command as a superuser other than root, the KFAA91558-W message is not output.

   > **Note**
   >
   > root indicates a user for which 0 is displayed by executing the OS's id -u command. This includes a case where a value of 0 is displayed when the OS's id -u command is executed after another OS user was changed to root by using the OS's su command.

9. Confirm that the HADB server has been swapped.

Confirm that swapping of the HADB servers performed in step *8. Swap the HADB server* has been completed. Execute the following HADB command to check the HADB server version.

Multi-node function

If you are using the multi-node function, check the HADB server version information by executing the following command on all HADB servers on all nodes:

```
adbls -d ver
```

If no version information is displayed even when you execute the `adbls -d ver` command, there might be a specification error in one of the following environment variables:

- The server directory swapped in step *8. Swap the HADB server* has not been set in the `ADBDIR` environment variable.
- The `bin` directory under the server directory that was swapped in step *8. Swap the HADB server* has not been set in the `PATH` environment variable.

> **Note**
>
> For details about environment variables, see 8.4 Setting environment variables.

10. Upgrade the database version.

Execute the following HADB command to start the HADB server. Version upgrading starts.

```
adbstart
```

After execution of the `adbstart` command, the `KFAA91107-Q` message is output, asking whether you want to upgrade the database version. Press **Y** to upgrade it.

If the `adbstart` command terminates normally, upgrading of the HADB server version is complete.

Multi-node function

If you are using the multi-node function, start the HADB servers in the multi-node configuration as explained in 16.4.1 Starting the HADB servers in the multi-node configuration. If you attempt to start the HADB server on the master node, the `KFAA91107-Q` message asking whether the database version is to be upgraded is displayed. Press **Y** to upgrade the database version.

If the HADB servers on all nodes start successfully, the HADB server version has been upgraded.

> **Important**
>
> If the `adbstart` command does not terminate normally and the version upgrade of the HADB server fails , see 8.6.4 Steps to take when version upgrading fails.

8.6.3 Steps to take after version upgrading

After the HADB server version has been upgraded, take the following steps.

For details about how to restore the previous version after a version upgrade of the HADB server, see 8.7 Downgrading the HADB server version (restoring the previous version).

(1) Acquiring a database backup

We recommend that you make a full backup of the database whose version has been upgraded. Acquiring this backup will allow you, if necessary, to restore the database to its status immediately following completion of version upgrading.

(2) Upgrading the HADB client version

Once you have finished upgrading the HADB server version, upgrade the HADB client version as well. An HADB client can only connect to an HADB server if its version is the same as the HADB server version.

For details about upgrading the HADB client version, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

(3) Re-examining the HADB user password and privileges (version 02-00)

When the HADB server is upgraded from version 02-00, an HADB user is created based on the schema name stored in the `SQL_SCHEMATA` dictionary table (base table). The authorization identifier and password of all created HADB users are the same as the schema name. The `DBA` privilege, `CONNECT` privilege, and schema definition privilege are also granted to the created HADB users.

Therefore, first change the passwords of all HADB users. You can do so by using the `ALTER USER` definition SQL statement. For details about changing the password of an HADB user, see 11.6.2 Changing an HADB user's password.

Next, check the user privileges granted to all HADB users. If an HADB user has an unnecessary user privilege, use the `REVOKE` definition SQL statement to revoke the privilege. For details about revoking privileges, see 11.7.3 Revoking an HADB user's user privileges and schema operation privilege.

(4) Re-examining the HADB user's privileges (versions earlier than 03-00)

If you are upgrading the HADB server from a version earlier than 03-00, check whether the schema definition privilege granted to HADB users is necessary.

If there is an HADB user who does not need the schema definition privilege (for example, an HADB user who does not need to create a database on his or her own, but who only retrieves databases created by other HADB users), use the `REVOKE` definition SQL statement to revoke the privilege. For details about revoking privileges, see 11.7.3 Revoking an HADB user's user privileges and schema operation privilege.

(5) Re-examining the HADB users' privileges (version 03-02 or later)

If you are upgrading the HADB server to version **03-02** or later, revise the access privileges of the following HADB users.

▪ HADB users whose access privileges must be revised

- HADB users to whom access privileges are granted by the `GRANT` statement (a definition SQL statement) with `ALL PRIVILEGES` specified before the version upgrade

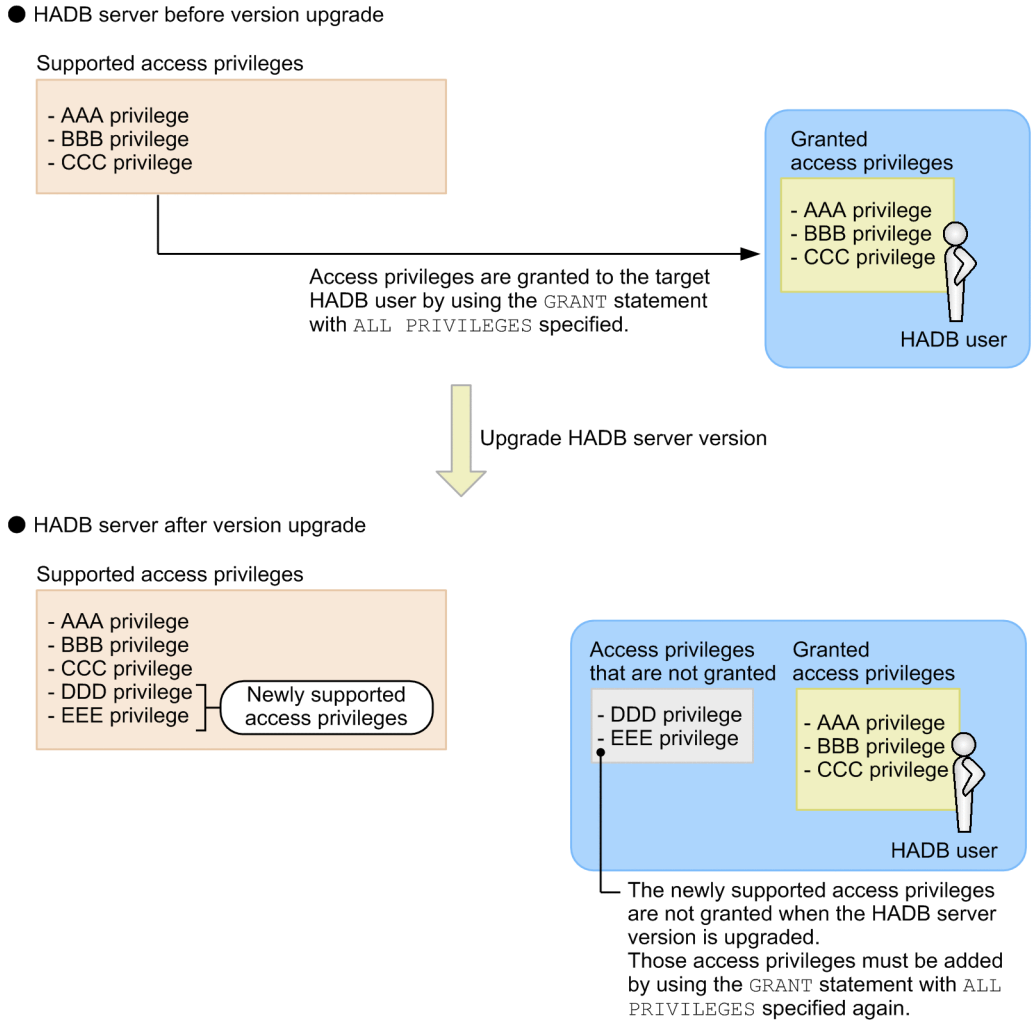
If new types of access privileges are supported in version **03-02** or later, upgrading the HADB server automatically grants the new types of access privileges to only the following HADB users:

- An HADB user who owns schema objects (excluding viewed tables)
- An HADB user who owns viewed tables (however, this applies only to the HADB user who owns all underlying tables of the viewed tables)

Therefore, when the HADB server is upgraded, new access privileges are not granted to the HADB users whose access privileges must be revised.

The following figure shows an overview of an HADB user whose access privileges must be revised.

Figure 8-2: Overview of an HADB user whose access privileges must be revised



If you want to grant all the access privileges, including the new ones, to an HADB user whose access privileges must be revised, re-execute the `GRANT` statement with `ALL PRIVILEGES` specified. For details about granting access privileges, see 11.8.1 Granting access privileges to an HADB user.

(6) Re-examining access privileges for viewed tables (version 03-06 or later)

The specifications of versions of HADB server earlier than **03-06** and versions **03-06** and later differ in the following ways:

- HADB server versions earlier than **03-06**

  Propagation of access privileges for viewed tables does not take place when new access privileges are granted for the underlying table.

  For example, if the `INSERT` privilege is granted for the underlying table `X.T1` of the viewed table `A.V1`, the `INSERT` privilege will not be granted for the viewed table `A.V1`. In this manner, because access privileges do not

propagate, the access privileges a user has for the underlying table do not necessarily match the access privileges the user has for the viewed tables.

- HADB server version **03-06** and later

  Propagation of access privileges for viewed tables takes place when new access privileges are granted for the underlying table.

  For example, if the `INSERT` privilege is granted for the underlying table `X.T1` of the viewed table `A.V1`, the `INSERT` privilege is also granted for the viewed table `A.V1`. Because the access privileges propagate, the access privileges the user has for the underlying table match the access privileges the user has for the viewed tables.

When you upgrade from a version earlier than **03-06** to version **03-06** or later, propagation of access privileges does not automatically take place for viewed tables that are already defined. Therefore, to have any changes you made to the access privileges for underlying tables also apply to the viewed tables, you need to grant the access privileges for the underlying table again. When you again grant access privileges for the underlying table to the owner of the viewed table, the access privileges propagate and the user will have the same access privileges for the viewed table as for the underlying table. It does not matter whether the owner of the viewed table already has the access privileges you are granting.

For details about the propagation of access privileges for viewed tables, see 2.7.6 Access privileges for viewed tables.

(7) Checking whether the uniqueness constraint has been violated

When the HADB server is upgraded from a version earlier than 02-02, it becomes unknown whether unique indexes satisfy the uniqueness constraint. Use one of the following methods to determine whether unique indexes satisfy the uniqueness constraint.

- **Checking using a command**

  1. Rebuild a unique index.

     Execute the `adbidxrebuild` command on the table for which the unique index is defined to rebuild a unique index. For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

  2. Check whether the uniqueness constraint has been violated.

     Use the `adbdbstatus` command to check whether the target unique index violates the uniqueness constraint. For details, see 10.9.3 Checking the status and usage of a B-tree index.

- **Checking using the `SELECT` statement**

  Execute the `SELECT` statement on the table for which the target unique index is defined to search for duplicated keys. For an example of specifying a `SELECT` statement that searches for duplicate keys, see (2) Steps to take when the uniqueness constraint is violated in 15.9.2 Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output).

If a unique index does not satisfy the uniqueness constraint (that is the index is in uniqueness constraint violation status), release the unique index from the uniqueness constraint violation status by following the steps in 15.9.2 Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output).

If it is not known whether a unique index satisfies the uniqueness constraint, the HADB server builds an access path, assuming that the unique index satisfies the uniqueness constraint.

(8) Rebuilding a text index for a word-context search

When both of the following conditions are met, you need to use the `adbidxrebuild` command to rebuild a *text index for a word-context search*.

- The HADB server was upgraded from version **03-06** to version **04-00** or later

- A text index for a word-context search was defined in version **03-06** of the HADB server (by specifying `TEXT WORDCONTEXT` in `INDEXTYPE` of a `CREATE INDEX` statement)

To check whether a text index for a word-context search was defined, see (34) Checking index options specified for text indexes in B.22 Searching a dictionary table.

If a text index for a word-context search was defined, use the `adbidxrebuild` command to rebuild it. For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

If you do not use the `adbidxrebuild` command to rebuild the text index for a word-context search, you will not be able to use that index. If you perform either of the following operations, an error occurs (the `KFAA34009-E` message is output).

- Searching using a text index for a word-context search
- Updating a text index for a word-context search

(9) Re-examining specification of the adbbuff operand in the server definition (version 04-01 or later)

From HADB server version **04-01**, a `-v` option was added to the `adbbuff` operand in the server definition as an option related to the use of table scan buffers. Therefore, you need to re-examine the specification of the `adbbuff` operand.

With the addition of the `-v` option, the options to be specified when using table scan buffers has changed as follows:

- HADB server version **04-00** and earlier
  The `-k` option is specified for the `adbbuff` operand.
- HADB server version **04-01** and later
  One or other of the `-v` option or the `-k` option is specified for the `adbbuff` operand.
  We recommend that you specify the `-v` option when you want to use a table scan buffer. Specifying the `-v` option allows table scan buffers to be used more efficiently than if the `-k` option is specified.

Therefore, if you specified the `-k` option in the `adbbuff` operand in version **04-00** or earlier, specify the `-v` option in version **04-01** and later. For details about the `-v` option of the `adbbuff` operand, see 7.2.11 Operands and options related to global buffers (command format).

When changing the specified option from the `-k` option to the `-v` option, check the following details before specifying the `-v` option:

■ Value specified for *memory-size-used-for-table-scan-buffer* in `-v` option

As the *memory-size-used-for-table-scan-buffer*, specify the value determined by the following formula:

```
Value specified for memory-size-used-for-table-scan-buffer =
    Value specified for -k option in version 04-00 or earlier × 4
```

When you specify the value determined by the preceding formula, the size of the table scan buffer will be the same as that allocated by the `-k` option in version **04-00** or earlier.

■ Value specified for *maximum-memory-size-used-per-real-thread* in `-v` option

In *maximum-memory-size-used-per-real-thread*, specify a value determined according to the explanation of the `-v` option of the `adbbuff` operand. For details about the `-v` option of the `adbbuff` operand, see 7.2.11 Operands and options related to global buffers (command format).

If you want memory to be evenly allocated among each real thread in the same manner as the -k option in version **04-00** and earlier, specify the value determined by the following formula:

```
Value specified for maximum-memory-size-used-per-real-thread =
    Value specified for -k option in version 04-00 or earlier × 4
    ÷ value specified for adb_sys_rthd_num operand
```

For details about the adb_sys_rthd_num operand in the server definition, see 7.2.2 Operands related to performance (set format).

Important

If there was insufficient table scan buffer during operation of HADB server version **04-00** or earlier, specify *maximum-memory-size-used-per-real-thread* in the -v option after upgrading to version **04-01** or later. If you do not specify this option, processing performance might suffer when executing multiple SQL statements concurrently.

(10) Re-examining the values specified in the server definition and command options (version 04-01 or later)

The adb_cmd_rthd_num operand, which specifies the number of processing real threads to be used for command execution, has been removed as a server definition operand in version **04-01** of HADB server. If you execute an adbstart command that still specifies the adb_cmd_rthd_num operand, an error will occur (the HADB server will not start). At this time, the KFAA52200-E message is output. If the adb_cmd_rthd_num operand was specified in the server definition, you must delete the adb_cmd_rthd_num operand before starting the HADB server.

With the removal of the adb_cmd_rthd_num operand, the default number of processing real threads used during command execution (the value used when the command option is omitted) is now the value specified in the adb_sql_exe_max_rthd_num operand in the server definition. As needed, you must re-examine the number of threads specified in the adb_sql_exe_max_rthd_num operand in the server definition and in various command options.

■ Target commands

- adbarchivechunk command
- adbexport command
- adbgetcst command
- adbidxrebuild command
- adbimport command
- adbmergechunk command
- adbunarchivechunk command

■ Re-examining the command options that specify numbers of threads

In HADB server version **04-01**, changes have been made to the command options that specify the processing real threads used by each command. An error occurs if a command is executed that specifies a command option from a previous version. At this time, the KFAA90500-E message is output. Therefore, you need to re-examine the command options for each command. The following table lists the affected commands and command options:

Table 8-9: Commands and command options to be re-examined

Command name	Name of command option in version 04-00 and earlier#	Name of command option in version 04-01 and later	Value to specify to realize the same number of threads as in version 04-00 and earlier
adbarchivechunk command	The following archive chunk option: • adb_arcv_scan_rthd_num	The following archive chunk option: • adb_arcv_rthd_num	For details, see (a) For the adbarchivechunk command (determining the value to specify for archive chunk option adb_arcv_rthd_num).
adbexport command	The following export option: • adb_export_scan_rthd_num	The following export option: • adb_export_rthd_num	For details, see (b) For the adbexport command (determining the value to specify for export option adb_export_rthd_num).
adbgetcst command	The following cost information collection option: • adb_getcst_scan_rthd_num	The following cost information collection option: • adb_getcst_rthd_num	For details, see (c) For the adbgetcst command (determining the value to specify for cost information collection option adb_getcst_rthd_num).
adbidxrebuild command	The following index rebuild options: • adb_idxrebuild_scan_rthd_num • adb_idxrebuild_sort_rthd_num • adb_idxrebuild_dividx_rthd_num	The following index rebuild option: • adb_idxrebuild_rthd_num	For details, see (d) For the adbidxrebuild command (determining the value of the index rebuild option adb_idxrebuild_rthd_num).
adbimport command	The following import options: • adb_import_dataload_rthd_num • adb_import_sort_rthd_num • adb_import_dividx_rthd_num	The following import option: • adb_import_rthd_num	For details, see (e) For the adbimport command (determining the value to specify for import option adb_import_rthd_num).
adbmergechunk command	The following merge chunk options: • adb_mergechunk_scan_rthd_num • adb_mergechunk_sort_rthd_num • adb_mergechunk_dividx_rthd_num	The following merge chunk option: • adb_mergechunk_rthd_num	For details, see (f) For the adbmergechunk command (determining merge chunk option adb_mergechunk_rthd_num).
adbunarchivechunk command	The following unarchive chunk options: • adb_unarcv_dataload_rthd_num • adb_unarcv_sort_rthd_num • adb_unarcv_dividx_rthd_num	The following unarchive chunk option: • adb_unarcv_rthd_num	For details, see (g) For the adbunarchivechunk command (determining the unarchive chunk option adb_unarcv_rthd_num).

#

No longer used in version **04-01**.

The value that must be specified to yield the same number of threads as in version **04-00** and earlier is shown for each command.

To determine the value of the variable *DEFAULT_RTHD_NUM* in a formula, use the formula in the following table:

Table 8-10: Determining variable DEFAULT_RTHD_NUM

Client-group facility		Value of variable *DEFAULT_RTHD_NUM*
When using the client-group facility	The associated command group is set in the `adbcltgrp` operand in the server definition	Whichever of the following values is smaller • Value specified in the `-r` option of the `adbcltgrp` operand in the server definition in which the command group is set • *A* - *B*
	The associated command group is not set in the `adbcltgrp` operand in the server definition	*A* - *B*
When not using the client-group facility		*A*

Legend:

A: The value specified for the `adb_sys_rthd_num` operand in the server definition

B: The total of the values specified in the `-e` option of the `adbcltgrp` operand in the server definition in which the client group is set

(a) For the adbarchivechunk command (determining the value to specify for archive chunk option adb_arcv_rthd_num)

By specifying the value determined by the formula in the following table in the `adb_arcv_rthd_num` archive chunk option, you can execute commands using the same number of threads as in version **04-00** or earlier.

Table 8-11: Determining value to specify for archive chunk option adb_arcv_rthd_num

No.	Value of archive chunk option or server definition in version 04-00 or earlier		Value to specify for archive chunk option adb_arcv_rthd_num in version 04-01 or later
1	When the archive chunk option `adb_arcv_scan_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Whichever of the following values is smaller • Value of variable *DEFAULT_RTHD_NUM*# • *Value specified for adb_sql_exe_max_rthd_num operand in server definition* × 2 + 1 If 0 is specified for the `adb_sql_exe_max_rthd_num` operand in the server definition, assume 1 for the calculation.
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	*Value specified for adb_cmd_rthd_num operand in server definition* × 2 + 1
4	When `0` is specified for the archive chunk option `adb_arcv_scan_rthd_num`		Whichever of the following values is smaller • Value of variable *DEFAULT_RTHD_NUM*# • *Value specified for adb_sql_exe_max_rthd_num operand in server definition* × 2 + 1 If 0 is specified for the `adb_sql_exe_max_rthd_num` operand in the server definition, assume 1 for the calculation.
5	When `1` or greater is specified for the archive chunk option `adb_arcv_scan_rthd_num`		*Value specified for archive chunk option adb_arcv_scan_rthd_num* × 2 + 1

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

(b) For the adbexport command (determining the value to specify for export option adb_export_rthd_num)

By specifying the value determined by the formula in the following table in the `adb_export_rthd_num` export option, you can execute commands using the same number of threads as in version **04-00** or earlier.

Table 8-12: Determining the value to specify for the adb_export_rthd_num export option

No.	Value of export option or server definition in version 04-00 or earlier		Value to specify for export option adb_export_rthd_num in version 04-01 or later
1	In either of the following cases: • When the export option `adb_export_scan_rthd_num` is not specified • When `0` is specified for the export option `adb_export_scan_rthd_num`	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Whichever of the following values is smaller • Value of variable *DEFAULT_RTHD_NUM*# • *Value specified for adb_sql_exe_max_rthd_num operand in server definition* × 2 + 1 If 0 is specified for the `adb_sql_exe_max_rthd_num` operand in the server definition, assume 1 for the calculation.
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	*Value specified for adb_cmd_rthd_num operand in server definition* × 2 + 1
4	When `1` or greater is specified for the export option `adb_export_scan_rthd_num`		*Value specified for export option adb_export_scan_rthd_num* × 2 + 1

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

(c) For the adbgetcst command (determining the value to specify for cost information collection option adb_getcst_rthd_num)

By specifying the value determined by the formula in the following table in the `adb_getcst_rthd_num` cost information collection option, you can execute commands using the same number of threads as in version **04-00** or earlier.

Table 8-13: Determining the value to specify for the adb_getcst_rthd_num cost-information collection option

No.	Value of cost-information collection option or server definition in version 04-00 or earlier		Value to specify for cost-information collection option adb_getcst_rthd_num in version 04-01 or later
1	When the cost-information collection option `adb_getcst_scan_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	*Value specified for adb_cmd_rthd_num operand in server definition* + 1
4	When `0` is specified for the cost-information collection option `adb_getcst_scan_rthd_num`		There is no specifiable value that obtains the same number of threads as in version **04-00** and earlier.

No.	Value of cost-information collection option or server definition in version 04-00 or earlier	Value to specify for cost-information collection option adb_getcst_rthd_num in version 04-01 or later
5	When 1 or greater is specified for the cost-information collection option adb_getcst_scan_rthd_num	*Value specified for cost information collection option adb_getcst_scan_rthd_num* + 1

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

(d) For the adbidxrebuild command (determining the value of the index rebuild option adb_idxrebuild_rthd_num)

By specifying the value determined by the following formula in the adb_idxrebuild_rthd_num index rebuild option, you can execute commands using the same number of threads as in version **04-00** or earlier.

Formula

```
value-specified-for-adb_idxrebuild_rthd_num
    = MAX(IDXREBUILD_SCAN_RTHD_NUM × 2,
          IDXREBUILD_SORT_RTHD_NUM,
          IDXREBUILD_DIVIDX_RTHD_NUM) + 1
```

Explanation of variables

IDXREBUILD_SCAN_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-14: Determining IDXREBUILD_SCAN_RTHD_NUM

No.	Value of index rebuild option or server definition in version 04-00 or earlier		Value to substitute for *IDXREBUILD_SCAN_RTHD_NUM*
1	When the index rebuild option adb_idxrebuild_scan_rthd_num is not specified	When the adb_cmd_rthd_num operand is not specified in the server definition	↓(*value of variable DEFAULT_RTHD_NUM*# - 1) ÷ 2↓
2		When 0 is specified for the adb_cmd_rthd_num operand in the server definition	
3		When 1 or greater is specified for the adb_cmd_rthd_num operand in the server definition	Value specified for the adb_cmd_rthd_num operand in the server definition
4	When 0 is specified for the index rebuild option adb_idxrebuild_scan_rthd_num		↓(*value of variable DEFAULT_RTHD_NUM*# - 1) ÷ 2↓
5	When 1 or greater is specified for the index rebuild option adb_idxrebuild_scan_rthd_num		Value specified for the index rebuild option adb_idxrebuild_scan_rthd_num

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

IDXREBUILD_SORT_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-15: Determining IDXREBUILD_SORT_RTHD_NUM

No.	Value of index rebuild option or server definition in version 04-00 or earlier		Value to substitute for *IDXREBUILD_SORT_RTHD_NUM*
1	When the index rebuild option `adb_idxrebuild_sort_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	Value specified for the `adb_cmd_rthd_num` operand in the server definition
4	When `0` is specified for the index rebuild option `adb_idxrebuild_sort_rthd_num`		Value of variable *DEFAULT_RTHD_NUM*#
5	When `1` or greater is specified for the index rebuild option `adb_idxrebuild_sort_rthd_num`		Value specified for the index rebuild option `adb_idxrebuild_sort_rthd_num`

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

IDXREBUILD_DIVIDX_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-16: Determining IDXREBUILD_DIVIDX_RTHD_NUM

No.	Value of index rebuild option or server definition in version 04-00 or earlier		Value to substitute for *IDXREBUILD_DIVIDX_RTHD_NUM*
1	When the index rebuild option `adb_idxrebuild_dividx_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	Value specified for the `adb_cmd_rthd_num` operand in the server definition
4	When `0` is specified for the index rebuild option `adb_idxrebuild_dividx_rthd_num`		Value of variable *DEFAULT_RTHD_NUM*#
5	When `1` or greater is specified for the index rebuild option `adb_idxrebuild_dividx_rthd_num`		Value specified for the index rebuild option `adb_idxrebuild_dividx_rthd_num`

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

(e) For the adbimport command (determining the value to specify for import option adb_import_rthd_num)

By specifying the value determined by the following formula in the `adb_import_rthd_num` import option, you can execute commands using the same number of threads as in version **04-00** or earlier.

Formula

```
value-specified-for-adb_import_rthd_num
    = MAX(IMPORT_DATALOAD_RTHD_NUM,
          IMPORT_SORT_RTHD_NUM,
          IMPORT_DIVIDX_RTHD_NUM) + 1
```

Explanation of variables

IMPORT_DATALOAD_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-17: Determining IMPORT_DATALOAD_RTHD_NUM

No.	Value of import option or server definition in version 04-00 or earlier		Value to substitute for *IMPORT_DATALOAD_RTHD_NUM*
1	When the import option `adb_import_dataload_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	*Value of variable DEFAULT_RTHD_NUM*# - 1
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	Value specified for the `adb_cmd_rthd_num` operand in the server definition
4	When `0` is specified for the import option `adb_import_dataload_rthd_num`		*Value of variable DEFAULT_RTHD_NUM*# - 1
5	When `1` or greater is specified for the import option `adb_import_dataload_rthd_num`		Value specified for the import option `adb_import_dataload_rthd_num`

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

IMPORT_SORT_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-18: Determining IMPORT_SORT_RTHD_NUM

No.	Value of import option or server definition in version 04-00 or earlier		Value to substitute for *IMPORT_SORT_RTHD_NUM*
1	When the import option `adb_import_sort_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	

No.	Value of import option or server definition in version 04-00 or earlier		Value to substitute for *IMPORT_SORT_RTHD_NUM*
3		When 1 or greater is specified for the adb_cmd_rthd_num operand in the server definition	Value specified for the adb_cmd_rthd_num operand in the server definition
4	When 0 is specified for the import option adb_import_sort_rthd_num		Value of variable *DEFAULT_RTHD_NUM*#
5	When 1 or greater is specified for the import option adb_import_sort_rthd_num		Value specified for the import option adb_import_sort_rthd_num

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

IMPORT_DIVIDX_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-19: Determining IMPORT_DIVIDX_RTHD_NUM

No.	Value of import option or server definition in version 04-00 or earlier		Value to substitute for *IMPORT_DIVIDX_RTHD_NUM*
1	When the import option adb_import_dividx_rthd_num is not specified	When the adb_cmd_rthd_num operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When 0 is specified for the adb_cmd_rthd_num operand in the server definition	
3		When 1 or greater is specified for the adb_cmd_rthd_num operand in the server definition	Value specified for the adb_cmd_rthd_num operand in the server definition
4	When 0 is specified for the import option adb_import_dividx_rthd_num		Value of variable *DEFAULT_RTHD_NUM*#
5	When 1 or greater is specified for the import option adb_import_dividx_rthd_num		Value specified for the import option adb_import_dividx_rthd_num

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

(f) For the adbmergechunk command (determining merge chunk option adb_mergechunk_rthd_num)

By specifying the value determined by the following formula in the adb_mergechunk_rthd_num merge chunk option, you can execute commands using the same number of threads as in version **04-00** or earlier.

Formula

```
value-specified-for-adb_mergechunk_rthd_num
    = MAX(MERGECHUNK_SCAN_RTHD_NUM × 2,
          MERGECHUNK_SORT_RTHD_NUM,
          MERGECHUNK_DIVIDX_RTHD_NUM) + 1
```

Explanation of variables

MERGECHUNK_SCAN_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-20: Determining MERGECHUNK_SCAN_RTHD_NUM

No.	Value of merge chunk option or server definition in version 04-00 or earlier		Value to substitute for *MERGECHUNK_SCAN_RTHD_NUM*
1	When the merge chunk option `adb_mergechunk_scan_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Whichever of the following values is larger: • ↓(*value specified for adb_sql_exe_max_rthd_num operand in server definition* - 1) ÷ 2↓ • 1
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	Value specified for the `adb_cmd_rthd_num` operand in the server definition
4	When `0` is specified for the merge chunk option `adb_mergechunk_scan_rthd_num`		Whichever of the following values is larger: • ↓(*value specified for adb_sql_exe_max_rthd_num operand in server definition* - 1) ÷ 2↓ • 1
5	When `1` or greater is specified for the merge chunk option `adb_mergechunk_scan_rthd_num`		Value specified for the merge chunk option `adb_mergechunk_scan_rthd_num`

MERGECHUNK_SORT_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-21: Determining MERGECHUNK_SORT_RTHD_NUM

No.	Value of merge chunk option or server definition in version 04-00 or earlier		*Value to substitute for MERGECHUNK_SORT_RTHD_NUM*
1	When the merge chunk option `adb_mergechunk_sort_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	Value specified for the `adb_cmd_rthd_num` operand in the server definition
4	When `0` is specified for the merge chunk option `adb_mergechunk_sort_rthd_num`		Value of variable *DEFAULT_RTHD_NUM*#
5	When `1` or greater is specified for the merge chunk option `adb_mergechunk_sort_rthd_num`		Value specified for the `adb_mergechunk_sort_rthd_num` merge chunk option

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

MERGECHUNK_DIVIDX_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-22: Determining MERGECHUNK_DIVIDX_RTHD_NUM

No.	Value of merge chunk option or server definition in version 04-00 or earlier		Value to substitute for *MERGECHUNK_DIVIDX_RTHD_NUM*
1	When the merge chunk option `adb_mergechunk_dividx_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	Value specified for the `adb_cmd_rthd_num` operand in the server definition
4	When `0` is specified for the merge chunk option `adb_mergechunk_dividx_rthd_num`		Value of variable *DEFAULT_RTHD_NUM*#
5	When `1` or greater is specified for the merge chunk option `adb_mergechunk_dividx_rthd_num`		Value specified for the merge chunk option `adb_mergechunk_dividx_rthd_num`

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

(g) For the adbunarchivechunk command (determining the unarchive chunk option adb_unarcv_rthd_num)

By specifying the value determined by the following formula in the `adb_unarcv_rthd_num` unarchive chunk option, you can execute commands using the same number of threads as in version **04-00** or earlier.

Formula

```
value-specified-for-adb_unarcv_rthd_num
    = MAX(UNARCHIVE_DATALOAD_RTHD_NUM,
          UNARCHIVE_SORT_RTHD_NUM,
          UNARCHIVE_DIVIDX_RTHD_NUM) + 1
```

Explanation of variables

UNARCHIVE_DATALOAD_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-23: Determining UNARCHIVE_DATALOAD_RTHD_NUM

No.	Value of unarchive chunk option or server definition in version 04-00 or earlier		Value to substitute for *UNARCHIVE_DATALOAD_RTHD_NUM*
1	When the unarchive chunk option `adb_unarcv_dataload_rthd_num` is not specified	When the `adb_cmd_rthd_num` operand is not specified in the server definition	*Value of variable DEFAULT_RTHD_NUM*# - 1

No.	Value of unarchive chunk option or server definition in version 04-00 or earlier		Value to substitute for *UNARCHIVE_DATALOAD_RTHD_NUM*
2		When 0 is specified for the adb_cmd_rthd_num operand in the server definition	
3		When 1 or greater is specified for the adb_cmd_rthd_num operand in the server definition	Value specified for the adb_cmd_rthd_num operand in the server definition
4	When 0 is specified for the unarchive chunk option adb_unarcv_dataload_rthd_num		*Value of variable DEFAULT_RTHD_NUM*# - 1
5	When 1 or greater is specified for the unarchive chunk option adb_unarcv_dataload_rthd_num		Value specified for the unarchive chunk option adb_unarcv_dataload_rthd_num

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

UNARCHIVE_SORT_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-24: Determining UNARCHIVE_SORT_RTHD_NUM

No.	Value of unarchive chunk option or server definition in version 04-00 or earlier		Value to substitute for *UNARCHIVE_SORT_RTHD_NUM*
1	When the unarchive chunk option adb_unarcv_sort_rthd_num is not specified	When the adb_cmd_rthd_num operand is not specified in the server definition	Value of variable *DEFAULT_RTHD_NUM*#
2		When 0 is specified for the adb_cmd_rthd_num operand in the server definition	
3		When 1 or greater is specified for the adb_cmd_rthd_num operand in the server definition	Value specified for the adb_cmd_rthd_num operand in the server definition
4	When 0 is specified for the unarchive chunk option adb_unarcv_sort_rthd_num		Value of variable *DEFAULT_RTHD_NUM*#
5	When 1 or greater is specified for the unarchive chunk option adb_unarcv_sort_rthd_num		Value specified for the unarchive chunk option adb_unarcv_sort_rthd_num

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

UNARCHIVE_DIVIDX_RTHD_NUM

Use the applicable formula in the following table to determine the value:

Table 8-25: Determining UNARCHIVE_DIVIDX_RTHD_NUM

No.	Value of unarchive chunk option or server definition in version 04-00 or earlier		Value to substitute for *UNARCHIVE_DIVIDX_RTHD_NUM*
1	When the unarchive chunk option	When the adb_cmd_rthd_num	Value of variable *DEFAULT_RTHD_NUM*#

No.	Value of unarchive chunk option or server definition in version 04-00 or earlier		Value to substitute for *UNARCHIVE_DIVIDX_RTHD_NUM*
	`adb_unarcv_dividx_rthd_num` is not specified	operand is not specified in the server definition	
2		When `0` is specified for the `adb_cmd_rthd_num` operand in the server definition	
3		When `1` or greater is specified for the `adb_cmd_rthd_num` operand in the server definition	Value specified for the `adb_cmd_rthd_num` operand in the server definition
4	When `0` is specified for the unarchive chunk option `adb_unarcv_dividx_rthd_num`		Value of variable *DEFAULT_RTHD_NUM*#
5	When `1` or greater is specified for the unarchive chunk option `adb_unarcv_dividx_rthd_num`		Value specified for the unarchive chunk option `adb_unarcv_dividx_rthd_num`

#

Determine the value of the variable *DEFAULT_RTHD_NUM* as explained in Table 8-10: Determining variable DEFAULT_RTHD_NUM.

(11) Enabling the updated-row columnizing facility

Enable the updated-row columnizing facility if you disabled it in (11) Disabling the updated-row columnizing facility in 8.6.1 Steps to take before upgrading the server version.

Execute the `adbcolumnize` command with the `--start` option specified. The updated-row columnizing facility is enabled.

Note

For details about the `adbcolumnize` command, see *adbcolumnize (Manage the Updated-Row Columnizing Facility)* in the manual *HADB Command Reference*.

8.6.4 Steps to take when version upgrading fails

This section explains the steps to take when upgrading of the HADB server version fails.

Steps to take

1. Check the messages.

   Use a text editor to open the server message log file and check the error messages that have been output. The server message log file is `/spool/adbmessage`*XX*`.log` in the server directory. *XX* in the file name is a sequential number between `01` and `04`. Open the file with the latest modification date and time.

   If none of the messages listed below have been output, proceed to step 2.

   If any of the following messages have been output, take the actions indicated in the referenced subsections of this chapter:

   - `KFAA30811-E` message

     See (1) Steps to take when the KFAA30811-E message is output.

   - `KFAA41210-E` message

See (2) Steps to take when the KFAA41210-E message is output.

- `KFAA50038-E` message

  See (3) Steps to take when the KFAA50038-E message is output.

- `KFAA51400-E` message

  See (4) Steps to take when the KFAA51400-E message is output.

2. Take action according to the Action section of the relevant message.

   Based on the ID of the output error message, see the relevant page in the manual *HADB Messages*. Take the step described in the Action section for that message.

   If the Action section says, Contact the customer support center, proceed to step 3.

3. Execute the `adbinfoget` command.

   If the Action section says, Contact the customer support center, you need to acquire troubleshooting information to investigate the cause of the error.

   Execute the `adbinfoget` command as explained in 14.2 Collecting troubleshooting information (adbinfoget command). When you do so, specify the `-f` option for the command to acquire the error information (core files) as well. For details about the `-f` option, see *adbinfoget (Collect Troubleshooting Information)* in the manual *HADB Command Reference*.

   When you contact the customer support center, send along the troubleshooting information you have acquired.

After you have taken whichever action was indicated as the response to the message that was output, restart the HADB server. If an attempt to restart the HADB server results in an error and version upgrading fails, you will have to recover the database from the backup. See (5) Recovering the database from a backup.

(1) Steps to take when the KFAA30811-E message is output

In this case, version upgrading failed because the number of schemas defined exceeds 30,000. You must first restore the HADB server to the previous version, and then take the steps necessary to recover from the error.

To restore the HADB server to the previous version and recover from the error:

Procedure

1. Delete the server directory.

   Delete the server directory by following the steps in (1) Deleting the server directory under 8.12.1 Uninstallation procedure.

2. Recover the previous version of the server directory from a backup.

   From a backup acquired earlier, copy the previous version of the server directory and return it to the storage location from which it was deleted in step 1.

3. Start the HADB server.

   Use the `adbstart` command to start the HADB server.

4. Reduce the number of schemas to 30,000 or fewer.

   Use the `DROP SCHEMA` statement to deleted unnecessary schemas so that the number of defined schemas is 30,000 or fewer. Version upgrading cannot be executed if the number of schemas exceeds 30,000.

5. Terminate the HADB server.

   Use the `adbstop` command to terminate the HADB server.

6. Confirm that the HADB server terminated normally.

   Check whether the HADB server terminated normally by following the steps in (12) Terminating the HADB server normally under 8.6.1 Steps to take before upgrading the server version.

After you have confirmed that the HADB server terminated normally, perform the version upgrade again by following the steps in 8.6.2 Upgrading the server version.

(2) Steps to take when the KFAA41210-E message is output

In this case, version upgrading failed because the number of user log files was insufficient. You must first restore the HADB server to the previous version, and then take the steps necessary to recover from the error.

To restore the HADB server to the previous version and recover from the error:

Procedure

1. Delete the server directory.
   Delete the server directory by following the steps in (1) Deleting the server directory under 8.12.1 Uninstallation procedure.
2. Recover the previous version of the server directory from a backup.
   From a backup acquired earlier, copy the previous version of the server directory and return it to the storage location from which it was deleted in step 1.
3. Check the value specified for the `adb_log_usrfile_num` operand in the server definition.
   Check whether `1` is specified for the `adb_log_usrfile_num` operand in the server definition. If `1` is specified, change it to at least `2`.
4. Start and terminate the HADB server.
   Use the `adbstart` command to start the HADB server. Then, use the `adbstop` command to terminate it.
5. Confirm that the HADB server terminated normally.
   Check whether the HADB server terminated normally by following the steps in (12) Terminating the HADB server normally under 8.6.1 Steps to take before upgrading the server version.

After you have confirmed that the HADB server terminated normally, perform the version upgrade again by following the steps in 8.6.2 Upgrading the server version.

(3) Steps to take when the KFAA50038-E message is output

The steps to take depend on which of the following was output to variable *aa....aa* in the `KFAA50038-E` message:

- Variable *aa....aa* is `of an abnormal termination the last time it ended`
- Variable *aa....aa* is `the adbstart command was never been executed in previous versions`
- Variable *aa....aa* is `of using the multiple node facility`

(a) If variable aa....aa is "of an abnormal termination the last time it ended"

Version upgrading failed because the HADB server terminated abnormally. You must first restore the HADB server to the previous version, and then take the steps necessary to recover from the error.

Multi-node function
If you are using the multi-node function, restore the previous version of the HADB servers on all nodes.

To restore the HADB server to the previous version and recover from the error:

Procedure

1. Delete the server directory.

   Delete the server directory by following the steps in (1) Deleting the server directory under 8.12.1 Uninstallation procedure.

   Multi-node function

   If you are using the multi-node function, perform this step on all nodes.

2. Recover the previous version of the server directory from a backup.

   From a backup acquired earlier, copy the previous version of the server directory and return it to the storage location from which it was deleted in step 1.

   Multi-node function

   If you are using the multi-node function, perform this step on all nodes.

3. Start and terminate the HADB server.

   Use the `adbstart` command to start the HADB server. Then, use the `adbstop` command to terminate it.

   Multi-node function

   If you are using the multi-node function, start the HADB servers in the multi-node configuration as explained in 16.4.1 Starting the HADB servers in the multi-node configuration. Then, terminate the HADB servers in the multi-node configuration as explained in 16.4.2 Terminating HADB servers in the multi-node configuration.

4. Confirm that the HADB server terminated normally.

   Check whether the HADB server terminated normally by following the steps in (12) Terminating the HADB server normally under 8.6.1 Steps to take before upgrading the server version.

After you have confirmed that the HADB server terminated normally, perform the version upgrade again by following the steps in 8.6.2 Upgrading the server version.

(b) If variable aa....aa is "the adbstart command was never been executed in previous versions"

Version upgrading failed because the previous version of the HADB server was never started.

You need to execute the `adbinit` command to initialize the database. For details about how to initialize the database, see 9.2 Initializing a database (creating data DB areas).

(c) If variable aa....aa is "of using the multiple node facility"

Version upgrading failed because an attempt was made to perform version upgrading together with an attempt to configure the system to use the multi-node function on an HADB server whose version is earlier than 03-00.

Version upgrade processing will fail if you specify operands of the multi-node function in the server definition before you upgrade the version. After you have upgraded your HADB server version, change the settings so that the system can use the multi-node function.

Procedure

1. Delete the `adb_sys_multi_node_info` operand from the server definition.

   Delete the `adb_sys_multi_node_info` operand from the server definition so that the multi-node function is not used.

   Alternatively, comment out the `adb_sys_multi_node_info` operand in the server definition. Specify a hash mark (#) at the beginning of the line containing the operand to treat that line as a comment.

2. Start the HADB server.

Start the HADB server by executing the `adbstart` command while the multi-node function is not used. Version upgrading starts.

After the `adbstart` command has executed, the `KFAA91107-Q` message is output, asking whether you want to upgrade the database version. Press **Y** to upgrade.

If the `adbstart` command terminates normally, upgrading of the HADB server version has been completed.

3. Terminate the HADB server normally.

   After the `adbstart` command has terminated normally and the upgrade processing has been completed, terminate the HADB server normally. To do this, execute the `adbstop` command.

4. Specify the `adb_sys_multi_node_info` operand in the server definition.

   Once the upgrade processing is completed, you can use the multi-node function. Specify the `adb_sys_multi_node_info` operand in the server definition that was deleted in step 1.

5. Start the HADB server in the multi-node configuration.

   After you have specified the `adb_sys_multi_node_info` operand in the server definition, start the HADB server in the multi-node configuration as explained in 16.4.1 Starting the HADB servers in the multi-node configuration.

(4) Steps to take when the KFAA51400-E message is output

Version upgrading failed because the database created by the HADB server version 02-00 did not contain any schema.

You need to execute the `adbinit` command to initialize the database. For details about how to initialize the database, see 9.2 Initializing a database (creating data DB areas).

(5) Recovering the database from a backup

If an attempt to restart the HADB server results in an error after upgrading of the HADB server has failed and you have taken whichever action was indicated as the response to the message that was output, you will have to recover the database from a backup.

The procedure below explains how to recover the database from a backup and re-configure the previous version of HADB server.

Multi-node function

If you are using the multi-node function, perform this step on all HADB servers on all nodes.

Procedure

1. Delete the server directory.

   Delete the server directory by following the steps in (1) Deleting the server directory under 8.12.1 Uninstallation procedure.

   Multi-node function

   If you are using the multi-node function, perform this step on all nodes.

2. Recover the previous version of server directory from a backup.

   From a backup acquired earlier, copy the previous version of the server directory and return it to the storage location from which it was deleted in step 1.

   Multi-node function

   If you are using the multi-node function, perform this step on all nodes.

3. Recover the DB directory from a backup.

Recover the DB directory from the backup by following the procedure in 10.3.2 Recovering the database from the backup.

Multi-node function

If you are using the multi-node function, recover the DB directory from the backup as described in 16.8.2 Recovering a database from a backup.

4. Check the value specified for the `adb_db_path` operand in the server definition.

   Check whether the path name of the DB directory recovered in step 3 is defined in the `adb_db_path` operand in the server definition.

5. Start and terminate the HADB server.

   Use the `adbstart` command to start the HADB server. Then, use the `adbstop` command to terminate it.

   Multi-node function

   If you are using the multi-node function, start the HADB servers in the multi-node configuration as explained in 16.4.1 Starting the HADB servers in the multi-node configuration. Then, terminate the HADB servers in the multi-node configuration as explained in 16.4.2 Terminating HADB servers in the multi-node configuration.

6. Confirm that the HADB server terminated normally.

   Check whether the HADB server terminated normally by following the steps in (12) Terminating the HADB server normally under 8.6.1 Steps to take before upgrading the server version.

If the HADB server terminated normally, you can rebuild the previous version of the HADB server. If you need to upgrade the HADB server version, see 8.6.2 Upgrading the server version.

8.6.5 Notes on version upgrading

This section provides cautionary notes that apply when upgrading the HADB server version.

(1) Collecting cost information again

The cost information that was collected by using an HADB server whose version is earlier than **04-03** cannot be used with an HADB server whose version is **04-03** or later. Therefore, if you upgrade the version of the HADB server from a version earlier than **04-03** to version **04-03** or later, use the `adbgetcst` command to collect the cost information again.

(2) Results of value expressions and set operations

When an HADB server is upgraded from a version earlier than 03-00, the data type and data length of the results of the value expressions and set options described below, as well as the result values, change after the version upgrade. The following two tables show the items whose results change after a version upgrade.

Table 8-26: Items for which the data type and data length of the results change

No.	Items for which the data type and data length of the results change		Explanation
1	Set functions	`PERCENTILE_CONT`	The data type of the result changes from aggregated argument to `DOUBLE PRECISION`.
2		`MEDIAN`	
3	0-byte literal		Changes from `CHAR(0)` to `VARCHAR(1)` data with an actual length of 0.

No.	Items for which the data type and data length of the results change		Explanation
4	CASE expression		If the data type of the result is character string data, the data type of the result changes to VARCHAR.
5	Scalar functions	COALESCE	If the data type of the result is character string data, the data type of the result changes to VARCHAR.
6		DECODE	
7		GREATEST	
8		LEAST	
9		NULLIF	
10		CEIL	If the target data is DECIMAL, the data type of the result changes from DECIMAL (precision of target data, scaling of target data) to DECIMAL (precision of target data, 0).
11		FLOOR	
12		ROUND	If the target data is DECIMAL (excludes a case in which precision is 38), the data type of the result changes from DECIMAL (precision of target data, scaling of target data) to DECIMAL (precision of target data + 1, scaling of target data).
13	Set operations	UNION	The data type of any column that is derived as a result of a set operation changes. If the data types of the columns corresponding to the columns derived in a result of each query primary are all CHAR, and their maximum lengths are equal, the data type of the result becomes CHAR. In all other cases, the data type of the result becomes VARCHAR.
14		UNION ALL	

Table 8-27: Items whose result values change

No.	Items whose result value changes		Explanation
1	Scalar functions	CAST	If the post-conversion length is longer than the data length of the result, no error occurs and the decimal portion is discarded (rounded to the nearest even number).
2		CONVERT (no number format specified, conversion from the DOUBLE PRECISION data type to character string data)	
3		SUBSTR	For the following items, data with an actual length of 0 bytes is returned instead of the null value: • When the length of the resulting character string is 0 • When data with an actual length of 0 bytes (character) is specified as the extraction source • When the number of extracted characters is 0 • When the absolute value of the start position is greater than the number of characters at the extraction source For the following item, no error occurs and 1 is treated as the start position: • When 0 is specified as the start position For the following item, data that can be extracted is returned instead of the null value: • When a value exceeding 32,000 is specified as the number of characters to be extracted
4		LEFT	
5		RIGHT	
6		TRIM	For the following items, data with an actual length of 0 bytes is returned instead of the null value: • When data with an actual length of 0 bytes (character) is specified as the target data
7		LTRIM	
8		RTRIM	

No.	Items whose result value changes		Explanation
			• When the character strings specified for the target data are all removed

(3) Retrieval results of viewed tables defined in an earlier version

If any of the following items are present in the viewed tables defined in an HADB server whose version is earlier than **03-00**, the data type and data length of the result is determined according to the description in (2) Results of value expressions and set operations.

After that, the scalar function CONVERT is executed to match the data type and data length to those determined in the HADB server whose version is earlier than 03-00.

Targeted items

- Set functions
  - PERCENTILE_CONT
  - MEDIAN
- 0-byte literal
- CASE expression
- Scalar functions
  - COALESCE
  - CEIL
  - DECODE
  - FLOOR
  - GREATEST
  - LEAST
  - NULLIF
  - ROUND (mathematical function)

(4) Changing to a system that uses the multi-node function

During a version upgrade, even if you want to change the system to one that uses the multi-node function, you cannot perform the upgrade and change the system to one that uses the multi-node function at the same time. If you specify the operand for the multi-node function in the server definition before a version upgrade, the upgrade fails (the KFAA50038-E message is output). Therefore, first perform the version upgrade, and then change the system to one that uses the multi-node function.

For details about the action to take when the KFAA50038-E message is output, see (c) If variable aa....aa is "of using the multiple node facility" in (3) Steps to take when the KFAA50038-E message is output under 8.6.4 Steps to take when version upgrading fails.

(5) Re-creation of viewed tables in the event of a version upgrade

When you upgrade the HADB server to version **05-01**, viewed tables that satisfy all of the following conditions are re-created as part of the upgrade process. Viewed tables whose re-creation failed are invalidated.

■ **Conditions for re-creation of viewed tables as part of a version upgrade**

Viewed tables that satisfy all of the following conditions will be re-created:

- **Viewed tables that depend on a dictionary table or system table that is modified by the version upgrade process**

  For details about how to find out which viewed tables depend on a dictionary table or system table, see (33) Finding a list of dependent viewed tables in B.22 Searching a dictionary table.

  For details about which dictionary tables and system tables are modified as part of the version upgrade process, see Table 8-28: Dictionary tables and system tables modified during version upgrade (1) and Table 8-29: Dictionary tables and system tables modified during version upgrade (2).

- **Viewed tables that are valid at the time of the version upgrade (viewed tables that have not been invalidated)**

  For details about how to find out whether a viewed table is valid, see (22) When checking whether a viewed table has been invalidated under B.22 Searching a dictionary table.

If viewed tables are successfully re-created during the version upgrade process, the message `KFAA51311-I` is output.

If re-creating the viewed tables fails during the version upgrade process, the message `KFAA51312-W` is output. Viewed tables whose re-creation failed are invalidated. If this occurs, the version upgrade processing continues.

If the message `KFAA51312-W` is output, wait until the version upgrade is complete and then check the message that was output immediately before `KFAA51312-W`. Based on the contents of the message, resolve the issue that caused re-creation of the viewed table to fail. Then, re-create the viewed table. For details, see (6) When rebuilding of viewed tables at version upgrade fails under 11.2.8 Releasing a viewed table from invalidation.

The following two tables list the dictionary tables and system tables that are modified when you upgrade the HADB server to version **05-01**.

Table 8-28: Dictionary tables and system tables modified during version upgrade (1)

No.	Dictionary table or system table name	HADB server from which version upgrade is being performed									
		02-00	02-01	02-02	03-00	03-01	03-02	03-03	03-04	03-05	03-06
1	SQL_TABLES	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
2	SQL_COLUMNS	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
3	SQL_INDEXES	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
4	SQL_DIV_TABLE	--	Y	Y	--	--	--	--	--	--	--
5	SQL_DIV_INDEX	--	Y	Y	--	--	--	--	--	--	--
6	SQL_DBAREAS	--	--	--	--	--	--	--	--	--	--
7	SQL_SCHEMATA	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
8	SQL_VIEWS	--	Y	Y	Y	Y	Y	Y	Y	Y	--
9	SQL_VIEW_TABLE_USAGE	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
10	SQL_DEFINE_SOURCE	--	Y	Y	Y	Y	Y	Y	Y	Y	--

No.	Dictionary table or system table name	HADB server from which version upgrade is being performed									
		02-00	02-01	02-02	03-00	03-01	03-02	03-03	03-04	03-05	03-06
11	SQL_DEFINE_ENVIRONMENT	--	Y	Y	Y	Y	Y	Y	Y	Y	--
12	SQL_USERS	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
13	SQL_TABLE_CONSTRAINTS	--	--	Y	--	--	--	--	--	--	--
14	SQL_INDEX_COLINF	--	--	Y	--	--	--	--	--	--	--
15	SQL_KEY_COLUMN_USAGE	--	--	--	--	--	--	--	--	--	--
16	SQL_REFERENTIAL_CONSTRAINTS	--	--	--	--	--	--	--	--	--	--
17	SQL_TABLE_PRIVILEGES	--	--	--	Y	Y	Y	Y	Y	Y	--
18	SQL_AUDITS	--	--	--	--	--	--	--	--	--	--
19	STATUS_TABLES	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
20	STATUS_INDEXES	--	Y	Y	--	--	--	--	--	--	--
21	STATUS_CHUNKS	--	Y	Y	Y	Y	Y	Y	Y	Y	Y
22	STATUS_SYNONYM_DICTIONARIES	--	--	--	--	--	--	--	--	--	--
23	STATUS_COLUMNS	--	--	--	--	--	--	--	Y	Y	Y

Legend:

Y: The dictionary table or system table is modified as part of the version upgrade process.

--: The dictionary table or system table is not modified as part of the version upgrade process.

Table 8-29: Dictionary tables and system tables modified during version upgrade (2)

No.	Dictionary table or system table name	HADB server from which version upgrade is being performed			
		04-00	04-01	04-02	04-03 or 05-00
1	SQL_TABLES	Y	--	--	--
2	SQL_COLUMNS	Y	--	--	--
3	SQL_INDEXES	--	--	--	--
4	SQL_DIV_TABLE	--	--	--	--
5	SQL_DIV_INDEX	--	--	--	--
6	SQL_DBAREAS	--	--	--	--

No.	Dictionary table or system table name	HADB server from which version upgrade is being performed			
		04-00	04-01	04-02	04-03 or 05-00
7	SQL_SCHEMATA	--	--	--	--
8	SQL_VIEWS	--	--	--	--
9	SQL_VIEW_TABLE_USAGE	--	--	--	--
10	SQL_DEFINE_SOURCE	--	--	--	--
11	SQL_DEFINE_ENVIRONMENT	--	--	--	--
12	SQL_USERS	--	--	--	--
13	SQL_TABLE_CONSTRAINTS	--	--	--	--
14	SQL_INDEX_COLINF	--	--	--	--
15	SQL_KEY_COLUMN_USAGE	--	--	--	--
16	SQL_REFERENTIAL_CONSTRAINTS	--	--	--	--
17	SQL_TABLE_PRIVILEGES	--	--	--	--
18	SQL_AUDITS	--	--	--	--
19	STATUS_TABLES	Y	Y	Y	--
20	STATUS_INDEXES	--	--	--	--
21	STATUS_CHUNKS	Y	--	--	--
22	STATUS_SYNONYM_DICTIONARIES	--	--	--	--
23	STATUS_COLUMNS	Y	Y	Y	--

Legend:

Y: The dictionary table or system table is modified as part of the version upgrade process.

--: The dictionary table or system table is not modified as part of the version upgrade process.

(6) Checking the viewed tables that might not be re-creatable

The ALTER VIEW statement might be unable to re-create a viewed table that was defined by using an HADB server whose version is earlier than **05-00** if the viewed table satisfies all the conditions shown in **■ Conditions under which a viewed table cannot be re-created**.

■ Conditions under which a viewed table cannot be re-created

1. A derived table is specified in the CREATE VIEW statement.
2. No derived column list is specified for the derived table in Condition 1.
3. The selection expression in the derived query for the derived table in Condition 1 includes at least one occurrence of each of the following specifications:
   - The AS clause includes either of the following items: a column specification that uses EXP*nnnn*_NO_NAME as a column name, or a value expression that uses EXP*nnnn*_NO_NAME as a column name.

     The *nnnn* portion is an unsigned integer in the range from 0001 to 1000.
   - A value expression that is not a column specification and does not include the AS clause

For a viewed table that cannot be re-created by using the ALTER VIEW statement, use the DROP VIEW statement to delete the viewed table, and then use the CREATE VIEW statement to redefine a viewed table. The following shows the procedure.

■ **Procedure for redefining a viewed table**

1. Check for viewed tables that cannot be re-created.

   Check whether the viewed tables that were defined by an HADB server whose version is earlier than **05-00** include viewed tables that satisfy the conditions under which a viewed table cannot be re-created.

2. Modify the CREATE VIEW statements for the viewed tables.

   You can check the CREATE VIEW statements for viewed tables in the DEFINE_SOURCE column of the dictionary table SQL_DEFINE_SOURCE. Before you check the DEFINE_SOURCE column, see (31) Finding out viewed table definition information in B.22 Searching a dictionary table.

   Use either of the following methods to modify the CREATE VIEW statements for the viewed tables that you checked in step 1:

   - Change the column name EXP*nnnn*_NO_NAME specified in the selection expression in the derived query for the derived table.
   - Add the AS clause to the selection expression in the derived query or add a derived column list to the derived table to prevent the name of the derived column from being EXP*nnnn*_NO_NAME.

3. Delete the viewed table.

   Use the DROP VIEW statement to delete the viewed tables that you checked in step 1.

4. Redefine the viewed tables.

   Redefine the viewed tables by executing the CREATE VIEW statement that you modified in step 2.

5. Re-create the viewed tables that have been disabled.

   If viewed tables that were deleted in step 3 are disabled, use the ALTER VIEW statement to re-create viewed tables in ascending order of view level.

(7) Re-creation of a viewed table (if the version is upgraded from a version earlier than 05-01)

The number of conditions for expanding the internal derived table for a query name was increased in version **05-01**. However, the added conditions are not applied to viewed tables that were defined in a version earlier than **05-01**. Therefore, to apply the conditions added in version **05-01** to such viewed tables, re-create the viewed tables by using the following procedure:

Procedure

1. Confirm the relevant viewed tables.

   Viewed tables that satisfy all of the following conditions are subject to re-creation:

   - Viewed table for which the WITH clause is specified
   - Viewed table defined in a version earlier than **05-01**

   To confirm the CREATE VIEW statements that were used when viewed tables were defined, see the DEFINE_SOURCE column of the dictionary table SQL_DEFINE_SOURCE. To confirm the versions in which viewed tables were defined, see the DEFINE_VR column of the dictionary table SQL_DEFINE_ENVIRONMENT. For details about how to confirm these with the DEFINE_SOURCE column and the DEFINE_VR column, see (31) Finding out viewed table definition information in B.22 Searching a dictionary table.

2. Re-create the relevant viewed tables.

Use the ALTER VIEW statement to re-create the relevant viewed tables in ascending order of view levels.

3. Re-create the viewed tables that have been disabled.

   If there are viewed tables that were disabled by the ALTER VIEW statement executed in step 2, use the ALTER VIEW statement to re-create them in ascending order of view levels.

(8) Notes about the JDBC driver (if the version is upgraded from a version earlier than 05-01)

Provision of the JDBC driver for JRE 6 or 7 (adbjdbc.jar) was discontinued in version **05-01**. If the JDBC driver for JRE 6 or 7 is used, replace it by the JDBC driver for JRE 8 (adbjdbc8.jar). To change the JDBC driver to be used, change the JAR file that is referenced by the application programs (from adbjdbc.jar to adbjdbc8.jar). Changing adbjdbc.jar to adbjdbc8.jar does not affect the application programs. Note that, to use adbjdbc8.jar, Java 8 or later is required.

8.7 Downgrading the HADB server version (restoring the previous version)

This section explains how to restore the previous version of HADB server after a version upgrade. The process of restoring the previous version of the HADB server is called a version downgrade.

Note that new facilities available after a version upgrade cannot be used by a previous version of HADB server.

Important

If you need to restore the previous version because the version upgrade process failed, see 8.6.4 Steps to take when version upgrading fails.

The version downgrading method depends on whether a full backup of the database had been acquired before version upgrade was performed. The following table explains how to perform a version downgrade depending on whether a full backup of the database was acquired.

Table 8-30: How to downgrade the HADB server version

No.	Whether a full backup of the database was acquired	Version downgrading method
1	Y	The backup of the previous version of the database is used to perform the version downgrade. For details, see 8.7.1 Using a backup of the previous version of the database.
2	N	The database must be rebuilt on the previous version of HADB server because a backup of the previous version of database was not acquired. For details, see 8.7.2 Rebuilding the database by using the previous version.

Legend:

Y: A full backup was acquired.

N: A full backup was not acquired.

Note

For details about how to back up the database, see 10.3 Backing up a database. If you are using the multi-node function, see 16.8 Backing up a database (when the multi-node function is being used). If HADB server is in a cold standby configuration, see 17.6 Backing up a database (in the case of the cold standby configuration).

8.7.1 Using a backup of the previous version of the database

This subsection explains how to downgrade the HADB server version by using a backup of the previous version of the database. This method assumes that you have acquired a full backup of the previous version of the database prior to performing a version upgrade.

Note

If you have not acquired a full backup of the previous version of the database, you will have to use the previous version to rebuild the database and then perform version downgrading. For details, see 8.7.2 Rebuilding the database by using the previous version.

(1) Tasks prior to version downgrading

Before you downgrade the HADB server version, make sure that the following tasks have been completed.

(a) Confirming the HADB server's environment settings

Before you downgrade the HADB server version, check that the values of environment variables, kernel parameters, and various server definition operands were not changed during the version upgrade processing. If any of these values were changed, you must restore the values that were being used by the previous version.

(b) Stopping application programs and commands

Before you downgrade the HADB server version, terminate any application programs and commands that are executing.

(c) Terminating the HADB server normally

Before downgrading the HADB server version, use the `adbstop` command to terminate the HADB server normally.

If the HADB server has already stopped, use the following procedure to confirm that it terminated normally:

Procedure

1. Execute the `adbls -d srv` command.
   Confirm that the HADB server's status, which is output to the `STATUS` output item, is `STOP`.
   If it is `STOP`, proceed to step 2 or 3.
2. Check the server message log file.
   Open the server message log file (`$ADBDIR/spool/adbmessage`*XX*`.log`#) and confirm that the termination mode reported in the `KFAA91154-I` message was `normally`.
   #
   XX is a sequential number between `01` and `04`.
3. Check syslog.
   Open syslog and confirm that the termination mode reported in the `KFAA91154-I` message was `normally`.

If the result in step 1 is `STOP` and the result in step 2 or 3 is `normally`, the HADB server has terminated normally.

If the HADB server did not terminate normally, use the `adbstart` command to start it, and then use the `adbstop` command to terminate it normally.

Multi-node function
If the HADB server in the multi-node configuration has not terminated normally, start the HADB servers on all nodes and then terminate the HADB servers on all nodes normally. In this case, there is no need to terminate HA Monitor (no need to execute the `monstop` command).

For details about how to start the HADB server in a multi-node configuration and how to terminate it normally, see 16.4.1 Starting the HADB servers in the multi-node configuration and 16.4.2 Terminating HADB servers in the multi-node configuration.

Cold standby configuration

If the HADB server in a cold standby configuration has not terminated normally, start all HADB servers on all hosts, and then terminate all of them normally. In this case, there is no need to terminate HA Monitor (no need to execute the `monstop` command).

For details about how to start the HADB server in a cold standby configuration and how to terminate it normally, see 17.4.1 How to start the cold standby configuration and 17.4.2 How to terminate the cold standby configuration.

(d) Backing up the server directory

Before downgrading the HADB server version, you also need to back up the server directory.

Before you downgrade the HADB server version, delete the server directory, because the HADB server needs to be uninstalled. When the server directory is deleted, files such as definition files and message log files are also deleted. Make a backup of the server directory in case these files are needed after version downgrade.

For details about the procedure for acquiring a backup of the server directory, see (2) Backup procedure in 10.3.1 Backup acquisition method.

(2) Downgrading the version

The procedure below explains how to downgrade the HADB server version.

Multi-node function

If you are using the multi-node function, perform this step on all HADB servers on all nodes.

Cold standby configuration

If you are using a cold standby configuration, perform this step on all HADB servers on all hosts.

Procedure

1. Uninstall the HADB server.

   Uninstall the HADB server as explained in 8.12 Uninstallation.

2. Install the previous version of the HADB server.

   Install the previous version of the HADB server. For details about how to install the HADB server, see 8.2 Installing the HADB server.

3. Recover the database from its backup.

   Recover the database from the backup you made earlier.

   For details about how to recover the database from a backup, see 10.3.2 Recovering the database from the backup.

   If you are using the multi-node function, see 16.8.2 Recovering a database from a backup. If HADB server is in a cold standby configuration, see 17.6.2 Recovering a database from a backup.

4. Configure an environment for the HADB server.

   Configure an environment for the previous version of the HADB server. For environment variables, kernel parameters, and server definition operands, specify the same values as were in effect for the previous version.

5. Start the HADB server.

   Use the `adbstart` command to start the HADB server.

   If you acquired the backup in the quiescence mode, you will have to set the HADB server operation mode to normal after the HADB server has started. To do this, use the `adbchgsrvmode` command to change the mode to normal.

Important

When the HADB server version is downgraded by using a backup of the previous version of the database, the contents of the database are restored to the point at which the backup was acquired.

(3) Tasks after a version downgrade

Once you have downgraded the HADB server version, you must also downgrade the HADB client versions (to match the version of the HADB server).

For details about how to downgrade HADB client versions, see *Downgrading an HADB client version (restoring the previous version)* in the *HADB Application Development Guide*.

8.7.2 Rebuilding the database by using the previous version

This subsection explains how to downgrade the HADB server version by rebuilding the database on the previous version of the HADB server. You use this method when you do not have a full backup of the previous version of the database.

Note

If you have acquired a full backup of the previous version of the database, you can use that backup to downgrade the HADB server version. For details, see 8.7.1 Using a backup of the previous version of the database.

(1) Tasks prior to version downgrade

Before you downgrade the HADB server version, make sure that the following tasks have been completed.

(a) Confirming the HADB server's environment settings

Before you downgrade the HADB server version, check that the values of environment variables, kernel parameters, and various server definition operands were not changed during the version upgrade processing. If any of these values were changed, you must restore the values that were being used by the previous version.

(b) Checking the status of DB areas (checking the initialization options)

Before you downgrade the HADB server version, use the `adbdbstatus` command to check the status of DB areas. This step is necessary when you execute the `adbinit` command with initialization options specified to rebuild the database on the previous version of the HADB server.

Apply the `adbdbstatus` to the past month to check DB area summary information. Among the output results, check the following output items:

- `DBarea_name` (DB area name)
- `Page_size` (page size)

Then use the `adbdbstatus` command to check Usage information. Among the output results, check the `DBarea_filename` (DB area file name) that corresponds to `DBarea_name` in the DB area summary information.

> **Note**
>
> For details about the adbdbstatus command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

When you rebuild the database, specify the initialization options of the adbinit command on the basis of the output items checked by the adbdbstatus command. The following table shows the correspondence of the output items of the adbdbstatus command to the initialization options of the adbinit command.

Table 8-31: Correspondence of the output items of the adbdbstatus command to the initialization options of the adbinit command

No.	Output item of the adbdbstatus command	Initialization option of the adbinit command
1	DBarea_name (DB area name)	-n option of adbinitdbarea
2	Page_size (page size)	-p option of adbinitdbarea#
3	DBarea_filename (DB area file name)	-v option of adbinitdbarea

#

Specify the -p option of the adbinitdbarea initialization option in the adbinit command in kilobytes. Because the value of Page_size (page size) that is output by the adbdbstatus command is in bytes, you must convert this value to kilobytes and then specify the obtained value in the -p option. The following is the formula for converting a value in bytes to a value in kilobytes:

Formula (kilobytes)

Value of page_size ÷ 1,024

(c) Checking the definition SQL statements

Before you downgrade the HADB server version, see the dictionary tables to check the definition SQL statements that need to be executed in the previous version of HADB server.

For details about dictionary tables, see B. Dictionary Tables.

(d) Exporting data

Before you downgrade the HADB server version, use the adbexport command to export all data from base tables.

If you want to maintain the chunk structure when you export base table data that had been stored by background import processing, execute the adbexport command with the -c option specified.

For details about the adbexport command, see *adbexport (Export Data)* in the manual *HADB Command Reference*.

(e) Stopping application programs and commands

Before you downgrade the HADB server version, terminate any application programs and commands that are executing.

(f) Terminating the HADB server normally

Before downgrading the HADB server version, use the adbstop command to terminate the HADB server normally.

If the HADB server has already stopped, use the following procedure to confirm that it terminated normally:

1. Execute the `adbls -d srv` command.

   Confirm that the HADB server's status, which is output to the `STATUS` output item, is `STOP`.

   If it is `STOP`, proceed to step 2 or 3.

2. Check the server message log file.

   Open the server message log file (`$ADBDIR/spool/adbmessage`*XX*`.log`[#]) and confirm that the termination mode reported in the `KFAA91154-I` message was `normally`.

   #

   XX is a sequential number between `01` and `04`.

3. Check syslog.

   Open syslog and confirm that the termination mode reported in the `KFAA91154-I` message was `normally`.

If the result in step 1 is `STOP` and the result in step 2 or 3 is `normally`, the HADB server has terminated normally.

If the HADB server did not terminate normally, use the `adbstart` command to start it, and then use the `adbstop` command to terminate it normally.

Multi-node function

If the HADB server in the multi-node configuration has not terminated normally, start the HADB servers on all nodes and then normally terminate the HADB servers on all nodes. In this case, there is no need to terminate HA Monitor (no need to execute the `monstop` command).

For details about how to start the HADB server in a multi-node configuration and how to terminate it normally, see 16.4.1 Starting the HADB servers in the multi-node configuration and 16.4.2 Terminating HADB servers in the multi-node configuration.

Cold standby configuration

If the HADB server in a cold standby configuration has not terminated normally, start all HADB servers on all hosts, and then terminate all of them normally. In this case, there is no need to terminate HA Monitor (no need to execute the `monstop` command).

For details about how to start the HADB server in a cold standby configuration and how to terminate it normally, see 17.4.1 How to start the cold standby configuration and 17.4.2 How to terminate the cold standby configuration.

(g) Acquiring a server directory backup

Before downgrading the HADB server version, you also need to back up the server directory.

When you downgrade the HADB server version, you delete the server directory because the HADB server needs to be uninstalled. When the server directory is deleted, files such as definition files and message log files are also deleted. Make a backup of the server directory in case these files are needed after version downgrade.

For details about the procedure for acquiring a backup of the server directory, see (2) Backup procedure in 10.3.1 Backup acquisition method.

(2) Downgrading the version

The procedure below explains how to downgrade the HADB server version.

Multi-node function

If you are using the multi-node function, perform this step on all HADB servers on all nodes.

Cold standby configuration

If you are using a cold standby configuration, perform this step on all HADB servers on all hosts.

Procedure

1. Uninstall the HADB server
   Uninstall the HADB server as explained in 8.12 Uninstallation.
2. Build the database.
   Build the database on the previous version of the HADB server based on the items confirmed in (1) Tasks prior to version downgrade. For details about the procedure for building a database, see 8.1 System construction procedure.

> **Note**
>
> If you want to restore a multi-chunk table with the previous chunk structure, execute the `adbimport` command to which the background-import facility applied (with the `-b` option specified) as many times as needed. Note that chunk IDs cannot be restored.
>
> For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

(3) Tasks after a version downgrade

Once you have downgraded the HADB server version, you must also downgrade the HADB client versions (to match the version of the HADB server).

For details about how to downgrade HADB client versions, see *Downgrading an HADB client version (restoring the previous version)* in the *HADB Application Development Guide*.

8.7.3 If the previous version needs to be restored because a version upgrade has failed

If you need to restore the previous version because the version upgrade process failed, see 8.6.4 Steps to take when version upgrading fails.

8.8 Swapping the HADB server with its revised version

This section describes how to swap the HADB server with its revised version.

If the following condition is met, you need to swap the HADB server with its revised version rather than upgrading the HADB server version.

- You reinstall the HADB server by overwriting the existing one when both HADB servers have the same version number and revision number and only their codes are different.

> **Note**
>
> The following describes the version number, revision number, and code.
>
> Example: If the HADB server version information is 03-02-/A
>
> - The version number is 03.
> - The revision number is 02.
> - The code is -/A in the underlined part.

The following describes example cases in which swapping of the HADB server with its revised version is needed, and example cases in which such swapping is not used.

▪ Cases in which swapping of the HADB server with its revised version is needed

In the following cases, for example, swapping of the HADB server with its revised version is needed because the version number and revision number are the same.

03-02 -> 03-02-/A

03-02-/B -> 03-02-/D

▪ Cases in which swapping of the HADB server with its revised version is not used

In the following cases, for example, swapping of the HADB server with its revised version is not used because the version number or revision number is different.

03-02 -> 03-03-/A

03-02-/B -> 03-03-/D

In such cases, you need to upgrade the HADB server version. For details about how to upgrade the HADB server version, see 8.6 Upgrading the HADB server version.

8.8.1 Steps to take before swapping the HADB server with its revised version

The following describes the steps you must take before swapping the HADB server with its revised version.

> **Important**
>
> Before swapping the HADB server with its revised version, do not change the values specified in the environment variables `ADBLANG` and `ADBCLTLANG`. If they are changed, you might not be able to correctly swap the HADB server.
>
> The language code and nation code specified in the `LANG` environment variable can be changed as follows:

- Change from `ja_JP.UTF-8` to `en_US.UTF-8`
- Change from `en_US.UTF-8` to `ja_JP.UTF-8`

The encoding set (type of character encoding) cannot be changed.

Note that if you change the value of the `LANG` environment variable, you also need to change the value of the `LANG` environment variable for the HADB client.

(1) Checking the memory requirement

Swapping the HADB server with its revised version might increase the HADB server's memory requirement. For details, see the Release Notes.

(2) Checking whether environment variables need to be changed

If you swap the HADB server with its revised version, you might need to change environment variables. For details, see the Release Notes.

(3) Checking whether kernel parameters need to be changed

If you swap the HADB server with its revised version, you might need to change kernel parameters. For details, see the Release Notes.

(4) Checking whether the default values of operands in the server definition have been changed

Swapping the HADB server with its revised version might change the default values of operands in the server definition. For details, see the Release Notes.

(5) Checking whether reserved words have been added

Swapping the HADB server with its revised version might add reserved words. You cannot specify a reserved word in an SQL statement (for example, as a table or column name). If a character string that is the same as an added reserved word is specified in an SQL statement, the SQL statement will cause an error after the HADB server is swapped with its revised version. Therefore, check whether a character string that is the same as an added reserved word is specified in an SQL statement. If a character string that is the same as an added reserved word is specified, enclose it in double quotation marks (`"`). For details, see *Reserved words* in *SQL Basics* in the manual *HADB SQL Reference*.

(6) Terminating application programs and commands normally

Before you swap the HADB server with its revised version, terminate the application programs and commands normally.

(7) Terminating the HADB server normally

Before you swap the HADB server with its revised version, use the `adbstop` command to terminate the HADB server normally. If the HADB server has already stopped, use the following procedure to confirm that it terminated normally.

Procedure:

1. Execute the `adbls -d srv` command.

Confirm that the HADB server status, which is output to the output item `STATUS`, is `STOP`.

If it is `STOP`, proceed to step 2 or 3.

2. Check the server message log file.

   Open the server message log file `$ADBDIR/spool/adbmessage`*XX*`.log`[#]) and confirm that the termination mode, output in the `KFAA91154-I` message, is `normally`.

   #: *XX* is a sequential number in the range from 01 to 04.

3. Check `syslog`.

   Open `syslog` and confirm that the termination mode, output in the `KFAA91154-I` message, is `normally`.

If the result in step 1 is `STOP` and the result in step 2 or 3 is `normally`, the HADB server has terminated normally.

If the HADB server has not terminated normally, use the `adbstart` command to start it, and then use the `adbstop` command to terminate it normally.

(8) Backing up the server directory

Before swapping the HADB server with its revised version, you need to back up the server directory.

If you swap the HADB server with the revised version while the HADB server is in abnormally terminated status, you will not be able to start the HADB server. To prevent such problems, make a backup of the server directory. Delete the backup after you finish swapping the HADB server with its revised version.

For details about the procedure for acquiring a backup, see (2) Backup procedure in 10.3.1 Backup acquisition method.

Multi-node function

If you are using the multi-node function, make a backup of the server directory on the HADB servers on all nodes.

8.8.2 Procedure for swapping the HADB server with its revised version

After you have taken the steps described in 8.8.1 Steps to take before swapping the HADB server with its revised version, swap the HADB server with its revised version by using the following procedure.

Procedure:

1. Log on to the OS from the HADB administrator's OS account.

   Log on to the server machine's OS from the HADB administrator's OS account that you have been using.

   > **Important**
   >
   > Do not modify the HADB administrator's OS account. If you modify the HADB administrator's OS account, you might not be able to correctly swap the HADB server with its revised version.

2. Create the directory for storing the installation command and installation data.

   Execute the OS's `mkdir` command to create the directory for storing the installation command (`adbinstall` command) and installation data (`tar.gz` file).

   ```
   mkdir /home/adbmanager/vup
   ```

   In this example, `/home/adbmanager/vup` is created as the directory for storing the installation command and installation data.

3. To the directory created in step 2, assign permissions that allow writing by the HADB administrator.

   Execute the OS's `chmod` command to assign the directory created in step 2 the permissions that allow writing by the HADB administrator.

```
chmod 755 /home/adbmanager/vup
```

4. Mount the CD-ROM file system.

   Execute the OS's `mount` command to automatically mount the CD-ROM file system containing the installation command and installation data. If the CD-ROM file system is not mounted automatically, you must mount it manually.

```
mount /dev/cdrom /media
```

   The underlined portion shows the name of the mount directory for the CD-ROM file system. It will vary depending on the environment.

> **Important**
>
> Depending on the server machine you are using, the directory and file names used on the CD-ROM might be different from those shown in the preceding command. Enter the directory name that is displayed exactly as it is shown when you execute the `ls` operating system command.

5. Copy the installation command.

   Execute the OS's `cp` command to copy the HADB server installation command (`adbinstall` command) to the directory created in step 2. The installation command is stored on the mounted CD-ROM file system.

```
cp /media/adbinstall /home/adbmanager/vup
```

   The underlined portion shows the name of the mount directory for the CD-ROM file system. It will vary depending on the environment.

6. Copy the installation data.

   Execute the OS's `cp` command to copy the HADB server installation data (`tar.gz` file) to the directory created in step 2. The installation data is stored on the mounted CD-ROM file system.

```
cp /media/hitachi_advanced_data_binder_server-$VR.tar.gz
   /home/adbmanager/vup
```

   The underlined portion shows the name of the mount directory for the CD-ROM file system. It will vary depending on the environment.

   `$VR` indicates the HADB version and release number.

   Example: For 03-02-/A, `0302a-0` is displayed in place of $VR.

> **Important**
>
> Copy the installation command and installation data to the same directory. If they are copied to different directories, you cannot swap the HADB server with its revised version.

7. Assign the installation command the permissions that allow execution by the HADB administrator.

   Assign the installation command the permissions that allow execution by the HADB administrator.

```
chmod 777 /home/adbmanager/vup/adbinstall
```

8. Execute the installation command.

   The HADB administrator must execute the installation command to install the revised version of the HADB server.

```
/home/adbmanager/vup/adbinstall -s /HADB/server
```

The underlined portion is the path name of the directory to which the installation command has been copied.

For the -s option, specify the absolute path for the server directory being used (in this example, /HADB/server).

Important

- If the installation command causes an error and the KFAA91553-E message is output, see **■ If the installation command causes an error (the KFAA91553-E message is output)**.
- If the KFAA91558-W message (warning message) is output when you execute the installation command, see **■ If the message KFAA91558-W (warning message) is output**.

9. Respond to the KFAA91554-Q message.

   When you execute the installation command, the KFAA91554-Q message is output, asking whether you want to overwrite the directory. Enter y or Y to overwrite the directory.

10. Confirm that the revised version of the HADB server has been installed.

    Execute the adbls -d ver command to check the HADB server version information.

```
adbls -d ver
```

Multi-node function

If you are using the multi-node function, check the HADB server version information for all nodes by executing the preceding command on all HADB servers on all nodes.

If no version information is displayed even when you execute the adbls -d ver command, there might be a specification error in one of the following environment variables:

- The server directory has not been set in the ADBDIR environment variable.
- The bin directory under the server directory has not been correctly set in the PATH environment variable.

For details about environment variables, see 8.4 Setting environment variables.

11. Start the HADB server normally.

    Execute the adbstart command to start the HADB server normally.

```
adbstart
```

If the HADB server starts successfully, swapping of the HADB server with its revised version is complete.

■ If the installation command causes an error (the KFAA91553-E message is output)

In the following cases, the installation command causes an error and the KFAA91553-E message is output. In such cases, assign the target directory the permissions that allow writing by the HADB administrator.

- The HADB administrator does not have write privileges for the directory specified in the -s option.
- The HADB administrator does not have write privileges for the directory containing the installation command and installation data.

■ If the message KFAA91558-W (warning message) is output

If you execute the adbinstall command as root rather than using the HADB administrator's OS account, the KFAA91558-W message (warning message) is output. Normally, the HADB administrator's OS account is used to execute the adbinstall command. Therefore, if the KFAA91558-W message is output, make sure that executing the adbinstall command as the root user (root) does not cause any problems.

If a problem exists, enter n or N in response to the input request in the KFAA91559-Q message that is output after the KFAA91558-W message. Then, execute the adbinstall command by using the HADB administrator's OS account.

If you execute the adbinstall command as a superuser other than root, the KFAA91558-W message is not output.

Note

root indicates a user for which 0 is displayed by executing the OS's id -u command. This includes a case where 0 is displayed by executing the OS's id -u command after another OS user was changed to root by using the OS's su command.

8.9 Temporarily upgrading the HADB server version and then downgrading it (test for checking operational behavior)

This section describes how to temporarily upgrade the HADB server version for test purposes, such as checking operational behavior of applications, and then restore the previous version.

The procedure differs depending on whether you update the database[#] when checking operational behavior.

If you do not update the database when checking operational behavior, use the procedures described in sections 8.9.1 Steps to take before upgrading the server version (test for checking operational behavior) to 8.9.5 Steps to take after downgrading the server version (test for checking operational behavior). If you do not update the database when checking operational behavior, upgrade the HADB server version and restore the previous version in accordance with the procedures described.

If you update the database when checking operational behavior, perform the following procedure.

■ If you update the database when checking operational behavior

Upgrade the HADB server version and downgrade it by performing the following procedure:

1. Upgrade the HADB server version.
   Perform the following procedures. When you back up the database, make sure that you create a full backup.
   - 8.6.1 Steps to take before upgrading the server version
   - 8.6.2 Upgrading the server version
   - 8.6.3 Steps to take after version upgrading
2. Perform a test for checking operational behavior.
3. Restore the previous HADB server version.
   Perform the procedure described in 8.7.1 Using a backup of the previous version of the database.

#

Executing any of the following SQL statements or commands updates the database:

- Update SQL statements
- Definition SQL statements
- `adbimport` command
- `adbidxrebuild` command
- `adbgetcst` command
- `adbmodarea` command
- `adbmergechunk` command
- `adbchgchunkcomment` command
- `adbchgchunkstatus` command
- `adbreorgsystemdata` command
- `adbarchivechunk` command
- `adbunarchivechunk` command
- `adbsyndict` command

The database is also updated if the updated-row columnizing facility is enabled. For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).

8.9.1 Steps to take before upgrading the server version (test for checking operational behavior)

Perform the procedure described in 8.6.1 Steps to take before upgrading the server version.

When you back up the database, make sure that you back up the system data only.

8.9.2 Upgrading the server version (test for checking operational behavior)

Upgrade the version of the HADB server in accordance with the procedure described in 8.6.2 Upgrading the server version. Note that the step *Upgrade the database version* in 8.6.2 Upgrading the server version must be replaced by the following procedure.

Procedure for upgrading the database version (test for checking operational behavior)

Execute the following HADB command to start the HADB server. Version upgrading starts.

```
adbstart --quiescence
```

After execution of the `adbstart` command, the `KFAA91107-Q` message is output, asking whether you want to upgrade the database version. Press **Y** to upgrade the database version.

If the `adbstart` command terminates normally, upgrading of the HADB server version is complete.

Multi-node function

If you are using the multi-node function, start the HADB servers in the multi-node configuration as explained in 16.4.1 Starting the HADB servers in the multi-node configuration. On the master node, however, execute the following HADB command:

```
adbstart --quiescence
```

If you attempt to start the HADB server on the master node, the `KFAA91107-Q` message asking whether the database version is to be upgraded is displayed. Press **Y** to upgrade the database version.

If the HADB servers on all nodes start successfully, the HADB server version has been upgraded.

8.9.3 Steps to take after upgrading the server version (test for checking operational behavior)

Upgrade the version of the HADB client in accordance with the procedure in (2) Upgrading the HADB client version in 8.6.3 Steps to take after version upgrading.

After that, perform a test for checking operational behavior.

8.9.4 Downgrading the server version (test for checking operational behavior)

After performing a test for checking operational behavior, downgrade the version of the HADB server in accordance with the procedure described in (2) Downgrading the version in 8.7.1 Using a backup of the previous version of the database.

When you restore the database from a backup, make sure that you use a backup of system data only.

8.9.5 Steps to take after downgrading the server version (test for checking operational behavior)

Downgrade the version of the HADB client in accordance with the procedure in (3) Tasks after a version downgrade in 8.7.1 Using a backup of the previous version of the database.

8.10 Changing the time in the server machine's OS

This section explains how to change the time in the OS of the server machine on which the HADB server is installed.

When changing the time in the server machine's OS, read 8.10.1 Notes (when changing the OS time) first and then use either of the following methods to change the time:

- 8.10.2 Moving the time forward in the server machine's OS
- 8.10.3 Moving back the time in the server machine's OS

8.10.1 Notes (when changing the OS time)

Read the following notes before changing the time in the server machine's OS:

- Change the time in the server machine's OS only after the HADB server has terminated normally. If you change the time while the HADB server is running, it and any currently running HADB clients might behave in unexpected ways.
- After the time in the server machine's OS is changed, the following times are displayed as the changed OS time, which might affect time-sensitive application programs:
  - Time acquired by an SQL statement
  - Time when a command is executed
  - Time stamp stored in a dictionary table
  - Time stamp stored in a system table
- If you use an NTP program to correct the OS time, select the delay adjustment method (slew method) (avoid a backward change in time or a large time change).

8.10.2 Moving the time forward in the server machine's OS

Read 8.10.1 Notes (when changing the OS time), and then move forward the time in the OS of the server machine on which the HADB server is installed by following this procedure:

Procedure

1. Terminate the HADB server normally.

   Execute the `adbstop` command to terminate the HADB server normally. For details about the `adbstop` command, see 10.2.2 Terminating the HADB server.

2. Move the time forward in the server machine's OS.

   Once the HADB server has terminated normally, move the time forward in the OS of the server machine on which the HADB server is installed.

3. Start the HADB server.

   After you have moved the time forward in the server machine's OS, execute the `adbstart` command to start the HADB server. For details about the `adbstart` command, see 10.2.1 Starting the HADB server.

You have now moved the time forward in the server machine's OS.

8.10.3 Moving back the time in the server machine's OS

Read 8.10.1 Notes (when changing the OS time), and then move back the time in the OS of the server machine on which the HADB server is installed by performing either of the following procedures:

- Setting the clock back in the OS by a few seconds to a few minutes (when you can wait for the amount of time that was moved back)
- Starting the HADB server immediately (when you cannot wait for the amount of time by which the clock was set back)

(1) Setting the clock back in the OS by a few seconds to a few minutes (when you can wait for the amount of time that was moved back)

Procedure

1. Terminate the HADB server normally.

   Execute the `adbstop` command to terminate the HADB server normally. For details about the `adbstop` command, see 10.2.2 Terminating the HADB server.
2. Set back the clock in the server machine's OS.

   Move back the time in the OS on the server machine on which the HADB server is installed.
3. Wait for at least the amount of time that was moved back (the amount of the time change), and then start the HADB server.

   Wait for at least the amount of time by which the time was changed, and then execute the `adbstart` command to start the HADB server. For example, if you have moved back the time in the OS by 10 minutes, wait for at least 10 minutes, and then start the HADB server.

   For details about the `adbstart` command, see 10.2.1 Starting the HADB server.

You have now set back the clock in the server machine's OS.

If you are setting back the clock by an amount of time that is too long to wait for, see (2) Starting the HADB server immediately (when you cannot wait for the amount of time that was moved back).

(2) Starting the HADB server immediately (when you cannot wait for the amount of time that was moved back)

Procedure

1. Terminate the HADB server normally.

   Execute the `adbstop` command to terminate the HADB server normally. For details about the `adbstop` command, see 10.2.2 Terminating the HADB server.
2. Back up the `spool` directory in the server directory.

   After the HADB server has terminated normally, use either of the following methods to back up all directories and files that are stored in the `spool` directory in the server directory.(`$ADBDIR/spool` directory):

   - Executing the operating system's `cp` command
   - Executing the `adbinfoget` command

   For details about the `adbinfoget` command, see *adbinfoget (Collect Troubleshooting Information)* in the manual *HADB Command Reference*.
3. Make a backup of the output destination directory for statistics log files.

If you omit the adb_sta_log_path operand in the server definition, ignore this step and go to the next step.

If you specify the adb_sta_log_path operand in the server definition, you need to back up the statistics log files. Use either of the following methods to back up all files that are stored in the output destination directory for statistics log files specified in the adb_sta_log_path operand:

- Executing the operating system's cp command
- Executing the adbinfoget command

For details about the adbinfoget command, see *adbinfoget (Collect Troubleshooting Information)* in the manual *HADB Command Reference*.

For details about the adb_sta_log_path operand in the server definition, see the description of the adb_sta_log_path operand in 7.2.7 Operands related to statistical information (set format).

4. Delete the information that was output by the HADB server.

   After you have made a backup, execute the adbinfosweep command to delete the information that was output by the HADB server. For details about the adbinfosweep command, see *adbinfosweep (Delete Troubleshooting Information)* in the manual *HADB Command Reference*.

   Important

   If you do not execute the adbinfosweep command, the first time the HADB server is started after the time in the server machine's OS is moved back, the HADB server and HADB clients might behave in an unexpected way. Before you execute the adbinfosweep command, back up the information that was output by the HADB server, because executing this command will delete the output information.

5. Delete the spool directory from the server directory.

   Use an OS command to delete the spool directory ($ADBDIR/spool directory) from the server directory.

6. Delete the statistics log files.

   If you omit the adb_sta_log_path operand in the server definition, ignore this step and go to the next step.

   If you specify the adb_sta_log_path operand in the server definition, you need to delete the statistics log files. Delete all files that are stored in the output destination directory for statistics log files specified in the adb_sta_log_path operand. For details about how to delete statistics log files, see (4) Deleting statistics log files in 10.10.5 Using the statistics log files.

7. Create an empty spool directory.

   Use an OS command to create an empty spool directory under the server directory.

8. Move back the time in the server machine's OS.

   After you have deleted the information that was output by the HADB server, move back the time in the OS of the server machine on which the HADB server is installed.

9. Start the HADB server.

   After you have moved back the time in the server machine's OS, execute the adbstart command to start the HADB server. For details about the adbstart command, see 10.2.1 Starting the HADB server.

You have now moved back the time in the server machine's OS.

8.11 Changing the host name or IP address of the server machine's OS

This section explains how to change the host name or IP address of the OS of the server machine on which the HADB server is installed.

To change the host name or IP address of the server machine's OS, you must change the settings of both the HADB server and HADB client.

■ **Tasks to be performed on the HADB server**

The following explains the tasks to be performed on the HADB server to change the host name or IP address of the server machine's OS.

1. **Terminate the HADB server normally.**

   Execute the `adbstop` command to terminate the HADB server normally. For details about the `adbstop` command, see 10.2.2 Terminating the HADB server.

2. **Change the host name or IP address of the server machine's OS.**

   When the HADB server has terminated normally, change the host name or IP address of the OS of the server machine on which the HADB server is installed.

3. **Start the HADB server.**

   After changing the host name or IP address of the server machine's OS, execute the `adbstart` command to start the HADB server. For details about the `adbstart` command, see 10.2.1 Starting the HADB server.

> **Note**
>
> You can change the host name or IP address of the server machine's OS without terminating the HADB server if all of the following conditions are met:
>
> - The multi-node function is not used.
> - The audit trail facility is not used.
> - In a cold standby configuration, both of the following two conditions are met:
>   - The host name of the LAN to be used as the monitoring path of HA Monitor is not changed.
>   - The alias IP address used for communication between the HADB client and the HADB server is not changed.
>
> Note the following if you change the host name or IP address without terminating the HADB server: The HADB clients (application programs) connected to the HADB server become unable to communicate with the HADB server when the host name or IP address is changed.

■ **Tasks to be performed on the HADB client**

The following explains the tasks to be performed on the HADB client to change the host name or IP address of the server machine's OS.

1. **Terminate the application programs.**

   Terminate all application programs that connect to the HADB server.

2. **Change the host name or IP address of the HADB server specified in the client definition.**

   After terminating the application programs, change the host name or IP address of the HADB server specified for the `adb_clt_rpc_srv_host` operand in the client definition.

   For details about the `adb_clt_rpc_srv_host` operand in the client definition, see *Contents of operands in the client definition* under *Designing Client Definitions* in the *HADB Application Development Guide*.

- If the JDBC driver is used

  Change the host name or IP address of the HADB server specified with the property that corresponds to the `adb_clt_rpc_srv_host` operand in the client definition. The JDBC driver can use a system property, user property, or connection URL property. For details, see *Setting Up an Environment for the JDBC Driver* in the *HADB Application Development Guide*.

- If the ODBC driver or a CLI function is used

  Change the host name or IP address of the HADB server specified for the `adb_clt_rpc_srv_host` operand in the client definition. For details, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

3. **Start the application programs.**

   After changing the host name or IP address of the HADB server specified in the client definition, start the application programs that are to be connected to the HADB server.

Important

- If the system uses the multi-node function, see 16.18 Changing the host name or IP address of the server machine's OS (when the multi-node function is being used).
- If the system is in a cold standby configuration, see 17.15 Changing the host name or IP address of the server machine's OS (in the case of the cold standby configuration).

8.12 Uninstallation

This section explains how to uninstall the HADB server and the tasks that must be performed following uninstallation.

If you are using the multi-node function, perform the uninstallation tasks explained in this section on all nodes.

8.12.1 Uninstallation procedure

Uninstall the HADB server by following the procedure described below. If the server directory contains files that are important to save, back them up before proceeding with uninstallation.

> **Important**
>
> Before uninstalling the HADB server, stop all commands and applications. Otherwise, the uninstallation might fail.

(1) Deleting the server directory

Delete the server directory by executing the following operating system command as the HADB administrator. You can now uninstall the HADB server.

```
rm -rf server-directory-path-name
```

An example follows in which `/HADB/server` is specified for *server-directory-path-name*.

■ Command execution example

```
rm -rf /HADB/server
```

(2) Deleting the installation data (adbinstall command and tar.gz file)

As the HADB administrator, execute the following operating system command to delete the installation data (the `adbinstall` command and the `tar.gz` file) that was used when installing the HADB server:

```
rm -rf path-name-of-selected-directory-for-storing-installation-data
```

In the following example, `/home/adbmanager/install` is specified for *path-name-of-selected-directory-for-storing-installation-data*.

■ Command execution example

```
rm -rf /home/adbmanager/install
```

8.12.2 Tasks that must be performed following uninstallation

Once you have uninstalled the HADB server successfully, perform the following tasks:

- Delete unneeded environment variables

- Delete unneeded directories and files
- Delete the ITRU collection pattern definition file
- Delete the output destination directory for statistics log files specified in the `adb_sta_log_path` operand in the server definition
- Delete the directory for storing the server directory

(1) Deleting unneeded environment variables

Of the environment variables set during installation of the HADB server according to 8.4 Setting environment variables, delete those that are no longer needed. Perform this task as the HADB administrator.

(2) Deleting unneeded directories and files

Delete unneeded directories and files as a superuser. The directories and files to be deleted vary depending on the version of the OS of the server machine.

- If the server machine's OS is Red Hat Enterprise Linux Server 6 (64-bit x86_64)
  See (a) Red Hat Enterprise Linux Server 6.
- If the server machine's OS is Red Hat Enterprise Linux Server 7 (64-bit x86_64)
  See (b) Red Hat Enterprise Linux Server 7.

(a) Red Hat Enterprise Linux Server 6

Delete the following directories as a superuser.

▪ Directories that must be deleted

- Output destination directory for error information (core files)
- Directories for storing communication-information files
  - `/dev/HADB/pth`
  - `/lib/udev/devices/HADB/pth`

If any files remain under these directories, also delete those files.

(b) Red Hat Enterprise Linux Server 7

Delete the following directories and files as a superuser.

▪ Directories and files that must be deleted

- Output destination directory for error information (core files)
- Directories and files for storing communication-information files
  - `/dev/HADB/pth`
  - `/etc/tmpfiles.d/dev-HADB-pth.conf`

If any files remain under these directories, also delete those files.

(3) Deleting the ITRU collection pattern definition file

If ITRU is installed on the server machine from which you have uninstalled the HADB server, you need to delete the ITRU collection pattern definition file (`!8A9_HADB`).

Delete the following file as a superuser.

▪ ITRU collection pattern definition file that must be deleted

`/etc/opt/hitachi/systoru/pattern/!8A9_HADB`

(4) Deleting the output destination directory for statistics log files specified in the `adb_sta_log_path` operand in the server definition

If the `adb_sta_log_path` operand in the server definition was specified on the HADB server that you have uninstalled, delete the output destination directory for statistics log files. Perform this operation as the HADB administrator.

You also need to delete any files under the output destination directory for statistics log files.

(5) Deleting the directory for storing the server directory

Delete the directory for storing the server directory (`/HADB`) as a superuser. Delete the directory for storing the server directory (`/HADB`) by entering the following OS command.

■ Command execution example

```
rm -rf /HADB
```

9 Creating a Database

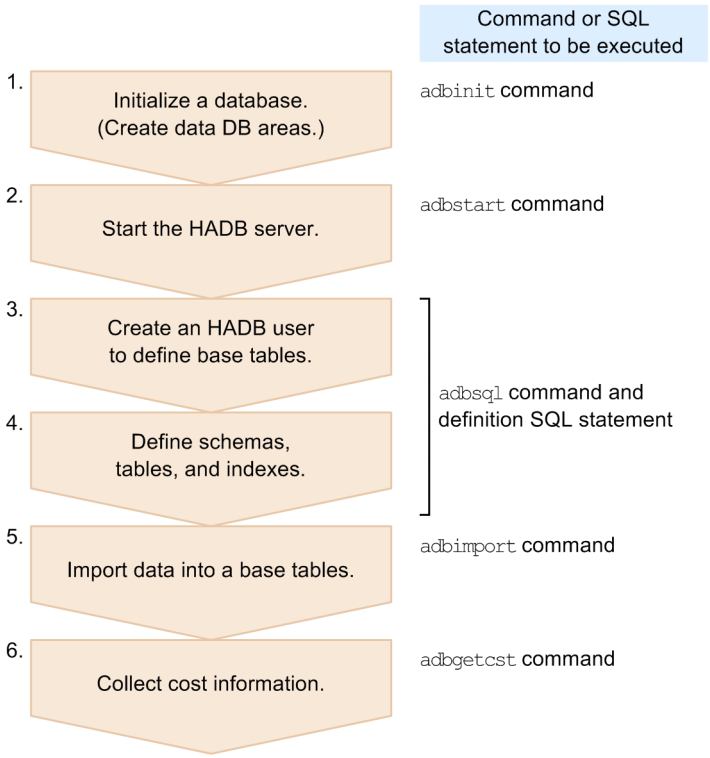
This chapter explains the general procedure for creating a database.

The procedures for building a hands-on environment are explained in 3. Guide for Building a Hands-on Environment. We recommend that you also read this chapter.

9.1 Steps for creating a database

After you finish setting up the environment variables and server definition, create a database by following the steps described in the following figure.

Figure 9-1: Steps for creating a database



These steps are explained in the sections that follow.

9.2 Initializing a database (creating data DB areas)

Use the `adbinit` command to execute database initialization and create data DB areas.

Example:

This example executes database initialization to create data DB areas (`DBAREA01` and `DBAREA02`).

Procedure:

1. Log on to the OS from the HADB administrator's OS account.

   Log on to the server machine's OS from the HADB administrator's OS account so that the HADB administrator can execute the `adbinit` command.

   This step is not necessary if the HADB administrator is already logged on.

2. Create an initialization option file.

```
set adb_init_wrk_page_size = 32
set adb_init_wrk_blk_path = /dev/mapper/VolG00
adbinitdbarea -n DBAREA01 \
    -p 4 \
    -v /dev/mapper/VolG01,/dev/mapper/VolG02 \
    -q \
    -i 500M,2
adbinitdbarea -n DBAREA02 \
    -p 4 \
    -v /dev/mapper/VolG03,/dev/mapper/VolG04,/dev/mapper/VolG05 \
    -q \
    -i 1G,3
```

   Create an initialization option file in which the initialization option is specified. The initialization option file created here becomes the input information for the `adbinit` command.

   In this example, the created initialization option file is saved under `/HADB/server/conf/adbinit.opt`.

   For details about the initialization option, see *Format of initialization options* in *Specification format for the adbinit command* under *adbinit (Initialize the Database)* in the manual *HADB Command Reference*.

3. Execute the `adbinit` command to initialize the database and create a data DB area.

```
adbinit -u ADBUSER01 -p '#HelloHADB_01'
        /HADB/server/conf/adbinit.opt
        /HADB/db
```

   `-u`

   Specifies an authorization identifier for the HADB user to be created.

   `-p`

   Specifies a password for the HADB user to be created.

   `/HADB/server/conf/adbinit.opt`

   Specifies the absolute path name of the initialization option file created in step 2.

   `/HADB/db`

   Specifies a DB directory name.

   For details about the specification rules for authorization identifiers and passwords, see the following subsections:

   - 9.4.2 Authorization identifier specification rules
   - 9.4.3 Password specification rules

Note

The HADB user with the authorization identifier specified in the `-u` option is created as the first HADB user and is registered in the HADB server. This HADB user has both the `DBA` privilege and the `CONNECT` privilege.

9.3 Starting the HADB server

When you have finished database initialization, use the `adbstart` command to start the HADB server.

Procedure:

1. Execute the `adbstart` command.

```
adbstart
```

2. Confirm that the HADB server terminated normally.
   Confirm that the return code in the following messages is `0`:

```
KFAA90000-I adbstart processing started.
KFAA91105-I The HADB system was started normally. (HADB server operation mode = "N
ORMAL")
KFAA90001-I adbstart processing ended. (return code = 0)
```

9.4 Creating an HADB user to define base tables

Next, you create an HADB user to define base tables.

9.4.1 Creating an HADB user

Use a definition SQL statement to create an HADB user, and grant this user both the CONNECT privilege and the schema definition privilege.

Example:

This example creates HADB user ADBUSER02 to define base tables, and grants both the CONNECT privilege and schema definition privilege to ADBUSER02.

Procedure:

1. Execute the adbsql command.

```
adbsql
  USER-ID ?
ADBUSER01       ← Connects to the HADB server as ADBUSER01.
  PASSWORD ?
XXXXXXXX        ← The entered password is not displayed.
```

   Specify the authorization identifier and password for the HADB user you created when executing the adbinit command in 9.2 Initializing a database (creating data DB areas).

2. Execute the CREATE USER statement to create an HADB user.

```
CREATE USER "ADBUSER02" IDENTIFIED BY '#HelloHADB_02'
```

   Executing the above SQL statement creates an HADB user who has ADBUSER02 as an authorization identifier and #HelloHADB_02 as a password.

   For details about the specification rules for authorization identifiers and passwords, see the following subsections:

   - 9.4.2 Authorization identifier specification rules
   - 9.4.3 Password specification rules

3. Execute the GRANT statement to grant the CONNECT privilege and the schema definition privilege.

```
GRANT CONNECT,SCHEMA TO "ADBUSER02"
```

   Executing the above SQL statement grants both the CONNECT privilege and the schema definition privilege to authorization identifier ADBUSER02.

You have now finished creating HADB user ADBUSER02, who can define base tables.

> **Note**
>
> When using the adbsql command to execute an SQL statement, make sure that a semicolon (;) is the last character you enter.

9.4.2 Authorization identifier specification rules

The authorization identifier specification rules are as follows:

- Characters that can be used in authorization identifiers are single-byte uppercase letters, single-byte lowercase letters, single-byte numbers, the backslash (\), the hash mark (#), and the at sign (@).
- When you use lowercase letters for an authorization identifier, enclose the authorization identifier in double quotation marks (").

  Example: `CREATE USER "ADBuser02" ...`

  If lowercase letters are not enclosed in double quotation marks ("), they are treated as uppercase letters. For example, specification `-u ADBuser02` is treated as `-u ADBUSER02`.
- Because authorization identifiers are specified as names, we recommend that you enclose them in double quotation marks (").
- You cannot specify `ALL`, `HADB`, `MASTER`, or `PUBLIC` as an authorization identifier.
- You can specify a maximum of 100 characters (100 bytes) as an authorization identifier.
- For further details about authorization identifier specification rules, see *Specifying names* in *SQL writing conventions* under *SQL Basics* in the manual *HADB SQL Reference*.

■ Rules for specifying an authorization identifier in the -u option of a command

The following rules apply when you are specifying an authorization identifier in the `-u` option of a command.

- When an authorization identifier contains lowercase letters

  If you are specifying an authorization identifier that contains lowercase letters, you must either enclose the authorization identifier with a backslash and double quotation mark (\"), or you must enclose the authorization identifier in double quotation marks (") and then enclose the entire string in single quotation marks (').

  Examples:

  ```
  -u \"ADBuser02\"
  -u '"ADBuser02"'
  ```

  If the authorization identifier is not specified as shown above, all characters are treated as uppercase letters. For example, specification `-u ADBuser02` is treated as `-u ADBUSER02`.
- When an authorization identifier contains a backslash (\)

  The backslash (\) is a special character used by the shell for a different purpose. Therefore, when an authorization identifier contains a backslash (\), specify an escape character before the backslash (\).

  Examples:

  • If you specify `ADBUSER\01` as an authorization identifier: `-u ADBUSER\\01`

  • If you specify `ADBuser\01` as an authorization identifier: `-u \"ADBuser\\01\"`

  In the above examples, the backslash (\) is specified as an escape character.

9.4.3 Password specification rules

The password specification rules are as follows:

- Characters that can be used for passwords are single-byte uppercase letters, single-byte lowercase letters, single-byte numbers, the backslash (\), and the following single-byte characters:

  ```
  @ ` ! " # $ % & ' ( ) * : + ; [ ] { } , = < > | - . ^ ~ / ? _
  ```
- If you are using a JDBC driver, we recommend that you do not use the following single-byte character in a password:

&

- If you are using an ODBC driver, we recommend that you do not use the following single-byte characters in a password:

  [ ] { } ( ) , ; ? * = ! @

- A password is specified in the character string literal format. Therefore, you need to enclose the password in single quotation marks (').

  Example: When specifying Password01 for a password

  IDENTIFIED BY 'Password01'

- When the password character string includes a single quotation mark ('), you need to specify two single quotation marks to indicate one single quotation mark.

  Example: When specifying Pass'01 for a password

  IDENTIFIED BY 'Pass''01'

- A blank character cannot be specified in a password. (The following specification is invalid:)

  Example: IDENTIFIED BY ''

- You can specify a maximum of 255 characters (255 bytes) in a password.

- For details about the password specification format (details about the character string literal specification format), see *Description format of literals* in *Literals* under *SQL Basics* in the manual *HADB SQL Reference*.

■ Rules for specifying a password in the -p option of a command

The following rules apply when you specify a password in the -p option of a command:

- When a password contains a special character used by the shell for a different purpose

  The backslash (\) and vertical bar (|) are examples of special characters used by the shell for different purposes. When a password contains one of these special characters, specify an escape character before the special character.

  Examples:

  • When specifying Password\01 for a password: -p Password\\01

  • When specifying Password|01 for a password: -p Password\|01

  In the above examples, the backslash (\) is specified as an escape character.

9.5 Defining schemas, tables, and indexes

You use definition SQL statements to define schemas, tables, and indexes.

Example:

In this example, the schema of HADB user ADBUSER02 is defined and the following tables and indexes are defined:

- Shop table (SHOPSLIST)
- B-tree index (SHOP_CODE_IDX)
- Text index (ADDRESS_TIDX)
- Range index (RGN_CODE_RIDX)

ADBUSER02 is the authorization identifier of the HADB user created in 9.4.1 Creating an HADB user.

Procedure:

1. Execute the adbsql command.

```
adbsql
  USER-ID ?
ADBUSER02       ← Connects to the HADB server as ADBUSER02.
  PASSWORD ?
XXXXXXXX        ← The entered password is not displayed.
```

2. Execute the CREATE SCHEMA statement to define a schema.

```
CREATE SCHEMA "ADBUSER02"
```

Executing the above SQL statement defines the schema of ADBUSER02.

3. Execute the CREATE TABLE statement to define the shop table (SHOPSLIST).

```
CREATE TABLE "SHOPSLIST"
       ("SHOP_CODE" CHAR(8) NOT NULL,
        "RGN_CODE" CHAR(6) NOT NULL,
        "SHOP_NAME" VARCHAR(20) NOT NULL,
        "TEL_NO" CHAR(10) NOT NULL,
        "ADDRESS" VARCHAR(300) NOT NULL BRANCH YES)
     IN "DBAREA01"
     PCTFREE=40
     CHUNK=100
```

4. Execute the CREATE INDEX statement to define a B-tree index (SHOP_CODE_IDX).

```
CREATE INDEX "SHOP_CODE_IDX"
       ON "SHOPSLIST" ("SHOP_CODE")
     IN "DBAREA01"
     PCTFREE = 50
     EMPTY
```

Executing the above SQL statement defines a B-tree index (SHOP_CODE_IDX) for the shop code column (SHOP_CODE) of the SHOPSLIST table.

5. Execute the CREATE INDEX statement to define a text index (ADDRESS_TIDX).

```
CREATE INDEX "ADDRESS_TIDX"
       ON "SHOPSLIST" ("ADDRESS")
     IN "DBAREA01"
     PCTFREE = 50
```

```
    EMPTY
    INDEXTYPE TEXT
```

Executing this SQL statement defines a text index (ADDRESS_TIDX) for the address column (ADDRESS) of the SHOPSLIST table.

6. Execute the CREATE INDEX statement to define a range index (RGN_CODE_RIDX).

```
CREATE INDEX "RGN_CODE_RIDX"
      ON "SHOPSLIST" ("RGN_CODE")
    IN "DBAREA01"
    EMPTY
    INDEXTYPE RANGE
```

Executing the above SQL statement defines a range index (RGN_CODE_RIDX) for the shop region code column (RGN_CODE) of the SHOPSLIST table.

Note

When you use the adbsql command to execute an SQL statement, make sure that a semicolon (;) is the last character you enter.

9.6 Importing data into a base table

After you have defined a base table, you use the adbimport command to import data into it.

Example:

This example imports data into the shop table (SHOPSLIST).

Procedure:

1. Log on to the OS from the HADB administrator's OS account.

   Log on to the OS from the HADB administrator's OS account so that the HADB administrator can execute the adbimport command.

   This step is not necessary if the HADB administrator is already logged on.

2. Create input data path files.

```
/home/adbmanager/imp_data/imp1.csv
/home/adbmanager/imp_data/imp2.csv
```

   Create input data path files that specify input data file paths. The input data files created here become the input information for the adbimport command.

   In this example, the created input data path files are saved in /home/adbmanager/imp_file/imp_data_path.txt.

3. Create an import option file.

```
set adb_import_errdata_file_name = "/home/adbmanager/imp_shopslist/errdata.csv"
set adb_import_errdata_num = 500
set adb_import_rthd_num = 2
set adb_import_sort_buff_size = 2
               :
```

   Create an import option file that specifies an import option. The import option file created here becomes the input information for the adbimport command.

   In this example, the created import option file is saved in /home/adbmanager/imp_file/imp_opt_file.txt.

   For details about import options, see *Format of import options* in *Specification format for the adbimport command* under *adbimport (Import Data)* in the manual *HADB Command Reference*.

4. Execute the adbimport command to import data into the shop table (SHOPSLIST).

```
adbimport -u ADBUSER02 -p '#HelloHADB_02' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file.txt
          SHOPSLIST
          /home/adbmanager/imp_file/imp_data_path.txt
```

   -u

   Specifies the authorization identifier of the HADB user who executes the adbimport command. Specify the authorization identifier of the HADB user who owns the table into which data is to be imported.

   -p

   Specifies the password of the HADB user who executes the adbimport command.

   -k

   Specifies the enclosing character used in the input data file.

-s

Specifies the delimiting character used in the input data file.

-g

Specifies the interval for outputting the progress message for data importing (KFAA80205-I).

-w

Specifies the directory for storing the temporary work files that are created during data importing.

-z

Specifies the absolute path name of the import option file created in step 3.

SHOPSLIST

Specifies the table into which data is to be imported.

/home/adbmanager/imp_file/imp_data_path.txt

Specifies the absolute path name of the import data path file created in step 2.

9.7 Collecting cost information

After you have imported data into the base table, execute the `adbgetcst` command to collect the cost information of the base table.

Example:

This example collects the cost information of the shop table (`SHOPSLIST`).

Procedure:

1. Log on to the OS from the HADB administrator's OS account.

   Log on to the OS from the HADB administrator's OS account so that the HADB administrator can execute the `adbgetcst` command.

   This step is not necessary if the HADB administrator is already logged on.

2. Create a cost-information collection-options file.

   ```
   set adb_getcst_rthd_num = 10
   ```

   Create a cost-information collection-options file that specifies a value for the cost-information collection option. The cost-information collection-options file created here becomes the input information for the `adbgetcst` command.

   In this example, the created cost-information collection-options file is saved in `/home/adbmanager/cost_file/cost_opt_file.txt`.

   For details about the cost information collection option, see *Format of the cost-information collection option* in *Specification format for the adbgetcst command* under *adbgetcst (Collect Cost Information)* in the manual *HADB Command Reference*.

3. Execute the `adbgetcst` command to collect cost information.

   ```
   adbgetcst -u ADBUSER02 -p '#HelloHADB_02' -t SHOPSLIST
             -z /home/adbmanager/cost_file/cost_opt_file.txt
   ```

   `-u`

   Specifies the authorization identifier of the HADB user who executes the `adbgetcst` command. Specify the authorization identifier of the HADB user who owns the base table from which cost information is to be collected.

   `-p`

   Specifies the password of the HADB user who executes the `adbgetcst` command.

   `-t`

   Specifies the name of the base table from which the cost information is to be collected.

   `-z`

   Specifies the absolute path name of the cost-information collection-options file created in step 2.

> **Note**
>
> For details about the timing with which cost information is collected, see 11.5 Collecting cost information.

10 Scheduled Operations

This chapter explains the operational items that you must perform on a regular basis to maintain the HADB server.

10.1 List of operation items

The following table lists the HADB server's operational items.

Table 10-1: List of HADB server's operation items

No.	Operation items			Reference
1	Scheduled operation	HADB server startup		10.2.1 Starting the HADB server
2		HADB server termination		10.2.2 Terminating the HADB server
3		Changing and confirming the HADB server operation mode		10.2.3 HADB server operation modes
4		Setting up the HADB server to restart automatically		10.2.4 Setting up the HADB server to restart automatically
5		Database backup		10.3 Backing up a database
6		Monitoring	Checking the messages	10.4 Checking the messages
7			Monitoring the resource usage	10.5 Monitoring the resource usage (list of messages to be monitored)
8			Checking the memory usage	10.6 Checking the memory usage
9			Checking the threads running on the HADB server	10.7 Checking the threads running on the HADB server
10			Checking the transaction processing status	10.8 Checking the transaction processing status
11			Checking the database status and usage	10.9 Checking the database status and usage
12			Checking the HADB server operation information	10.10 Performing statistical analysis (checking HADB server operation information)
13			Acquiring and checking SQL statement execution log information (SQL trace information)	10.11 Running SQL tracing
14			Working with the system log files	10.12 Working with the system log files
15	Unscheduled operation	Base table operations		11.1 Base table operations
16		Viewed table definition, retrieval, and deletion		11.2 Viewed table operations
17		Index operation		11.3 Index operations
18		Performing operations on regular multi-chunk tables and archivable multi-chunk tables		11.4 Performing operations on multi-chunk tables
19		Collecting cost information		11.5 Collecting cost information
20		HADB user management		11.6 Managing HADB users
21		Managing user privileges and the schema operation privilege		11.7 Managing user privileges and the schema operation privilege
22		Managing access privileges		11.8 Managing access privileges
23		Handling schemata		11.9 Handling schemas

No.	Operation items		Reference
24		Handling data DB areas	11.10 Handling data DB areas
25		Handling the work table DB area	11.11 Handling the work table DB area
26		Using buffers	11.12 Performing operations on buffers
27		Checking client group settings	11.13 Checking client group settings
28		Operating centralized management of client definitions	11.14 Operating centralized management of client definitions
29		Retrieving data from CSV files	11.15 Handling of data retrieval from CSV files
30		Performing synonym search operations	11.16 Performing synonym search operations
31		Reorganizing system tables	11.17 Reorganizing system tables
32		Using the updated-row columnizing facility	11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables)
33		Operating the audit trail facility	12. Audit Trail Facility Operations
34	Tuning	Improving processing performance	13.1 Tuning to improve processing performance
35		Tuning to shorten SQL statement execution time by re-examining the buffers	13.2 Tuning to shorten SQL statement execution time by re-examining the buffers
36		Shortening command execution time	13.3 Tuning to shorten command execution time
37		Reducing memory usage	13.4 Tuning to reduce memory usage
38		Tuning to shorten SQL statement execution time by re-examining the hash table area size	13.5 Tuning to shorten SQL statement execution time by re-examining the hash table area size
39		Tuning to shorten SQL statement execution time by re-examining the hash group area size	13.6 Tuning to shorten SQL statement execution time by re-examining the hash group area size
40		Tuning to shorten SQL statement execution time by re-examining the join order of `INNER JOIN` to which hash join is applied	13.7 Tuning to shorten SQL statement execution time by re-examining the join order of INNER JOINs to which a hash join is applied
41		Tuning to shorten SQL statement execution time by re-examining the size of the hash filter area	13.8 Tuning to shorten SQL statement execution time by re-examining the hash filter area size
42		Tuning to shorten SQL statement execution time by re-examining the table-definition pool size	13.9 Tuning to shorten SQL statement execution time by re-examining the table-definition pool size

No.	Operation items		Reference
43	Operation when an error occurs	Checking whether abnormal termination has occurred	14.1 Error-handling flow
44		Collecting troubleshooting information	14.2 Collecting troubleshooting information (adbinfoget command)
45		Deleting the collected troubleshooting information	14.3 Deleting troubleshooting information (adbinfosweep command)
46		Troubleshooting	15. Troubleshooting

10.2 Starting and terminating the HADB server and its operation modes

This section explains how to start and terminate the HADB server and its operation modes.

10.2.1 Starting the HADB server

To start the HADB server, execute the `adbstart` command. The HADB server starts.

Enter the command as follows:

```
adbstart
```

Important

Multiple operation modes are available for the HADB server. For details about the HADB server operation modes, see 10.2.3 HADB server operation modes.

By specifying an option when you execute the `adbstart` command, you can change the operation mode after the HADB server has started. If no option is specified, the HADB server inherits the operation mode that was in effect when the HADB server terminated previously. For details about how to specify options for the `adbstart` command, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

After you have started the HADB server, set it up to restart automatically as needed. For details, see 10.2.4 Setting up the HADB server to restart automatically.

(1) HADB server startup modes

The HADB server can be started in either of the modes described below. How it operates differs depending on the startup mode.

Table 10-2: HADB server startup mode

No.	Startup mode	Command to be executed	Explanation	Previous termination mode
1	Normal start	`adbstart`	This is the normal startup mode. In a normal start, database recovery processing does not take place.	Normal termination
2	Restart		If the previous termination mode was either of the two shown to the right, the HADB server is automatically restarted. During the restart, system logs are used to perform database recovery processing.	• Forced termination • Abnormal termination

(2) Checking whether the startup process has been completed

To check whether the HADB server has completed startup, check the return code of the `adbstart` command.

- **If the return code is `0`**
  Startup processing ended normally. The HADB server can now accept database access requests from applications.

- **If the return code is 4**

  Although startup processing has ended, a warning message was output.

  The HADB server has started. Check the displayed message or the message output to the message log file, and take the necessary action.

- **If the return code is 8**

  An error occurred during HADB server startup processing and startup processing ended abnormally. Check the displayed message or the message output to the message log file, and take the necessary action.

10.2.2 Terminating the HADB server

To stop the HADB server, execute the `adbstop` command. This stops the HADB server.

Enter the command as follows:

```
adbstop#
```

#

By specifying an option when executing the `adbstop` command, you can change the HADB server termination mode. For details, see (1) HADB server termination modes and (2) Option specification and normal termination types.

> **Important**
>
> If there are applications or commands connected to the HADB server, executing the `adbstop` command will not terminate the HADB server.

(1) HADB server termination modes

The HADB server can be terminated in any of the modes described below. The options you specify in the `adbstop` command differ depending on the termination mode.

Table 10-3: HADB server termination mode

No.	Termination mode	Command and option to be executed	Explanation
1	Normal termination	`adbstop`	This is the normal termination mode. There are four types of normal termination, and the termination process differs depending on the option specified. For details, see (2) Option specification and normal termination types.
2	Forced termination	`adbstop --force`	The HADB server terminates without waiting for any transactions currently being processed to be completed. Once the `adbstop --force` command is executed, no new connection requests are accepted. The transactions being processed will be the target of recovery processing when the HADB server is restarted.
3	Abnormal termination	--	This is the mode in which the HADB server is terminated by some error. The HADB server terminates without waiting for any transactions being processed to be completed. The transactions being processed will be the target of recovery processing when the HADB server is restarted.

Legend:

--: There is no command to be executed.

(2) Option specification and normal termination types

The normal termination process differs depending on the option specified for the `adbstop` command. The table below shows the options that can be specified for the `adbstop` command and the types of normal termination.

For details about how to specify options for the `adbstop` command, see *adbstop (Terminate the HADB Server)* in the manual *HADB Command Reference*.

Table 10-4: Options to be specified for the adbstop command and the normal termination types

No.	Command and option to be executed	Explanation
1	`adbstop`	When the `adbstop` command (with no option specified) is executed, the HADB server terminates if no application program is connected to the HADB server or if no command is being executed. **Situation in which the adbstop command is executed** Execute this command if you want to terminate the HADB server normally when no application program is connected to the HADB server and no command is being executed. **Processing that occurs when the adbstop command is executed** When the `adbstop` command is executed and the HADB server termination process starts, no new connection requests are accepted. If there is an application program that is connected to the HADB server or a command that is being executed, an error occurs and the `KFAA91152-E` message is output (the HADB server cannot be terminated). In such a case, see the manual *HADB Messages* and take the step described in the action column of the `KFAA91152-E` message.
2	`adbstop --wait connection`	When the `adbstop --wait connection` command is executed, the HADB server terminates after waiting for the following processes to be completed: • Disconnection of any connected application programs from the HADB server • Termination of the command being executed **When to execute the adbstop --wait connection command** Execute this command in the following situations: • If you want to normally terminate the HADB server as soon as the command being executed (such as the `adbimport` command) terminates • If you want to normally terminate the HADB server immediately after processing by any connected application programs finishes and the application programs are disconnected from the HADB server **Processing that occurs when the adbstop --wait connection command is executed** After the `adbstop --wait connection` command is executed, the HADB server enters termination standby processing and waits for the following processes (the wait time in termination standby processing can be specified using the `-t` option): • Disconnection of connected application programs from the HADB server • Termination of commands being executed During this time, the HADB server accepts no new connection requests (it still accepts new transaction start requests from an application program connected to the HADB server or a command being executed). When all application programs connected to the HADB server have been disconnected and all commands being executed terminate, the HADB server termination process starts.

No.	Command and option to be executed	Explanation
3	adbstop --wait transaction	When the adbstop --wait transaction command is executed, the HADB server terminates after waiting for the transactions being processed to finish and the commands being executed to terminate. **When to execute the adbstop --wait transaction command** Execute this command in the following situations: • If you want to normally terminate the HADB server as soon as the transactions being processed finish • If you want to normally terminate the HADB server as soon as the command being executed (such as the adbimport command) terminates • If you want to normally terminate the HADB server when an application program is connected but no transaction has started **Processing that occurs when the adbstop --wait transaction command is executed** After the adbstop --wait transaction command is executed, the HADB server enters termination standby processing and waits for the transactions being processed to finish and the commands being executed to terminate (the wait time in termination standby processing can be specified using the -t option). During this time, the HADB server accepts no new connection requests or new transaction start requests. When all transactions being processed finish and all commands being executed terminate, all application programs and commands connected to the HADB server are disconnected and the HADB server termination process starts.
4	adbstop --cancel	When the adbstop --cancel command is executed, the HADB server terminates, canceling the transactions being processed and the commands being executed. **When to execute the adbstop --cancel command** Execute this command when you want to normally terminate the HADB server immediately by canceling processing. **Processing that occurs when the adbstop --cancel command is executed** After the adbstop --cancel command is executed, the HADB server cancels the transactions being processed and the commands being executed. During this time, the HADB server accepts no new connection requests or new transaction start requests. When all transactions being processed and all commands being executed are canceled, all application programs and commands connected to the HADB server are forcibly disconnected, and the HADB server termination process starts.

■ Termination standby processing

Termination standby processing is a status in which the HADB server is waiting for the following processes:

- Disconnection of connected application programs from the HADB server
- Termination of commands being executed
- Completion of transactions being processed

The HADB server enters termination standby processing only when the adbstop --wait connection command and the adbstop --wait transaction command are executed. Termination standby processing has a wait time, which can be specified using the -t option of the adbstop command. If the -t option is omitted, the default wait time is applied.

■ Wait time in termination standby processing

A timeout occurs in any of the following cases: if an application program connected to the HADB server is not disconnected, if a command being executed does not terminate, or if a transaction being processed does not finish within the wait time in termination standby processing. When a timeout occurs, the HADB server does not terminate. In this case, eliminate the cause of the error based on the information in the KFAA91152-E message that was output.

When the wait time in termination standby processing is exceeded (a timeout occurs), the HADB server can once again accept new connection requests and new transaction start requests.

■ Checking for termination standby processing

You can use the `adbls -d srv` command to check whether the HADB server is in termination standby processing. If the output item `STATUS` shows `STOPWAIT`, the HADB server is in termination standby processing.

If you want to cancel termination standby processing, use the operating system's `kill` command to terminate the `adbstop` command process. This cancels termination standby processing.

When termination standby processing is canceled, the HADB server can once again accept new connection requests and new transaction start requests.

For details about the `adbls -d srv` command, see *adbls -d srv (Display the HADB Server Status)* in the manual *HADB Command Reference*.

(3) Checking whether the termination process has been completed

To check whether the HADB server has completed terminating, check the return code of the `adbstop` command.

- **If the return code is `0`**

  Termination processing ended normally.

- **If the return code is `4`**

  Although termination processing has ended, a warning message was output.

  The HADB server has terminated. Check the displayed message or the message output to the message log file, and take the necessary action.

- **If the return code is `8`**

  An error occurred during termination processing of the HADB server and termination processing ended abnormally. Check the displayed message or the message output to the message log file, and take the necessary action.

10.2.3 HADB server operation modes

The following operation modes are available for the HADB server:

- Normal mode
- Quiescence mode
- Offline mode
- Maintenance mode

The following table describes these operation modes in detail.

Table 10-5: Details of the HADB server's operation modes

No.	Operation mode	Explanation
1	Normal mode	This is the normal operation mode. In the normal mode, the HADB server and HADB clients can execute SQL statements and commands as usual. If you execute the `adbstart` command with its option omitted when you start the HADB server for the first time, the HADB server starts in the normal mode.
2	Quiescence mode	This operation mode prevents database updating. For example, use this mode to back up a database while keeping the HADB server running.

No.	Operation mode	Explanation
		In the quiescence mode, the HADB server and HADB clients can execute SQL statements that search the database. However, SQL statements that update the database cannot be executed. Some commands cannot be executed either. For details about the SQL statements and commands that can be executed in the quiescence mode, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.
3	Offline mode	This operation mode rejects connection requests from HADB clients installed on server machines other than the server machine on which the HADB server is installed. Use this mode, for example, when you want to execute commands taking advantage of the maximum processing performance of the HADB server. In the offline mode, HADB clients installed on the same server machine as the HADB server can execute SQL statements and commands as usual. However, HADB clients installed on server machines other than the server machine on which the HADB server is installed cannot execute SQL statements and commands. For details about the SQL statements and commands that can be executed in the offline mode, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.
4	Maintenance mode	In this operation mode, the HADB server rejects connection requests from applications. You can execute only the specified commands (such as the `adbmodarea` command) in the maintenance mode. For details about the commands that can be executed in the maintenance mode, see *SQL statements and commands that can be executed in each HADB server operation mode* in *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*. Use the maintenance mode in the following cases: • When you execute a command that cannot be executed concurrently with other commands and applications. • When you return a node to the multi-node configuration. For details about returning a node to the multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

(1) Changing the operation mode

You can change the HADB server's operation mode while you are starting the server or while it is running. The methods of changing the operation mode are as follows:

■ Changing the operation mode to the normal mode

- To change the mode while starting the HADB server
  Execute the `adbstart --normal` command when you start the HADB server.
- To change the mode while the HADB server is running
  Execute the `adbchgsrvmode --normal` command while the HADB server is running in the quiescence, offline, or maintenance mode.

■ Changing the operation mode to the quiescence mode

- To change the mode while starting the HADB server
  Execute the `adbstart --quiescence` command when you start the HADB server.
- To change the mode while the HADB server is running
  Execute the `adbchgsrvmode --quiescence` command while the HADB server is running in the normal mode. You can change the operation mode from the normal mode to the quiescence mode.

■ Changing the operation mode to the offline mode

- To change the mode while starting the HADB server
  Execute the `adbstart --offline` command when you start the HADB server.
- To change the mode while the HADB server is running

Execute the `adbchgsrvmode --offline` command while the HADB server is running in the normal mode. You can change the operation mode from the normal mode to the offline mode.

▪ **Changing the operation mode to the maintenance mode**

- To change the mode while starting the HADB server
  Execute the `adbstart --maintenance` command when you start the HADB server.
- To change the mode while the HADB server is running
  Execute the `adbchgsrvmode --maintenance` command while the HADB server is running in the normal mode. You can change the operation mode from the normal mode to the maintenance mode.

(2) Checking the operation mode

You can use the following command to check the HADB server's operation mode:

```
adbls -d srv
```

Executing this command displays the HADB server's status. Check the item `STATUS` in the execution result.

- **When `STATUS` is `ACTIVE`**
  The operation mode is the normal mode.
- **When `STATUS` is `QUIESCE`**
  The operation mode is the quiescence mode.
- **When `STATUS` is `OFFLINE`**
  The operation mode is the offline mode.
- **When `STATUS` is `MAINTNCE`**
  The operation mode is the maintenance mode.

10.2.4 Setting up the HADB server to restart automatically

When the HADB server terminates abnormally, it does not normally restart automatically. However, if the `adbmonitor` command has been executed, the HADB server restarts automatically after terminating abnormally.

The procedure for setting the HADB server to restart automatically is described below.

Note that you can set the HADB server to restart automatically only if the multi-node function is not being used.

Procedure

1. Start the HADB server.
   Execute the `adbstart` command.
2. Set the HADB server to restart automatically.
   Execute the `adbmonitor -r` command.

When you execute the `adbmonitor -r` command, the `adbmonitor` command monitors the status of the HADB server. Then, if the HADB server terminates abnormally, it restarts from the command process of the `adbmonitor` command.

Important

Depending on the cause of abnormal termination of the HADB server, it might not be able to restart automatically (it might terminate abnormally due to the same cause). In this case, use **Ctrl**+**C**, for example, to forcibly terminate the command process of the `adbmonitor` command.

You can check whether the HADB server terminated abnormally based on whether the message `KFAA60003-E` was output to the message log file. Even if the HADB server terminated abnormally and then restarted normally via the automatic restart process, use the `adbinfoget` command to collect troubleshooting information. Then contact the customer support center.

If you terminate the HADB server normally by executing the `adbstop` command, the command process of the `adbmonitor` command, which was monitoring the HADB server, also terminates. Therefore, the next time you start the HADB server, execute the `adbmonitor -r` command after you have started the server if you want to set up automatic restart.

10.3 Backing up a database

This section explains how to back up a database and how to restore the database from the backup.

10.3.1 Backup acquisition method

This section explains the timing of backup acquisition, the types of backup, and the procedure for making a backup.

For details about the backup acquisition method for when the multi-node function is being used, see 16.8 Backing up a database (when the multi-node function is being used).

(1) Backup acquisition timing

As a rule, make a backup on a regular basis so that the backup can be used for restoring the database in the event of an error. It is especially important to back up your database before you perform any of the following operations:

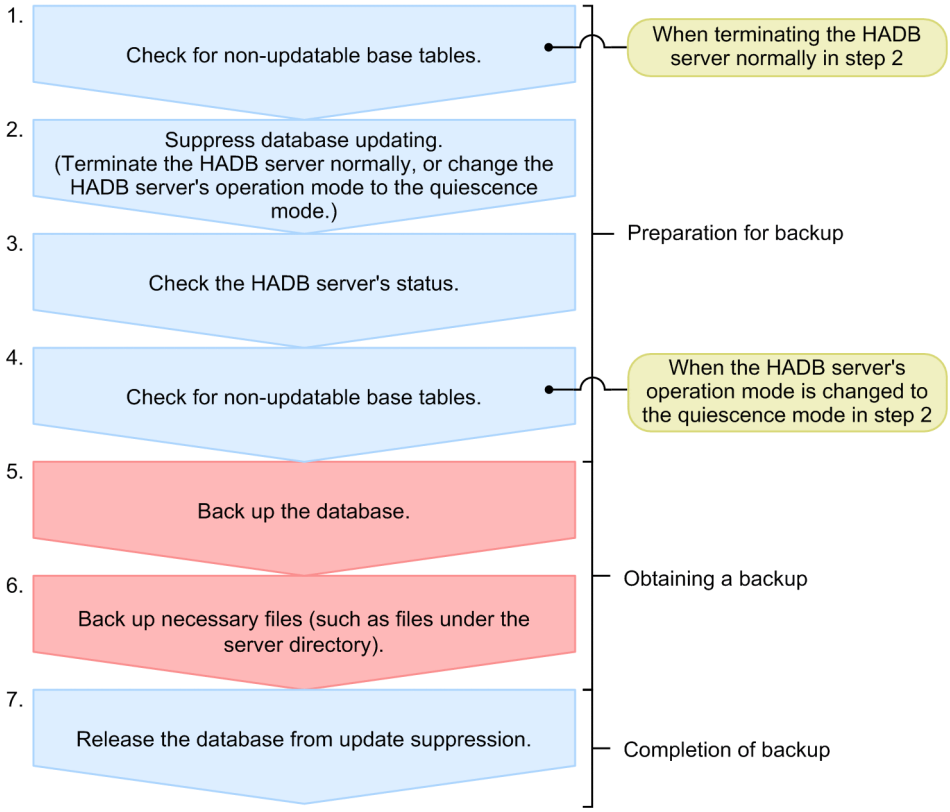
- Executing the `adbmodarea` command
- Upgrading the HADB server version

> **Important**
>
> When you back up a database, back up all necessary files including the data DB area file. This backup method is called full backup.

(2) Backup procedure

The following figure shows the procedure for backing up a database.

Figure 10-1: Backup procedure



Each of the steps in the above figure is explained in detail below. Note that the numbers in the figure correspond to the step numbers shown below.

If you plan to use the *Change the HADB server's operation mode to the quiescence mode* in *Suppress database updating* in step 2, skip step 1 and start with step 2.

1. Check for non-updatable base tables.

   Before backing up a database, you need to confirm that there is no non-updatable base table. Execute the `adbdbstatus` command with the `-c table` option specified. If you plan to change the HADB server's operation mode to the quiescence mode in step 2, this action in step 1 is not required. Proceed to step 2.

   The following specification example checks the status of all base tables:

   ```
   adbdbstatus -c table
   ```

   Confirm that `non-updatable` is not output in the `Non-updatable` column of the output result. If `non-updatable` is output, the base table is non-updatable. Release the non-updatable state of the base table based on the explanation in 15.8.1 Steps to take when a base table becomes non-updatable.

   Important

   If you make a backup while a base table is non-updatable and you then restore the database using that backup, you might not be able to release the base table from non-updatable status. If a base table is non-updatable, you cannot execute retrieval or updating.

2. Suppress database updating.

   Before backing up a database, you need to prevent the database from being updated. To do so, use one of the following approaches:

   - **Terminate the HADB server normally.**

If terminating the HADB server will not cause any problems, execute the `adbstop` command to terminate the HADB server normally.

- **Change the HADB server's operation mode to the quiescence mode.**
  If the HADB server is running and cannot be terminated, execute the `adbchgsrvmode --quiescence` command to change the operation mode to the quiescence mode.

> **Important**
>
> If you change the operation mode to quiescence mode and make a backup, and if you then start the HADB server after the database is recovered, the HADB server restarts. In this case, if the server definition was changed after the backup acquisition, the HADB server might not be able to restart after it is restored from the backup that was made. Therefore, when making a backup after changing the operation mode to the quiescence mode, we recommend that you also back up the server definition file as described in step 6. If the HADB server cannot restart because a server definition was changed, take the necessary corrective actions based on messages `KFAA50024-E` and `KFAA50025-E`.

3. Check the HADB server's status.
   When you have completed step 2, check the HADB server's status. Execute the following command to confirm that database updating has been suppressed.

```
adbls -d srv
```

   Check the item `STATUS` in the execution result.

   - **If the HADB server terminated normally**
     Confirm that the output item `STATUS` shows `STOP`. Then proceed to step 5.
   - **If you changed the HADB server's operation mode to the quiescence mode**
     Confirm that the output item `STATUS` shows `QUIESCE`. Then proceed to step 4.

4. Check for non-updatable base tables.
   Before backing up a database, you need to confirm that there is no non-updatable base table. Execute the `adbdbstatus` command with the `-c table` option specified.
   If the HADB server terminated normally in step 2, proceed to step 5.
   The following specification example checks the status of all base tables:

```
adbdbstatus -c table
```

   Confirm that `non-updatable` is not output in the `Non-updatable` column of the output result. If `non-updatable` is output, the base table is non-updatable. Release the non-updatable state of the base table based on the explanation in 15.8.1 Steps to take when a base table becomes non-updatable.

5. Back up the database.
   Once you have suppressed database updating, you can make a backup. Back up all files that need to be backed up. If you back up only some of the files, database compatibility cannot be achieved when the database is restored from the backup, and the validity of subsequent operations cannot be guaranteed.
   If the target file is a symbolic link file, also back up the linked file and block special file. Do not use the backed-up files as sparse files.
   The following table shows the files that need to be backed up.

Table 10-6: List of files that needed to be backed up

No.	File that needs to be backed up	File storage location	File type
1	Command status file	$DBDIR/ADBSYS/ADBUTL	Regular file
2	System log file	$DBDIR/ADBSYS/ADBSLG	Regular file
3	Status file	$DBDIR/ADBSYS/ADBSTS	Regular file
4	Master directory DB area file	$DBDIR/ADBMST	Regular file or block special file
5	Dictionary DB area file	$DBDIR/ADBDIC	Regular file or block special file
6	System-table DB area file	$DBDIR/ADBSTBL	Regular file or block special file
7	All files comprising the data DB area	$DBDIR/DBAREA[#1]	Regular file or block special file
8	Archive file	Archive directory[#2]	Regular file
9	Synonym dictionary file	Directory for storing synonym dictionary files[#3]	Regular file

#1

This is the name specified by the user in the adbinit or adbmodarea command.

#2

This is the directory specified for ARCHIVEDIR in the chunk-archive specification of the CREATE TABLE statement.

#3

This is the directory specified in the adb_syndict_storage_path operand in the server definition.

You can re-create a synonym dictionary file from a synonym list definition file. Therefore, if a synonym list definition file is saved, you do not have to perform full backup for a synonym dictionary file. Note, however, that in this case, a database and a synonym dictionary file are not synchronized with each other when the database is restored. As a result, if a synonym dictionary is updated after full backup, the following search results might be different from each other:

- Search result for synonyms immediately after full backup
- Search result for synonyms when a database is restored to the state when full backup was performed

Note

The files under the following directories do not need to be backed up:

- Work directory ($DBDIR/ADBWORK)
- Output destination directory for error information (core files) ($DBDIR/SPOOL)

  This is the directory specified in the adb_core_path operand if the adb_core_path operand in the server definition is specified.

Note that there is no problem if you back up these files along with the files listed in the table above, and recover them from a backup.

> **Note**
>
> There is no need to back up the audit trail files in the audit trail directory (the directory specified for the `adb_audit_log_path` operand in the server definition). Instead of backing up the audit trail files in the audit trail directory, you can move them to the audit trail storage directory. For details about this process, see 12.3.1 Moving audit trail files (to audit trail storage directory).

6. Back up necessary files (such as files under the server directory).

   If a problem such as a disk error occurs, you will need to recover the server directory or client directory. Therefore, back up the files listed below as well. These files are not directly used in database recovery.

   For the HADB server

   - All definition files stored under the server directory (`$ADBDIR/conf`)
   - All user-created files under the server directory (`$ADBDIR`)

   For an HADB client

   - All definition files stored under the client directory (`%ADBCLTDIR%\conf` for Windows and `$ADBCLTDIR/conf` for Linux)
   - All user-created files under the client directory (`%ADBCLTDIR%` for Windows and `$ADBCLTDIR` for Linux)

7. Release the database from update suppression.

   Once you have completed the backup acquisition operation, release the database from update suppression.

   - **If the HADB server terminated normally in step 2**

     Execute the `adbstart` command to start the HADB server.
   - **If you changed the HADB server's operation mode to the quiescence mode in step 2**

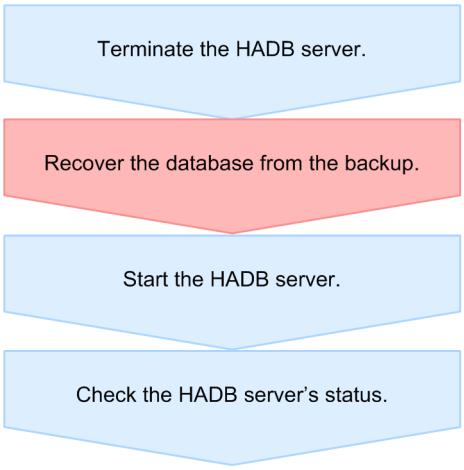
     Execute the `adbchgsrvmode --normal` command to change the operation mode to the normal mode.

   You have now completed backup acquisition.

10.3.2 Recovering the database from the backup

This section explains how to recover the database from a backup that was made.

Figure 10-2: Procedure for recovering the database from a backup



1. Terminate the HADB server.

Before you can recover the database from a backup, you need to terminate the HADB server. Execute the following command to terminate the HADB server:

```
adbstop
```

Important

- Before you recover the database from a backup, you must terminate the HADB server. If the database is recovered from a backup while the HADB server is running, database compatibility can no longer be maintained, and the validity of subsequent operations of the HADB server cannot be guaranteed.
- Even when the HADB server's termination mode is abnormal termination or forced termination, it can be recovered from a backup as long as the HADB server terminated.

2. Recover the database from the backup.

   Recover the database from the backup that was made. Store under the target directory all files acquired in step *5. Back up the database* in (2) Backup procedure under 10.3.1 Backup acquisition method.

   If the files to be restored include symbolic link files, you must also recover the target files and block special files. Do not use sparse files as files to be restored.

   Important

   - When you recover the database from a backup, be sure to re-store all files that were backed up. If you re-store only some of the files, database compatibility cannot be achieved and the validity of subsequent operations cannot be guaranteed.

     When recovering the database from a backup, store only the files that were backed up. If the files under the target directory are overwritten with files that were not backed up, database compatibility cannot be achieved and the validity of subsequent operations cannot be guaranteed.
   - If a block special file is specified for the work table DB area, first restore the DB area. Next, specify the `adb_blk_path_wrk` operand in the server definition, and then start the HADB server. For details about the `adb_blk_path_wrk` operand in the server definition, see the description of the adb_blk_path_wrk operand in 7.2.1 Operands related to system configuration (set format).

3. Start the HADB server.

   Once you have recovered the database from the backup that was made, start the HADB server. Execute the following command to start the HADB server:

   ```
   adbstart
   ```

4. Check the HADB server's status.

   After you have started the HADB server, check whether the HADB server's operation mode is the normal mode. Execute the following command to check the HADB server's status:

   ```
   adbls -d srv
   ```

   Check the item `STATUS` in the execution result. If `STATUS` is `ACTIVE`, the operation mode is set to the normal mode. If `STATUS` is `QUIESCE`, the operation mode is set to the quiescence mode. Execute the `adbchgsrvmode --normal` command to change the operation mode to the normal mode.

To recover the server directory or client directory after a problem such as a disk error has occurred, see 15.4.2 Recovering the server directory or 15.4.3 Recovering the client directory.

10.3.3 Backup operation example (using OS commands)

This subsection provides an example of a backup operation using OS commands.

(1) Directory configuration example

In this example, you will back up the following directories:

- DB directory (`/HADB/db`)
- Archive directory (`/HADB/archive`)
- Directory for storing synonym dictionary files (`/HADB/syndict`)

The following table lists and describes the configuration example of the preceding directories.

Table 10-7: Configuration example of directories to be backed up

No.	Names of directories and files to be backed up	Description	File type	To be backed up?
1	`/HADB/db/ADBDIC`	Dictionary DB area file	Regular file	Y
2	`/HADB/db/ADBMST`	Master directory DB area file	Regular file	Y
3	`/HADB/db/ADBSTBL`	System-table DB area file	Regular file	Y
4	`/HADB/db/ADBSYS`	System directory	Regular file group	Y
5	`/HADB/db/ADBWORK`	Work directory	Regular file	N
6	`/HADB/db/ADBWRK`	Work table DB area file	Regular file	N
7	`/HADB/db/ADBUTBL01`[#1]	Data DB area file	Block special file	Y
8	`/HADB/db/ADBUIDX01`[#1]	Data DB area file	Block special file	Y
9	`/HADB/db/SPOOL`	Error information (core file) output destination file	Regular file group	N
10	`/HADB/archive`	Archive directory[#2]	Regular file	Y
11	`/HADB/syndict`	Directory for storing synonym dictionary files[#3]	Regular file	Y
12	`/HADB/audit`	Audit trail directory	Regular file	N

Legend:

Y: A backup is acquired.

N: A backup is not acquired.

#1

Assumed to consist of eight block special files.

#2

If an archivable multi-chunk table is defined, you need to back up the archive directory.

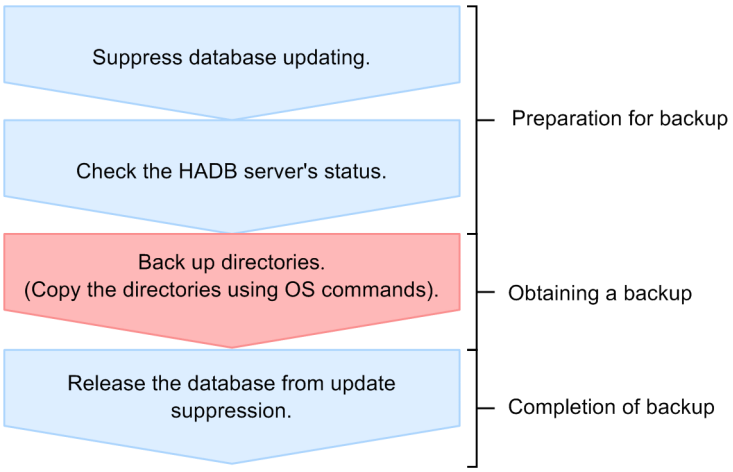
#3

If synonym search is performed, you need to back up the directory for storing synonym directory files.

(2) Using OS commands to make a backup

The following figure shows an example of using OS commands to back up a database.

Figure 10-3: Example of using OS commands to make a backup



In this example, a backup is acquired in line with the following assumptions:

Conditions:

- Following database initialization, the HADB server was started at least once.
- It has been confirmed that there are no non-updatable base tables.
- To suppress database updating, the HADB server is terminated normally.
- A full backup is used.
- The DB directory, archive directory, and the directory for storing synonym dictionary files are to be backed up.

Procedure:

1. Suppress database updating.
   Before backing up the database, suppress database updating. Here, terminate the HADB server normally.

   ```
   adbstop
   ```

2. Check the HADB server's status.
   Confirm that the HADB server terminated normally in step 1.

   ```
   adbls -d srv
   ```

   Confirm that the item `STATUS` in the execution result shows `STOP`.

3. Back up directories (by copying them using OS commands).
   Use OS commands to copy all files that need to be backed up, as described in (1) Directory configuration example.
   - Copying destination for the files under the DB directory
     `/HADB_bkup/db` directory
   - Copying destination for the files under the archive directory
     `/HADB_bkup/archive` directory
   - Copying destination for the files under the directory storing synonym dictionary files
     `/HADB_bkup/syndict` directory

The OS commands that you need to execute are listed below. We recommend that you use the cp command to back up regular files and the dd command to back up block special files.

```
cp -r /HADB/db/ADBSYS  /HADB_bkup/db/ADBSYS

cp -H /HADB/db/ADBMST  /HADB_bkup/db/ADBMST

cp -H /HADB/db/ADBDIC  /HADB_bkup/db/ADBDIC

cp -H /HADB/db/ADBSTBL /HADB_bkup/db/ADBSTBL

dd if=/HADB/db/ADBUTBL01
   of=/HADB_bkup/db/ADBUTBL01 bs=524288

dd if=/HADB/db/ADBUTBL01.00001
   of=/HADB_bkup/db/ADBUTBL01.00001 bs=524288

dd if=/HADB/db/ADBUTBL01.00002
   of=/HADB_bkup/db/ADBUTBL01.00002 bs=524288

dd if=/HADB/db/ADBUTBL01.00003
   of=/HADB_bkup/db/ADBUTBL01.00003 bs=524288

dd if=/HADB/db/ADBUTBL01.00004
   of=/HADB_bkup/db/ADBUTBL01.00004 bs=524288

dd if=/HADB/db/ADBUTBL01.00005
   of=/HADB_bkup/db/ADBUTBL01.00005 bs=524288

dd if=/HADB/db/ADBUTBL01.00006
   of=/HADB_bkup/db/ADBUTBL01.00006 bs=524288

dd if=/HADB/db/ADBUTBL01.00007
   of=/HADB_bkup/db/ADBUTBL01.00007 bs=524288

dd if=/HADB/db/ADBUIDX01
   of=/HADB_bkup/db/ADBUIDX01 bs=524288

dd if=/HADB/db/ADBUIDX01.00001
   of=/HADB_bkup/db/ADBUIDX01.00001 bs=524288

dd if=/HADB/db/ADBUIDX01.00002
   of=/HADB_bkup/db/ADBUIDX01.00002 bs=524288

dd if=/HADB/db/ADBUIDX01.00003
   of=/HADB_bkup/db/ADBUIDX01.00003 bs=524288

dd if=/HADB/db/ADBUIDX01.00004
   of=/HADB_bkup/db/ADBUIDX01.00004 bs=524288

dd if=/HADB/db/ADBUIDX01.00005
   of=/HADB_bkup/db/ADBUIDX01.00005 bs=524288

dd if=/HADB/db/ADBUIDX01.00006
   of=/HADB_bkup/db/ADBUIDX01.00006 bs=524288

dd if=/HADB/db/ADBUIDX01.00007
   of=/HADB_bkup/db/ADBUIDX01.00007 bs=524288

cp -r /HADB/archive  /HADB_bkup/archive

cp -r /HADB/syndict  /HADB_bkup/syndict
```

Important

The data size in block special files will become larger than its actual size because entire volumes are copied.

4. Release the database from update suppression.

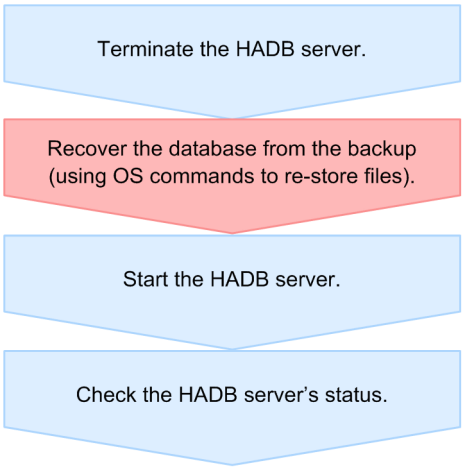
   Once you have used OS commands to make a backup, you can release the database from update suppression. Here, execute the following command to start the HADB server.

```
adbstart
```

(3) Using OS commands to recover a database from a backup

The following figure shows an example of using OS commands to recover a database from a backup.

Figure 10-4: Example of using OS commands to recover a database from a backup



Procedure:

1. Terminate the HADB server.

   Before recovering a database from a backup, you must terminate the HADB server. Use the following command to terminate the HADB server.

```
adbstop
```

2. Recover the database from the backup (using OS commands to re-store files).

   Recover the database from the backup that was made.

   First, to delete the system logs before recovering the database from the backup, delete the system directory (`/HADB/db/ADBSYS`) located under the DB directory. Execute the following OS command:

```
rm -r /HADB/db/ADBSYS
```

   Next, to delete the archive files that were created before recovering the database from the backup, delete the archive files under the archive directory. Execute the following OS command:

```
rm -r /HADB/archive/*
```

   Next, delete the synonym dictionary files that were created before recovering the database from the backup. Execute the following OS command:

```
rm -r /HADB/syndict/*
```

Then, store again all files that were acquired in step *3. Back up directories (by copying them using OS commands)* in (2) Using OS commands to make a backup.

- Storing destination for the files under the DB directory for which a backup was made

  `/HADB/db` directory
- Storing destination for the files under the archive directory for which a backup was made

  `/HADB/archive` directory
- Storing destination for the files under the directory for storing synonym dictionary files for which a backup was made

  `/HADB/syndict` directory

Execute the following OS commands:

```
cp -r /HADB_bkup/db/ADBSYS  /HADB/db/ADBSYS

cp -H /HADB_bkup/db/ADBMST  /HADB/db/ADBMST

cp -H /HADB_bkup/db/ADBDIC  /HADB/db/ADBDIC

cp -H /HADB_bkup/db/ADBSTBL /HADB/db/ADBSTBL

dd if=/HADB_bkup/db/ADBUTBL01
   of=/HADB/db/ADBUTBL01 bs=524288

dd if=/HADB_bkup/db/ADBUTBL01.00001
   of=/HADB/db/ADBUTBL01.00001 bs=524288

dd if=/HADB_bkup/db/ADBUTBL01.00002
   of=/HADB/db/ADBUTBL01.00002 bs=524288

dd if=/HADB_bkup/db/ADBUTBL01.00003
   of=/HADB/db/ADBUTBL01.00003 bs=524288

dd if=/HADB_bkup/db/ADBUTBL01.00004
   of=/HADB/db/ADBUTBL01.00004 bs=524288

dd if=/HADB_bkup/db/ADBUTBL01.00005
   of=/HADB/db/ADBUTBL01.00005 bs=524288

dd if=/HADB_bkup/db/ADBUTBL01.00006
   of=/HADB/db/ADBUTBL01.00006 bs=524288

dd if=/HADB_bkup/db/ADBUTBL01.00007
   of=/HADB/db/ADBUTBL01.00007 bs=524288

dd if=/HADB_bkup/db/ADBUIDX01
   of=/HADB/db/ADBUIDX01 bs=524288

dd if=/HADB_bkup/db/ADBUIDX01.00001
   of=/HADB/db/ADBUIDX01.00001 bs=524288

dd if=/HADB_bkup/db/ADBUIDX01.00002
   of=/HADB/db/ADBUIDX01.00002 bs=524288

dd if=/HADB_bkup/db/ADBUIDX01.00003
   of=/HADB/db/ADBUIDX01.00003 bs=524288

dd if=/HADB_bkup/db/ADBUIDX01.00004
   of=/HADB/db/ADBUIDX01.00004 bs=524288

dd if=/HADB_bkup/db/ADBUIDX01.00005
   of=/HADB/db/ADBUIDX01.00005 bs=524288
```

```
dd if=/HADB_bkup/db/ADBUIDX01.00006
   of=/HADB/db/ADBUIDX01.00006 bs=524288

dd if=/HADB_bkup/db/ADBUIDX01.00007
   of=/HADB/db/ADBUIDX01.00007 bs=524288

cp -r /HADB_bkup/archive/*  /HADB/archive

cp -r /HADB_bkup/syndict/*  /HADB/syndict
```

Important

When you recover the files under the DB directory from a backup, copy them into the DB directory (`/HADB/db`) by overwriting their contents. If you simply copy and store the backup files in an empty directory, database compatibility can no longer be maintained, and the validity of subsequent operations of the HADB server cannot be guaranteed.

3. Start the HADB server.

   Once you have recovered the database from the backup that was made, start the HADB server. Execute the following command to start the HADB server:

   ```
   adbstart
   ```

4. Check the HADB server's status.

   After you have started the HADB server, check whether the HADB server's operation mode is the normal mode. Execute the following command to check the HADB server's status:

   ```
   adbls -d srv
   ```

   Check the item `STATUS` in the execution result. If `STATUS` is `ACTIVE`, the operation mode is set to the normal mode.

10.3.4 Backup operation example (using ShadowImage)

This section provides an example of a backup operation using ShadowImage, which is a volume replication facility of a Hitachi disk array system.

Using ShadowImage can reduce the time needed to make a backup and restore from the backup. For details about ShadowImage, see the *ShadowImage User's Guide*.

(1) Note on using ShadowImage

Do not generate a pair by specifying the Conceal mode (`-m noread`).

(2) Directory configuration example

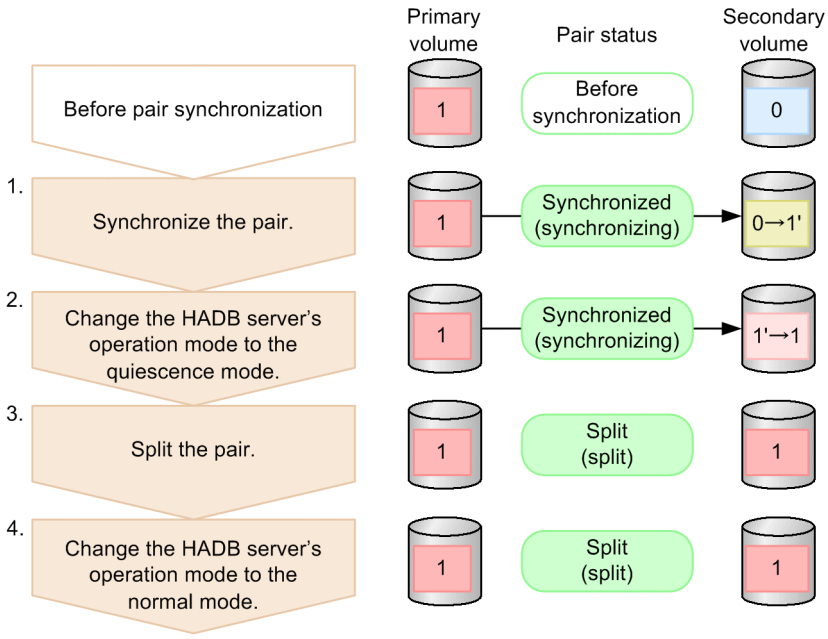
In this example, all the following directories are backed up and stored in the primary volume of ShadowImage.

- DB directory that includes block special files
- Archive directory
- Directory for storing synonym dictionary files

(3) Making a backup

This subsection provides an operation example using ShadowImage to make a backup.

Figure 10-5: Example of using ShadowImage to make a backup



Conditions:

- Following database initialization, the HADB server was started at least once.
- It has been confirmed that there are no non-updatable base tables.
- To suppress database updating, the HADB server's operation mode is changed to the quiescence mode.
- A full backup is used.
- The DB directory, archive directory, and the directory for storing synonym dictionary files are to be backed up.

Procedure:

1. Synchronize the pair.

   Start pair synchronization. Since pair synchronization can take some time, we recommend that you start the pair volume synchronization before changing the HADB server's operation mode to the quiescence mode.

2. Change the HADB server's operation mode to the quiescence mode.

   Execute the `adbchgsrvmode --quiescence` command to change the HADB server's operation mode to the quiescence mode. After you have changed the operation mode to the quiescence mode, wait for pair synchronization to finish.

3. Split the pair.

   When pair synchronization is complete, backup acquisition is also complete. Therefore, split the pair. In this way, the backup that was acquired when pair synchronization started is held in the sub-volume.

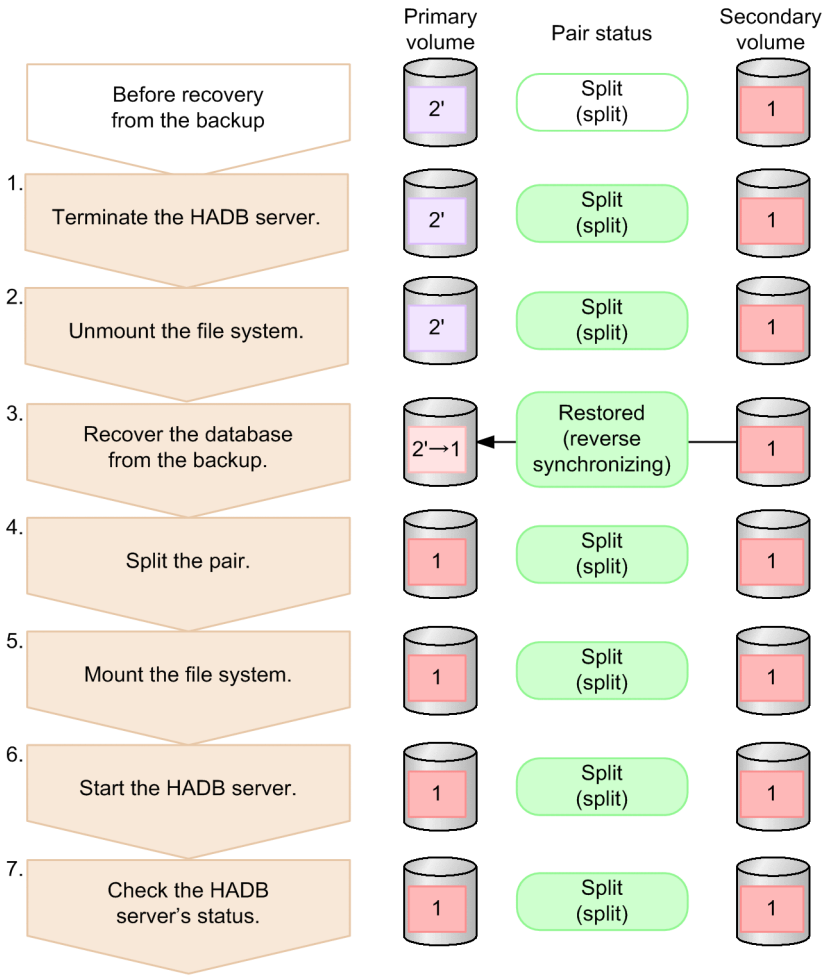
4. Change the HADB server's operation mode to the normal mode.

   Execute the `adbchgsrvmode --normal` command to change the HADB server's operation mode to the normal mode. Backup acquisition using ShadowImage is now finished.

(4) Recovering a database from a backup

This section provides an example of recovering a database from a backup acquired according to (3) Making a backup.

Figure 10-6: Example of using ShadowImage to recover a database from a backup



Procedure:

1. Terminate the HADB server.

   Before recovering the database from the backup, terminate the HADB server. Execute the following command to terminate the HADB server:

   ```
   adbstop
   ```

2. Unmount the file system.

   Unmount the file system that is stored in the primary volume.

3. Recover the database from the backup.

   Reverse-synchronize the pair and recover the database from the backup. Synchronize the content of the sub-volume to the primary volume.

   > **Important**
   >
   > Be careful not to synchronize the content of the primary volume to the sub-volume.

4. Split the pair.

When reverse-synchronization of the pair is complete, recovery from the backup is complete. Therefore, split the pair.

5. Mount the file system.

   Mount the file system that was unmounted in step 2.

6. Start the HADB server.

   Execute the following HADB command to start the HADB server:

   ```
   adbstart
   ```

7. Check the HADB server's status.

   After you have started the HADB server, check whether the HADB server's operation mode is the normal mode. Execute the following command to check the HADB server's status:

   ```
   adbls -d srv
   ```

   Check the item `STATUS` in the execution result. If `STATUS` is `ACTIVE`, the operation mode is set to the normal mode.

10.4 Checking the messages

You monitor HADB's operational status using messages. The messages output by HADB are saved as statistical information and are called a message log.

10.4.1 Message output destinations

The HADB server outputs messages to the standard output, standard error output, server message log files, and syslog. The output destination is determined based on the message type. This subsection describes the message output destinations and explains the types of messages that are output each one.

(1) Standard output

Information messages and response messages are output to the console's standard output.

(2) Standard error output

Error messages and warning messages are output to the console's standard error output.

(3) Server message log file

Messages output by the HADB server are stored in message logs using files created under the server directory. These files are called server message log files.

All HADB server messages are output here.

Four server message log files are created in the HADB server. You can use the environment variable `ADBMSGLOGSIZE` to specify the maximum size of the server message log files that are created.

Note that the server message log files do not store messages that are output by HADB clients.

Note

■ Client message log file

Messages output by an HADB client are stored in message logs using files created under the client directory. These files are called client message log files.

Four client message log files are created in an HADB client. You can use the environment variable `ADBMSGLOGSIZE` to specify the maximum size of the client message log files that will be created.

(4) syslog

Messages are also output to syslog. You can use the HADB server's environment variable `ADBSYSLOGLV` to specify the types of messages to be output to syslog.

10.4.2 Viewing the message logs (message log output destination)

Messages output by HADB are output in message logs to message log files and syslog. The message logs provide the statistical information on the messages out by HADB in the following cases:

- To verify the messages out by HADB
- To investigate the cause when an error occurs in HADB

You can use a text editor to view the message logs that were output by the HADB server or an HADB client to the message log files. To view the message logs that are output to syslog, follow the procedure used for the OS.

The storage destinations of the message log files created in the HADB server or in an HADB client are shown below. *XX* in the file name is a sequential number from `01` to `04`.

- HADB server (server message log file)
  `$ADBDIR/spool/adbmessage`*XX*`.log`
- HADB client (Linux) (client message log file)
  `$ADBCLTDIR/spool/adbmessageclt`*XX*`.log`
- HADB client (Windows) (client message log file)
  `%ADBCLTDIR%\spool\adbmessageclt`*XX*`.log`

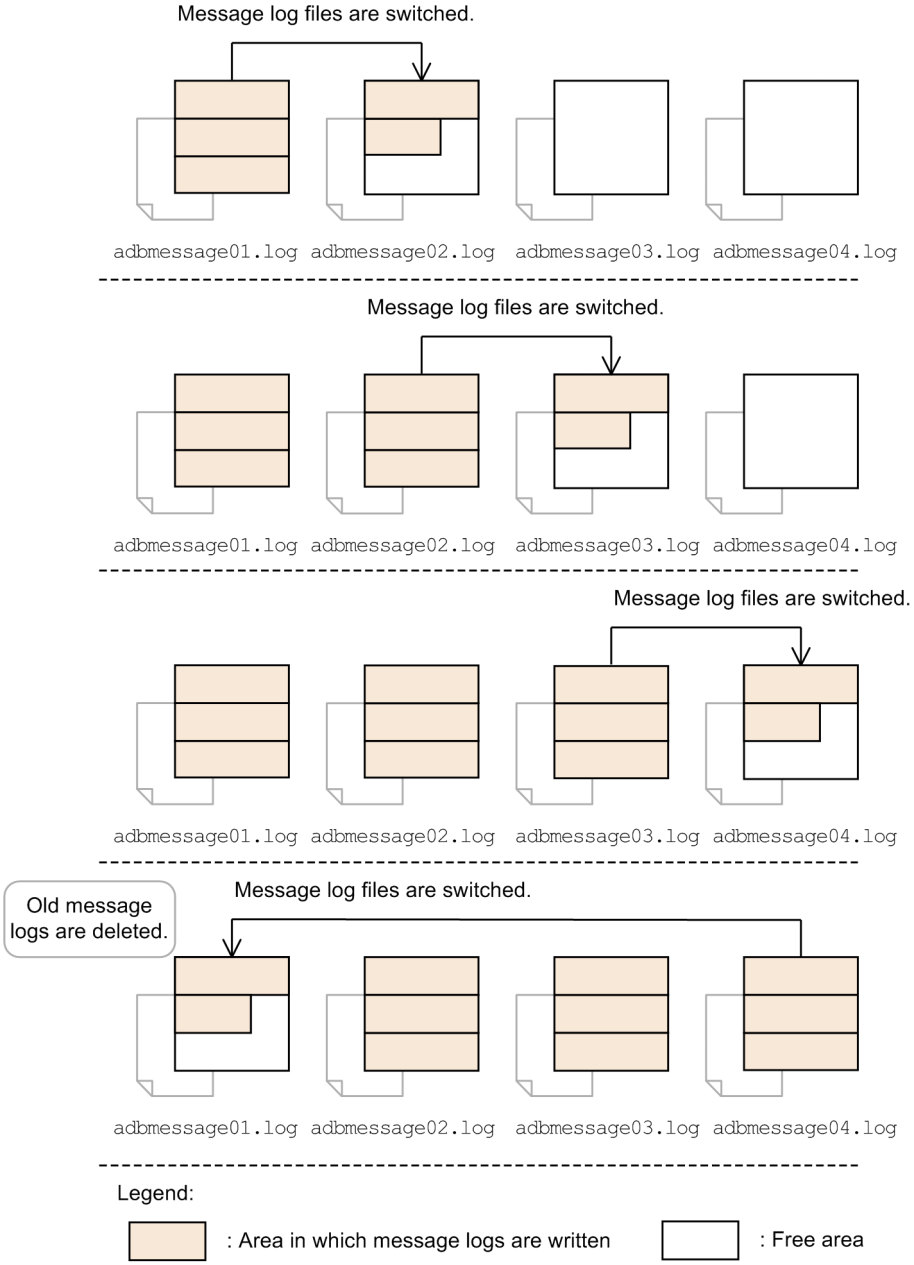
10.4.3 Working with the message log files

The HADB server and an HADB client each creates and uses four message log files.

The maximum size of each message log file is determined by the value specified in the environment variable `ADBMSGLOGSIZE`. When a message log file reaches its maximum capacity, HADB stops outputting message logs to that message log file and outputs message logs to the next message log file. When this occurs, any existing message logs stored in the message log file that is being switched to are deleted.

The following figure shows how message log files are switched, using the four server message log files created in the HADB server as examples.

Figure 10-7: Message log file switching



Explanation

When message logs are output and `adbmessage01.log` reaches its maximum capacity, the message log output destination switches to `adbmessage02.log`.

Likewise, when `adbmessage02.log` reaches its maximum capacity, the message log output destination switches to `adbmessage03.log`. When `adbmessage03.log` reaches its maximum capacity, the message log output destination switches to `adbmessage04.log`.

When `adbmessage04.log` reaches its maximum capacity, the message log output destination switches to `adbmessage01.log`. When this happens, the old message logs stored in `adbmessage01.log` are deleted before new message logs are output.

Note that server message log files are not switched when the HADB server starts. Therefore, message logs are output to the server message log file that was the output destination when the HADB server was last terminated.

To see to which message log file new message logs are being output, check the last file update date and time using the operating system's `ls` command, or the like. The file with the last modified date and time is the message log file currently being used.

However, if you in any way update a message log file, say by using a text editor, the last modified date and time will be changed. Then, you must find the message log file containing the message with the last output date as indicated by the information attached to the message to determine the message log file in which new message logs will be stored.

> **Note**
>
> The message log files of an HADB client work in the same way.

10.4.4 Fall-back mode of the message log file

If the HADB server detects a shortage of free space of the disk that stores message log files, the server outputs only the available amount of message logs depending on the disk free space. This state is called fall-back mode of the message log file.

When the message log file goes into the fall-back mode, the output destination of messages is switched to the next message log file. At this time, the messages in the switching destination message log file are deleted. After switching of message log files is completed, warning message `KFAA40025-W` is output to the switching destination message log file and syslog. However, if the message log file on an HADB client (for Windows) goes into the fall-back mode, the `KFAA40025-W` message is output only to the message log file on the HADB client.

The server message log file on the HADB server might go into the fall-back mode as the result of execution of an SQL statement on an HADB client. In that case, the `KFAA40025-W` message is output also to the message log file on the HADB client.

> **Note**
>
> The processing of forcibly switching a message log file is executed only once when the message log file goes into the fall-back mode. The `KFAA40025-W` message is also output only once when the message log file goes into the fall-back mode. While a message log file is in the fall-back mode, messages are output to the switching destination message log file if it is possible. If it is not possible, messages are not output. If the size of the switching destination message log file reaches the size specified for environment variable `ADBMSGLOGSIZE`, messages are no longer output.

The message log file automatically returns to the normal state in the following condition. An SQL statement or command is executed when the free space of the disk that stores message log files is no less than the double of the value specified for environment variable `ADBMSGLOGSIZE`. After that, If the HADB server detects again a shortage of free space of the disk that stores message log files, the message log file goes again into the fall-back mode.

> **Note**
>
> For environment variable `ADBMSGLOGSIZE`, specify in megabytes the maximum size of a message log file. If specification of the value for `ADBMSGLOGSIZE` is omitted, 16 megabytes is assumed for the maximum size of a message log file.

Checking by a command whether the message log file on the HADB server is in the fall-back mode

To check whether the message log file on the HADB server is in the fall-back mode, execute the `adbls -d srv` command. Check the `MSGLOG_STATUS` column of the execution result.

Example:

```
adbls -d srv

SVID    STATUS    START_TIME             MSGLOG_STATUS
5360    ACTIVE    2017/02/02 09:31:04    FALLBACK
```

If `FALLBACK` is displayed in the `MSGLOG_STATUS` column, the message log file on the HADB server is in the fall-back mode. If the message log file is in the normal state, `NORMAL` is displayed.

For details about how to release the fall-back mode of the message log file, see 15.16.1 Releasing the message log file from fall-back mode.

10.4.5 Suppressing message output to syslog

HADB outputs various types of information to syslog, including error information, transaction information, and system file information. These messages provide important information that indicates HADB's operational status.

However, depending on the operating environment, a large number of messages might be output. If the size of syslog continues to increase because of the messages output by HADB, the disk might run out of free space. You can prevent HADB from causing the disk to run out of free space by suppressing the output of unnecessary messages to syslog.

To suppress messages from being output to syslog, specify a message output level in the environment variable `ADBSYSLOGLV`. For details about environment variable `ADBSYSLOGLV`, see 8.4 Setting environment variables.

The following table shows message output levels and describes the messages that are output to syslog.

Table 10-8: Message output levels and description of messages that are output to syslog

Output level	Description of output message
0	No messages are output to syslog.
1	Only the following messages are output to syslog: • Warning messages indicating that an abnormality occurred in the message log file • Error messages indicating a fatal error such as a process breakdown • Response messages that require a user response
2	Messages output for output level 1 plus the following messages are output to syslog: • Error messages indicating an HADB error • Information messages indicating HADB startup, termination, and other major events
3	Messages output for output levels 1 to 2 plus the following messages are output to syslog: • Error messages indicating an error at the job level
4	Messages output for output levels 1 to 3 plus the following messages are output to syslog: • Warning messages that do not hinder continued operation but report a status (error) to the user
5	Messages output for output levels 1 to 4 plus the following messages are output to syslog: • Information messages indicating thread startup, connection establishment and other minor events
6	All messages that are to be output to syslog are output.

For details about the output level of each message, see *List of message output locations* in the manual *HADB Messages*.

10.4.6 Converting character encoding and improving reliability for syslog (Applying the extended syslog function)

By applying the extended syslog function, you can perform the following operations:

- Converting character encoding for syslog (Converting from Shift-JIS to Unicode)
- Improving reliability for syslog (Re-outputting messages)

> **Note**
>
> The extended syslog function is provided as part of the log environment enhancement option of Hitachi Support 360.

The following describes the operations that you can perform when applying the extended syslog function.

(1) Converting character encoding for syslog (Converting from Shift-JIS to Unicode)

When the character encoding used on the HADB server is Shift-JIS, character encoding for the messages output to syslog is converted from Shift-JIS to Unicode (UTF-8).

Character encoding for the messages output to syslog is unified to Unicode (UTF-8), which has the following advantages:

- Corruption of messages can be prevented when syslog is read.
- Monitoring and managing messages become easy.

Note that conversion of character encoding for syslog is enabled only when Shift-JIS is applied to the following environment variables:

▪ Applicable environment variables

- `LANG`
- `ADBLANG`

For details about environment variables, see 8.4 Setting environment variables.

(2) Improving reliability for syslog (Re-outputting messages)

When outputting a message to syslog fails, the message is output again. Thus, you can prevent loss of messages.

(3) Prerequisite software programs and notes (Applying the extended syslog function)

To apply the extended syslog function, you need to install the following prerequisite software programs before starting the HADB server.

Prerequisite software programs

- HA Logger Kit for Linux (Extended syslog function): 04-00-/A or later
- Hitachi Code Converter[#]: 03-03-/A or later

#

You need to install this software program if you want to convert character encoding for syslog (convert from Shift-JIS to Unicode).

> **Important**
>
> To use the multi-node function, or to use a cold standby configuration, install the prerequisite software programs on all server machines.

If prerequisite software programs have been installed, `KFAA50050-I` message is output to the syslog and server message log file when the HADB server starts. `KFAA50050-I` message indicates that the extended syslog function has been applied.

Note that, if you install the prerequisite software programs while the HADB server is running, the extended syslog function is not applied. If the HADB server is running, install the prerequisite software programs in the following procedure.

Procedure:

1. Terminate the HADB server normally.
2. Install the prerequisite software programs.
3. Start the HADB server.

Also, note that the extended syslog function is not applied when environment variable `ADBSYSLOGLV` on the server machine on which the HADB server is installed is set to `0`. This is because, in this condition, no messages are output to syslog.

10.5 Monitoring the resource usage (list of messages to be monitored)

This section describes the messages that need to be monitored when you monitor the usage of resources used by the HADB server.

The HADB server runs by using various resources. Therefore, if any of the following events occur, the operational status of the HADB server might be affected:

- The resources of the server machine or the OS (such as disks, CPU, or memory) are insufficient.
- The resources managed by the HADB server (such as the numbers of connections and chunks) are insufficient.
- Failure occurs in a disk or the network.
- The lock control causes a wait for processing.

If such an event occurs, you need to promptly take the appropriate actions. To do this, the HADB administrator needs to check the message output by the HADB server, and monitor the resource usage.

The following table lists and describes the resources used by the HADB server and the messages to be monitored. For details about how to handle individual output messages, see the manual *HADB Messages*.

Table 10-9: List of the resources used by the HADB server and the messages to be monitored

No.	Resource type		Message to be monitored	Description
1	Locked resource		• KFAA31371-E • KFAA31663-E • KFAA40005-E • KFAA50290-E	If one of the messages in the left column is output, the process of allocating a locked resource is placed in wait status. For details about how to handle the message, see the column that indicates the handling of the applicable message. For details about how to check whether the process of allocating a locked resource has gone into wait status, see 10.8.3 Checking whether the process of reserving locked resources has gone into wait status. If the process of allocating a locked resource repeatedly goes into wait status, the application or command that allocated the locked resource might have been running for a long time. Also, see 10.8.1 Checking the application or command processing status.
2	Memory	Memory managed by the HADB server	• KFAA40021-W • KFAA40024-I	The KFAA40021-W message is a warning message that is output when the memory usage on the HADB server is no less than the ratio specified for the adb_sys_memory_limit_wrn_pnt operand in the server definition. (This operand indicates the trigger for outputting the warning message regarding the memory usage on the HADB server.) If the KFAA40021-W message is output, check the memory usage on the HADB server. For details about how to check the memory usage, see 10.6.3 Checking the memory usage status for each real thread. The KFAA40024-I message is an information message that is output when the memory usage on the HADB server is no more than the ratio specified for the adb_sys_memory_limit_wrn_pnt operand in the server definition. (This operand indicates the trigger for resetting the status indicating that a warning message has been output.) Multi-node function When a multi-node function is being used, the messages in the left column might be output on all nodes.
3			• KFAA30930-E • KFAA40007-E	If one of the messages in the left column is output, the memory managed by the HADB server is insufficient.

No.	Resource type		Message to be monitored	Description
			• KFAA40203-E • KFAA40212-E • KFAA51011-W • KFAA96605-E	When you execute an SQL statement or command, if memory shortage occurs, any of the following messages is output: **When SQL statement is executed** • KFAA30930-E • KFAA40007-E • KFAA51011-W **When a command is executed** • KFAA40007-E • KFAA40203-E • KFAA40212-E • KFAA96605-E For details about how to handle the above output messages, see the column that indicates the handling of KFAA40007-E message. In addition, regularly execute the adbstat command to check the usage status of the memory used by the HADB server. (Check whether memory usage is near the upper limit.) For details, see 10.6.1 Checking the usage status of all memory.
4		Memory used by the libraries that the HADB server uses	• KFAA40207-E • KFAA40208-E • KFAA40293-E • KFAA91000-E • KFAA92000-E • KFAA96211-E • KFAA96457-E	If one of the messages in the left column is output, the memory used by the libraries that the HADB server uses is insufficient. When you execute a command, if memory shortage occurs, any of the messages in the left column is output. For details about how to handle the message, see the column that indicates the handling of the applicable message.
5	Shared memory		KFAA40002-E	If the message in the left column is output, an error occurred when a shared memory was accessed. For details about how to handle the message, see the column that indicates the handling of the applicable message.
6	CPU		None	The HADB server does not output messages about CPU usage rate. To monitor the CPU usage rate of the HADB server, execute the following OS commands: • sar command • mpstat command • top command
7	Inter-process communication		• KFAA30722-E • KFAA30723-E • KFAA50300-E • KFAA50301-E • KFAA50303-E • KFAA50304-E • KFAA50305-E • KFAA50306-E • KFAA50307-E • KFAA50308-E • KFAA50309-E • KFAA50310-E • KFAA50311-E	If one of the messages in the left column is output, an error occurred in inter-process communication of the HADB server. For details about how to handle the message, see the column that indicates the handling of the applicable message.

No.	Resource type		Message to be monitored	Description
8	Directory	DB directory	KFAA30959-E	If the message in the left column is output, an error occurred when the DB directory was accessed. For details about how to handle the message, see the column that indicates the handling of the applicable message.
9		Archive directory	KFAA31666-E	If the message in the left column is output, an error occurred when the archive directory was accessed. For details about how to handle the message, see the column that indicates the handling of the applicable message.
10			KFAA61212-W	If the message in the left column is output, an error occurred in the processing for deleting the archive directory. For details about how to handle the message, see the column that indicates the handling of the applicable message.
11	File	Message log file	KFAA40025-W	If the message in the left column is output, one of the following message log files is in the fall-back mode: • Server message log file • Client message log file If the message in the left column is output, release the fall-back mode of the message log file. For details about how to release the fall-back mode, see 15.16.1 Releasing the message log file from fall-back mode. The message in the left column is output to syslog even if the message cannot be output to the message log file. Therefore, you need to monitor both the message log file and syslog.
12			KFAA50057-W	Multi-node function If the message in the left column is output, the message log file on a node is in the fall-back mode. In this case, release the fall-back mode of the message log file. For details about how to release the fall-back mode, see 15.16.1 Releasing the message log file from fall-back mode.
13		DB area file	• KFAA30959-E • KFAA60014-W	If the message in the left column is output, operation on the applicable file failed. For details about how to handle the message, see the column that indicates the handling of the applicable message.
14		Archive file		
15		System file		
16		CSV file whose information is input to the ADB_CSVREAD function		
17		File on which the command is executed	• KFAA40204-E • KFAA40205-E	
18		HADB dump file	• KFAA40015-E • KFAA40016-E	
19		Message catalog file	• KFAA50018-E • KFAA50019-E	
20		Shared memory ID storage file	KFAA50028-E	

No.	Resource type		Message to be monitored	Description
21		Shared memory dump file	KFAA50029-E	
22		Semaphore set ID storage file	KFAA50037-E	
23		Command status file	KFAA50243-E	
24		Temporary work file	KFAA50245-E	
25		Index record file	KFAA50246-E	
26		SQL trace file	KFAA51010-W	
27		Synonym dictionary file	• KFAA30959-E • KFAA34008-E	
28			KFAA51537-W	Multi-node function If the message in the left column is output, the latest synonym dictionary file has not been stored in the directory specified for the adb_syndict_node_storage_path operand in the server definition. In this case, performance of synonym search might be deteriorated. For details about how to handle the message, see the column that indicates the handling of that message.
29		Unload file	KFAA61400-W	If the message in the left column is output, deletion of the unload file that was temporarily created when the adbreorgsystemdata command was executed failed. For details about how to handle the message, see the column that indicates the handling of the applicable message.
30		Audit trail file	KFAA51404-E	If the message in the left column is output, an operation on an audit trail file has failed. For details about how to handle the message, see the column that indicates the handling of that message.
31			KFAA51411-W	The message in the left column is output if the current audit trail file cannot be found when attempting to swap the audit trail file. The audit trail information output to the current audit trail file might have been lost.
32			KFAA81401-I	The message in the left column is output when the audit trail file has been swapped. If you perform any of the following operations for the audit trail file, monitor the message in the left column: • Moving the audit trail file from the audit trail directory to the audit trail storage directory • Converting the audit trail file into a common format audit trail file
33			KFAA81402-I	The message in the left column is output when the audit trail file has been renamed. If you perform any of the following operations for the audit trail file, monitor the message in the left column: • Moving the audit trail file from the audit trail directory to the audit trail storage directory

No.	Resource type		Message to be monitored	Description
				• Converting the audit trail file into a common format audit trail file
34		File used by the updated-row columnizing facility	KFAA41220-W	If the message in the left column is output, there is an error in a file that is used for the updated-row columnizing facility. For details about how to handle the message, see the column that indicates the handling of that message.
35	Node		• KFAA50053-E • KFAA50153-E • KFAA50154-W • KFAA60008-W • KFAA60009-E • KFAA60010-W	If one of the messages in the left column is output, an abnormality occurs in the node on which the message is output. For details about how to handle the message, see the column that indicates the handling of the applicable message.
36	DB area		• KFAA30756-E • KFAA41206-I • KFAA61210-E	If one of the messages in the left column is output, the free disk space is insufficient, and data cannot be stored in the target DB area. For details about how to handle the message, see the column that indicates the handling of the applicable message.
37	Log file		• KFAA31711-E • KFAA41210-E	If one of the messages in the left column is output, the number of user log files is insufficient. For details about how to handle the message, see the column that indicates the handling of the applicable message. Note that, if you omit specifying the adb_log_usrfile_num operand in the server definition, an error due to insufficient user log files will not occur. For details about the adb_log_usrfile_num operand in the server definition, see the description of the adb_log_usrfile_num operand in 7.2.3 Operands related to system logs (set format).
38			KFAA61211-E	If the message in the left column is output, the free space of the disk in which the system log file is stored is insufficient. For details about how to handle the message, see the column that indicates the handling of the applicable message.
39			KFAA81215-I	If the message in the left column is output, database recovery processing is being performed. For details about how to handle the message, see the column that indicates the handling of that message. The message in the left column is output at intervals specified for the adb_log_rec_msg_interval operand in the server definition until the database recovery processing is complete. For details about the adb_log_rec_msg_interval operand in the server definition, see the description of the adb_log_rec_msg_interval operand in 7.2.3 Operands related to system logs (set format).
40	Buffer		• KFAA30919-E • KFAA41201-E	If one of the messages in the left column is output, the number of buffers is insufficient. For details about how to handle the message, see the column that indicates the handling of the applicable message.
41			• KFAA30953-E • KFAA41202-E	If one of the messages in the left column is output, the load on the buffer is too high. For details about how to handle the message, see the column that indicates the handling of the KFAA41202-E message.
42	Chunk		KFAA51245-E	If the message in the left column is output, a new chunk cannot be created because the number of chunks will exceed the maximum

No.	Resource type	Message to be monitored	Description
			number. The number of chunks that can be created in a DB area is about to exceed the maximum number. For details about how to handle the message, see the column that indicates the handling of the applicable message.
43		KFAA51246-E	If the message in the left column is output, any new chunk cannot be created because the number of chunks will exceed the maximum number. The number of chunks that can be created in a table is about to exceed the maximum number. For details about how to handle the message, see the column that indicates the handling of the applicable message.
44	Hash table area	KFAA51130-W	If the message in the left column is output, a work table has been created because the hash table area was insufficient. For details about how to handle the message, see the column that indicates the handling of the applicable message. Also, check whether any problems occur when the hash table area is insufficient. (Check whether the situation in which the hash table area is insufficient is assumed in the database design.)
45	Connection	• KFAA40020-W • KFAA40023-I	The KFAA40020-W message is a warning message that is output when the number of connections to the HADB server is no less than the ratio specified for the adb_sys_max_users_wrn_pnt operand in the server definition. (This operand indicates the trigger for outputting the warning message regarding the maximum number of concurrent connections.) If the KFAA40020-W message is output, check whether there is an application or command that is illegally connecting to the HADB server. For details about how to check this, see 10.8.1 Checking the application or command processing status. The KFAA40023-I message is an information message that is output when the number of connections to the HADB server is no more than the ratio specified for the adb_sys_max_users_wrn_pnt operand in the server definition. (This operand indicates the trigger for resetting the status indicating that a warning message has been output.) If the client-group facility is being used, instead of the value specified for the adb_sys_max_users_wrn_pnt operand, the following values specified for the -w option of the adbcltgrp operand are applied: • *Trigger for outputting the warning message regarding the maximum number of concurrent connections* • *Trigger for resetting the status indicating that the warning message has been output* Multi-node function When a multi-node function is being used, the messages in the left column are output on the master node.
46		KFAA30932-E	If the message in the left column is output, an application or command cannot connect to the HADB server. The number of connections to the HADB server exceeds the maximum number of concurrent connections. For details about how to handle the message, see the column that indicates the handling of the applicable message. Check whether the value specified for the adb_sys_max_users operand in the server definition is appropriate. For details about the adb_sys_max_users operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).

No.	Resource type	Message to be monitored	Description
47	Statement handle	KFAA30931-E	If the message in the left column is output, the number of statement handles exceeds the upper limit. For details about how to handle the message, see the column that indicates the handling of the applicable message.
48	Base table	• KFAA30811-E • KFAA30812-E	If one the messages in the left column is output, the number of base tables or indexes that can be defined in the system exceeds the upper limit. Otherwise, the number of base tables or indexes that can be stored in the target DB area exceeds the upper limit. For details about how to handle the message, see the column that indicates the handling of the applicable message.
49	Index		
50	Viewed table	KFAA30811-E	If the message in the left column is output, the number of viewed tables that can be defined in the system exceeds the upper limit. For details about how to handle the message, see the column that indicates the handling of the applicable message.
51	system table (base table)	• KFAA61213-W • KFAA61214-W	If one of the messages in the left column is output, there are too few unused segments in the system-table DB area. Therefore, you need to execute the adbreorgsystemdata command to reorganize the system table. For details about how to handle the message, see the column that indicates the handling of the applicable message.
52	HADB user	KFAA30811-E	If the message in the left column is output, the number of HADB users that can be defined in the system exceeds the upper limit. For details about how to handle the message, see the column that indicates the handling of the applicable message.

10.6 Checking the memory usage

This section explains how to check the usage status of the memory used by the HADB server.

10.6.1 Checking the usage status of all memory

To check the usage status of all memory used by the HADB server, execute the `adbstat` command on the HADB server. This subsection explains the timing and method for checking the usage status of all memory.

For details about the `adbstat` command options and individual output items, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

(1) Timing for checking the usage status of all memory

To determine if the size of all memory being used by the HADB server matches the memory size specified in the server definition, check the usage status of all memory on a regular basis.

(2) Method of checking the usage status of all memory

The `adbstat` command entered as follows is used to check the usage status of all memory.

■ Command to be executed

```
adbstat
```

In the HADB server statistical information that is output by executing the `adbstat` command, check the following item:

- `Total_memory_max_size` (maximum usage of all memory)

Check the value of `Total_memory_max_size` and the value of the `adb_sys_memory_limit` operand in the server definition. If the `adb_sys_memory_limit` operand has not been specified in the server definition, check the values of the following operands:

- `adb_sys_rthd_area_max` operand
- `adb_sys_proc_area_max` operand

If the value of `Total_memory_max_size` differs greatly from the operand value in the server definition, the memory size might not be set appropriately. In such a case, change the operand value in the server definition so that it is greater than the value of `Total_memory_max_size`.

10.6.2 Checking the usage status of the shared memory

To check the usage status of the shared memory, execute the `adbls -d shm` command on the HADB server. This displays the total amount of shared memory being used by the HADB server. This subsection explains when to check the usage status of the shared memory, how to check it, and what to do if the amount of free space becomes too small.

For details about the `adbls -d shm` command options and individual output items, see *adbls -d shm (Display Shared Memory Information)* in the manual *HADB Command Reference*.

(1) When to check the usage status of the shared memory

To ensure that the shared memory does not run out of free space, check the usage status of the shared memory regularly.

If the shared memory runs out of free space, an error message is output. Take the necessary corrective action according to the error message. When doing so, execute the `adbls -d shm` command to check the usage status of the shared memory.

(2) How to check the usage status of the shared memory

The following is an example of the shared memory usage status that is output when the `adbls -d shm` command is executed.

■ Shared memory usage status example

```
SHM_NUMBER    SHM_SIZE
8             1424683947
```

The following describes how to check the usage status of shared memory based on the above example.

To check the usage status of shared memory:

The amount of shared memory (in bytes) being used by the HADB server is displayed under `SHM_SIZE`. In this example, 1,424,683,947 bytes (approximately 1.4 gigabytes) is being used.

If the difference between this value and the value specified in the OS kernel parameter `SHMMAX` is small, the amount of available free space in the shared memory is considered insufficient. In this case, take action as explained in (3) Steps to take when the amount of free space in the shared memory is insufficient.

If the difference is large, no action is necessary because there seems to be enough free memory.

(3) Steps to take when the amount of free space in the shared memory is insufficient

If the difference between the value displayed under `SHM_SIZE` and the value specified in `SHMMAX` is small, you might need to increase the values specified for the OS kernel parameters `SHMMAX`, `SHMMNI`, and `SHMALL`.

10.6.3 Checking the memory usage status for each real thread

To check the memory usage status for each real thread, execute the `adbls -d mem` command on the HADB server.

For details about the options of the `adbls -d mem` command and the information that is output, see *adbls -d mem (Display the Memory Usage Status)* in the manual *HADB Command Reference*.

■ Timing of checking the memory usage status

Check the memory usage status regularly for each real thread.

■ Checking the memory usage status

To check the memory usage status on the HADB server or for each real thread on the HADB server, execute the following command:

```
adbls -d mem
```

Example of the command execution result

```
THREAD-NO   LIMIT          USE          FREE         MAX
0           1073741824     105840224    162660848    105840224
1           1073741824     901024       267600048    901056
2           1073741824     900512       267600560    900512
3           1073741824     900512       267600560    900512
4           1073741824     900512       267600560    900512
5           1073741824     900512       267600560    900512
:               :            :            :          :
```

Explanation

- On a line whose `THREAD-NO` is 1 or larger, the memory usage status for the corresponding real thread is displayed. On the line whose THREAD-NO is 0, information regarding the process common memory is displayed.
- The `LIMIT` column displays the upper limit value of the memory size (in bytes) that can be used by the HADB server.
- The `USE` column displays the memory size (in bytes) that is being used by the HADB server.
- The `FREE` column displays the available memory size (in bytes).
- The `MAX` column displays the maximum value of the memory size (in bytes) that was used by the HADB server.

To check which application or command is using an individual real thread, see 10.7 Checking the threads running on the HADB server.

10.7 Checking the threads running on the HADB server

To perform the following, execute the `adbls -d thd` command:

- Checking real threads and pseudo-threads that are running on the HADB server
- Identifying the connection or SQL statement that is using a real thread

The following shows an example of the execution result of the `adbls -d thd` command.

```
adbls -d thd

THREAD-NO           T-ID(OS)            TOTAL_PSEUDO_THREAD  TYPE    CNUMBER    SQL_ID              CLIENT_GROUP
1                   4102559552          6                    SYSTEM
2                   0951879424          6                    SYSTEM
3                   0962369280          6                    SYSTEM
4                   2897991424          6                    SYSTEM
5                   3882063616          6                    CONNECT
6                   3867367168          6                    CONNECT
7                   3852678912          6                    CONNECT
8                   3837990656          6                    CONNECT
9                   3619669760          6                    CONNECT
10                  3604981504          6                    CONNECT 6                              GROUP01
11                  3590293248          6                    CONNECT
12                  3575604992          6                    CONNECT
13                  3221223168          6                    CONNECT
14                  3206526720          6                    CONNECT
15                  3191838464          6                    CONNECT
16                  2942297856          6                    PROCESS
17                  1912383232          6                    PROCESS
18                  0972859136          6                    PROCESS
19                  0033335040          6                    PROCESS 6          1                   GROUP01
20                  3388778240          6                    PROCESS
21                  2449254144          6                    PROCESS
22                  1509730048          6                    PROCESS
23                  0570205952          6                    PROCESS
24                  3925649152          6                    PROCESS
25                  2986125056          6                    PROCESS 6          1                   GROUP01
```

Explanation

A list of threads that are running on the HADB server is displayed.

For an explanation of individual items of the execution result of the `adbls -d thd` command, see *Example* in *adbls -d thd (Display the Thread Status)* in the manual *HADB Command Reference*.

To check the processing status of the application or command that corresponds to the connection that is using a real thread, see 10.8.1 Checking the application or command processing status.

10.8 Checking the transaction processing status

This section explains how to check the transaction processing status after executing an application or command.

10.8.1 Checking the application or command processing status

If the transaction processing time is longer than expected when an application or command is executed, you need to check the transaction processing status and take the necessary action.

How to check the transaction processing status is described below. Note that you cannot check the processing status of the `adbinit` or `adbstat` command.

To check the transaction processing status:

1. Execute the `adbls -d cnct` command.

   Check whether the target connection is displayed under the item `PROGRAM`.

   ■ To check the application processing status

   The application identifier specified for the `adb_clt_ap_name` operand in the client definition is displayed. The following shows an example of the execution result of the `adbls -d cnct` command.

   Execution result example

```
CID  CNUMBER  CONNECT_TIME         PROGRAM  C-PID  IP-ADDRESS   STATUS   TRN_ISO_LV      ACCESS_MODE
1    2        2016-06-08 10:11:54  AP001    20770  (127.0.0.1)  STARTED  READ_COMMITTED  READ_ONLY

CONNECTION_INFORMATION           NODE_NO  CLIENT_TYPE    CLIENT_GROUP  SQL_ELAPSED_TIME
000010000000002-000050c5df220700          AP(C Library)  GROUP1        3005125
```

   If an application identifier is displayed, proceed to step 3.

   If the `adb_clt_ap_name` operand is not specified, `********` is displayed. In this case, proceed to step 2.

   > **Note**
   >
   > When SQL statements are being executed, `SQL_ELAPSED_TIME` displays the elapsed time (in microseconds) of the SQL statement that has been executing for the longest time since it started.

   ■ To check the command processing status

   If connection is established, the applicable command name (such as `adbimport`) is displayed. For the `adbsql` command, the application identifier specified for the `adb_clt_ap_name` operand in the client definition is displayed.

   If a command name is displayed, proceed to step 3. Otherwise, proceed to step 2.

2. Execute the `adbls -d srv` command.

   If the target connection is not displayed, check whether the HADB server is running. The following shows an example of the execution result of the `adbls -d srv` command.

   Execution result example

```
SVID  STATUS  START_TIME            MSGLOG_STATUS
6477  ACTIVE  2017/03/02 14:18:34   NORMAL
```

   If the item `STATUS` shows `ACTIVE`, the HADB server is active. Proceed to step 6.

   If the item `STATUS` does not show `ACTIVE`, execute the `adbstart` command to start the HADB server, and then re-execute the application or command.

3. Check the connection ID.

   If a target connection is displayed when you execute the `adbls -d cnct` command in step 1, check whether a connection ID is displayed under CID. If a connection ID is displayed, proceed to step 4. Otherwise, proceed to step 6.

4. Execute the `adbstat -c cnct -n` command.

   In order to check the processing status of the transaction, execute the `adbstat -c cnct -n` command based on the application identifier checked in step 1. For the `-n` option, specify the application identifier.

   When the connection operation information is displayed, refer to the following output items to check the transaction processing status:

   - `SQL_execute_wait_total_time` (total SQL statement execution wait time)
   - `SQL_execute_wait_cnt` (number of times SQL statements waited for execution)
   - Execution time of each SQL statement executed

     `SELECT_total_time` (`SELECT` statement execution time)

     `INSERT_total_time` (`INSERT` statement execution time)

     `UPDATE_total_time` (`UPDATE` statement execution time)

     `DELETE_total_time` (`DELETE` statement execution time)

     `PURGE_CHUNK_total_time` (`PURGE CHUNK` statement execution time)

   If the value of any output item has increased significantly since the last time the `adbstat` command was executed, processing might be taking too long. If you need to terminate forcibly the transaction being processed, go to step 5.

   If you are executing the `adbstat` command for the first time or if the results of the previously executed `adbstat` command are not available, you cannot determine whether processing is taking too long.

5. Execute the `adbcancel` command.

   If needed, forcibly terminate the transactions being processed.

   To forcibly terminate the transaction being processed, execute the `adbcancel` command and specify in the `-u` option the connection ID confirmed in step 3. After the forced termination, re-execute the application or command.

6. Examine the application.

   If the HADB server is active but no connection ID is displayed, the target application has not successfully connected to the HADB server. Therefore, check whether the application is able to connect to the HADB server. In addition, reassess the values specified for the `adb_clt_rpc_srv_host` and `adb_clt_rpc_srv_port` operands of the client definition.

10.8.2 Checking whether the process of allocating processing real threads has gone into wait status

This subsection described how to check whether the process of allocating processing real threads has gone into wait status after execution of an application or command.

To check the transaction processing status:

1. Execute the `adbls -d cnct` command.

   Check the item `STATUS`. If `THREAD_WAITING` is displayed, the process of allocating processing real threads has gone into wait status.

   If you cannot wait for other applications or commands that are using processing real threads to terminate, proceed to step 2.

2. Execute the `adbcancel` command.

   Execute the `adbcancel` command to forcibly terminate other applications and commands that are using processing real threads.

To prevent the process of allocating processing real threads from going into wait status when you execute an application or command, see 6.23.2 Points to consider about the number of processing real threads to be used during command execution.

For details about the `adbls -d cnct` and `adbcancel` commands, see the manual *HADB Command Reference*.

10.8.3 Checking whether the process of reserving locked resources has gone into wait status

This subsection described how to check whether the process of reserving locked resources has gone into wait status after execution of an application or command.

To check the transaction processing status:

1. Execute the `adbls -d cnct` command.

   Check the connection ID of the target application or command. Check the item `CID`.

2. Execute the `adbls -d lock` command.

   Compare the connection ID checked in step 1 with the connection ID displayed under `CID`. Then, check the item `STATUS`. If `WAITING` is displayed, the process of reserving the locked resources to be used by the target application or command has gone into wait status.

   If you cannot wait for other applications or commands that are using the locked resources to terminate, proceed to step 3.

3. Execute the `adbcancel` command.

   Execute the `adbcancel` command to forcibly terminate other applications and commands that are using the locked resources.

To prevent the process of reserving locked resources from going into wait status when you execute an application or command, take whatever action is necessary based on the explanation in 6.23.3 Points to consider about locking during concurrent command execution

For details about the `adbls -d cnct`, `adbls -d lock`, and `adbcancel` commands, see the manual *HADB Command Reference*.

> **Note**
>
> If the type of locked resource is a table or pre-processing table, you can determine the table's ID by checking the `RESOURCEID` item that is output by the `adbls -d lock` command. You can then use the table ID as a search key to identify the table name of the table whose acquisition is being requested by application programs and commands. The following bullet points explain how to determine a table name from its table ID:
>
> - When the table ID is in the range from `0x00020001` to `0x000200C8`
>
>   You can identify the table name from the table ID as explained in (1) List of dictionary tables under B.1 Dictionary table overview.

- When the table ID is in the range from 0x000200C9 to 0x00020190

  You can identify the table name from the table ID as explained in (1) List of system tables under C.1 System table overview.

- Other table IDs

  You can identify the table name from the table ID by searching the dictionary table as explained in (1) When identifying the table name from a table ID under B.22 Searching a dictionary table.

10.8.4 Checking the status when the transaction has been rolled back

This subsection describes how to check whether the transaction has been rolled back after execution of an application or command. This subsection also describes how to check the progress of the rollback when a transaction has been rolled back.

■ Checking whether the transaction has been rolled back

To check whether the transaction has been rolled back, execute the `adbls -d cnct` command. Check the output item `STATUS` to determine whether there is an application or command for which "`ROLLBACKING`" is displayed for that item. If "`ROLLBACKING`" is displayed, the corresponding application or command has been rolled back.

■ Checking the progress of the rollback

If the rollback operation for a transaction takes more time than you expected, check whether the `KFAA81215-I` message has been output. In the `KFAA81215-I` message, you can check the progress of database recovery processing in connection with the rollback of a transaction.

Based on the information in the `KFAA81215-I` message, estimate the time necessary until the database recovery processing is complete and the subsequent processing can resume.

Note that the `KFAA81215-I` message is output at regular intervals, according to the value specified for the `adb_log_rec_msg_interval` operand in the server definition.

■ What to be performed after the rollback operation is completed

After the rollback operation of a transaction is completed, check whether the `KFAA81211-I` message has been output. In the `KFAA81211-I` message, information regarding the transaction recovery processing is output.

Check the `KFAA81211-I` message and consider reducing the number of updates of database per transaction. For example, consider increasing the number of commits for a transaction that took a long time for recovery processing.

Note that the `KFAA81211-I` message is output if there is a transaction that took a longer time for its recovery processing than the value specified for the `adb_log_rec_msg_interval` operand in the server definition.

Based on the transaction ID that is output to the `KFAA81211-I` message, you can identify the SQL statement that was executed for the corresponding transaction. Use the transaction ID as the key, and check the value for `tran_id` that was output to the "SQL statement execution information" in the SQL trace information. The SQL statement whose `tran_id` matches the transaction ID is the SQL statement executed for the corresponding transaction. For details about the SQL statement execution information, see (2) SQL statement execution information in 10.11.2 Information that is output as SQL trace information.

10.9 Checking the database status and usage

This section explains how to use the `adbdbstatus` command to check the status and usages of the HADB database.

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

10.9.1 Checking the database usage

This subsection explains how to use the `adbdbstatus` command to check the HADB database status and its usage.

Example of the command to be executed

This example outputs summary information for all DB areas:

```
adbdbstatus -d summary -c dbarea -S M
```

Executing this command outputs summary information for all DB areas to the standard output. The following are the items to be checked in the results that are output by the `adbdbstatus` command:

(1) Checking the usage rate of each DB area

Using a value output under `DBarea_name` as a key, check the associated value output under `Used_ratio`. If the value output under `Used_ratio` is close to 100%, consider one of the following corrective measures based on the type of DB area file.

When the DB area file is a block special file

If you continue to store data when the value is close to 100%, the DB area file might become full. Therefore, consider allocating free space on the disk that stores block special files based on the explanation in 15.3.1 When a free space shortage is caused by an increase in the size of the DB area files.

When the DB area file is a regular file

If you continue to store data when the value is close to 100%, the DB area will be extended automatically. When automatic extension occurs, the size of the DB area will automatically increase, but the performance of update processing while DB area file expansion is underway will decline. If DB area automatic extension fails, see 15.3.2 When a free space shortage is caused by failed DB area automatic extension.

(2) Checking the usage of each DB area

Using the value output under `DBarea_name`, check the value output under `MB_Used_segments`. You can determine the size of the area (in megabytes) that is being used to store data in each DB area.

10.9.2 Checking the status and usage of a base table

If you acquire table summary information by using the `adbdbstatus` command, you can check the status and usage of a base table.

(1) Checking whether a base table is non-updatable

To check whether a base table is non-updatable, execute the following `adbdbstatus` command.

Example of the command to be executed

This example outputs summary information for table ADBUSER01.T1.

```
adbdbstatus -d summary -c table -n ADBUSER01.T1
```

Executing the adbdbstatus command outputs summary information for table ADBUSER01.T1 to the standard output.

In the table summary information, check the value output under Non-updatable. If non-updatable is output, the base table cannot be updated.

To release the base table from non-updatable status, check the values output under Command_status and Rerun_command, and re-execute the command that is output under Rerun_command.

(2) Checking the usage of a base table

To check the usage of a base table, execute the following adbdbstatus command.

Example of the command to be executed

This example outputs summary information for table ADBUSER01.T1.

```
adbdbstatus -d summary -c table -n ADBUSER01.T1 -S M --shared-lock
```

Executing the adbdbstatus command outputs summary information for table ADBUSER01.T1 to the standard output.

In the table summary information, check the value output under MB_Used_pages. You can determine the size of the area (in megabytes) that is being used by the base table data stored in the DB area.

10.9.3 Checking the status and usage of a B-tree index

If you acquire index summary information by using the adbdbstatus command, you can check the status and usage of a B-tree index.

(1) Checking whether a B-tree index is in unfinished status

To check whether a B-tree index is in unfinished status, execute the following adbdbstatus command.

Example of the command to be executed

This example outputs summary information for B-tree index ADBUSER01.IDX1.

```
adbdbstatus -d summary -c index -n ADBUSER01.IDX1
```

Executing the adbdbstatus command outputs summary information for B-tree index ADBUSER01.IDX1 to the standard output.

Check the value output under Unfinished in the summary information for the index. If unfinished is output, the B-tree index is in unfinished status.

To release the B-tree index from unfinished status, see 15.9.1 Steps to take when unfinished status is applied to a B-tree index.

(2) Checking whether a unique index violates the uniqueness constraint

To check whether a unique index violates the uniqueness constraint, execute the following `adbdbstatus` command.

Example of the command to be executed

This example outputs summary information for B-tree index `ADBUSER01.IDX1`.

```
adbdbstatus -d summary -c index -n ADBUSER01.IDX1
```

Executing the `adbdbstatus` command outputs summary information for B-tree index `ADBUSER01.IDX1` to the standard output.

Check the value output under `Unique_constraint_violated` in the summary information for the index.

- If `unique_constraint_violated` is output, the B-tree index is in violation of the uniqueness constraint. To release the uniqueness constraint violation, see (2) Steps to take when the uniqueness constraint is violated in 15.9.2 Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output).
- If `unknown` is output, it cannot be determined whether the B-tree index is in uniqueness constraint violation. After using the `adbidxrebuild` command to rebuild the B-tree index, re-execute the `adbdbstatus` command to determine whether the uniqueness constraint has been violated.

  For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

(3) Checking the usage of a B-tree index

To check the usage of a B-tree index, execute the following `adbdbstatus` command.

Example of the command to be executed

This example outputs summary information for B-tree index `ADBUSER01.IDX1`.

```
adbdbstatus -d summary -c index -n ADBUSER01.IDX1 -S M --shared-lock
```

Executing the `adbdbstatus` command outputs summary information for B-tree index `ADBUSER01.IDX1` to the standard output.

Check the value output under `MB_Used_pages` in the summary information for the index. You can determine the size of the area (in megabytes) being used by the B-tree index data stored in the DB area.

10.9.4 Checking the status and usage of a text index

When you acquire summary information for index by using the `adbdbstatus` command, you can check the status and usage of a text index.

(1) Checking whether a text index is in unfinished status

To check whether a text index is in unfinished status, execute the following `adbdbstatus` command.

Example of the command to be executed

This example outputs summary information for text index `ADBUSER01.TIDX1`.

```
adbdbstatus -d summary -c index -n ADBUSER01.TIDX1
```

Executing the adbdbstatus command outputs summary information for text index ADBUSER01.TIDX1 to the standard output.

Check the value output under Unfinished in the summary information for the index. If unfinished is output, the text index is in unfinished status.

To release the text index from unfinished status, see 15.10.1 Steps to take when unfinished status is applied to a text index.

(2) Checking the usage of a text index

To check the usage of a text index, execute the following adbdbstatus command.

Example of the command to be executed

This example outputs summary information for text index ADBUSER01.TIDX1.

```
adbdbstatus -d summary -c index -n ADBUSER01.TIDX1 -S M --shared-lock
```

Executing the adbdbstatus command outputs summary information for text index ADBUSER01.TIDX1 to the standard output.

Check the value output under MB_Used_pages in the summary information for the index. You can determine the size of the area (in megabytes) being used by the text index data stored in the DB area.

10.9.5 Checking the status and usage of range indexes

When you acquire summary information for index by using the adbdbstatus command, you can check the status and usage of a range index.

(1) Checking whether a range index is in unfinished status

To check whether a range index is in unfinished status, execute the following adbdbstatus command.

Example of the command to be executed

This example outputs summary information for range index ADBUSER01.RIDX1.

```
adbdbstatus -d summary -c index -n ADBUSER01.RIDX1
```

Executing the adbdbstatus command outputs summary information for range index ADBUSER01.RIDX1 to the standard output.

Check the value output under Unfinished in the summary information for the index. If unfinished is output, the range index is in unfinished status.

To release the range index from unfinished status, see 15.11.1 Steps to take when unfinished status is applied to a range index.

(2) Checking the usage of a range index

To check the usage of a range index, execute the following adbdbstatus command.

Example of the command to be executed

This example outputs summary information for range index ADBUSER01.RIDX1.

```
adbdbstatus -d summary -c index -n ADBUSER01.RIDX1 -S M --shared-lock
```

Executing the adbdbstatus command outputs summary information for range index ADBUSER01.RIDX1 to the standard output.

Check the value output under MB_Used_pages in the summary information for the index. You can determine the size of the area (in megabytes) being used by the range index data stored in the DB area.

(3) Checking the page group size of the range index

To check the page group size of a range index, execute the following adbdbstatus command.

Example of the command to be executed

This example outputs summary information for range index ADBUSER01.RIDX1.

```
adbdbstatus -d summary -c index -n ADBUSER01.RIDX1 -S M --shared-lock
```

Executing the adbdbstatus command outputs summary information for range index ADBUSER01.RIDX1 to the standard output.

Check the value output under Pagegroup_size in the summary information for the index. The value output under Pagegroup_size is the page group size (pages).

10.9.6 Checking the usage of DB area files

When you acquire DB area usage information, you can check the usage of tables and indexes in a DB area and the usage of each DB area file.

Example of the command to be executed

This example outputs usage information for DB area DBAREA01:

```
adbdbstatus -d used -c dbarea -n DBAREA01 -S M --shared-lock
```

Executing this command outputs usage information for DB area DBAREA01 to the standard output. The following are the items to be checked in the results that are output by the adbdbstatus command:

(1) Checking the usage per table or index

Using the values output under Object_type, Schema_name, and Object_identifier as keys, check the value output under MB_Used_pages.

Checking the usage of a table

Check the values output under Schema_name and Object_identifier for a target table, and make sure the Object_type value is table. Using the values output under Schema_name and Object_identifier as keys, check the values output under MB_Used_pages. You can use this information to determine the usage (in megabytes) of the target table.

If the same Schema_name and Object_identifier values exist in multiple rows for the target table, total all of the corresponding values output under MB_Used_pages. The totaled result is the usage of the table.

Checking the usage of an index

Check the values output under `Schema_name` and `Object_identifier` for a target index, and make sure that the `Object_type` value is `index`. Using the values output under `Schema_name` and `Object_identifier` as keys, check the values output under `MB_Used_pages`. You can use this information to determine the usage (in megabytes) of the target index.

If the same `Schema_name` and `Object_identifier` values exist in multiple rows for the target index, total all of the corresponding values output under `MB_Used_pages`. The totaled result is the usage of the index.

> **Note**
>
> When you specify the `--shared-lock` option for the `adbdbstatus` command, segment-related information is not output for the deletion-pending chunks that correspond to system tables (base tables) and the indexes defined in the system tables (base tables).

(2) Checking the usage rate of a DB area file

Using a value output under `DBarea_filename` as a key, check the associated values output under `MB_Total_segments` and `MB_Used_segments`. You can use the following formula to determine the usage rate of each DB area file.

Formula (%)

```
(MB_Used_segments ÷ MB_Total_segments) × 100
```

If the result determined using the formula is close to 100%, consider one of the following corrective measures based on the type of DB area file:

When the DB area file is a block special file

If you continue to store data when the value is close to 100%, the DB area file might become full. Therefore, consider allocating free space on the disk that stores block special files based on the explanation in 15.3.1 When a free space shortage is caused by an increase in the size of the DB area files.

When the DB area file is a regular file

If you continue to store data when the value is close to 100%, the DB area will be extended automatically. When automatic extension occurs, the size of the DB area will automatically increase, but the performance of update processing while DB area file expansion is underway will decline. If DB area automatic extension fails, see 15.3.2 When a free space shortage is caused by failed DB area automatic extension.

For the system-table DB area file, also consider that you execute the adbreorgsystemdata command, in addition to take the preceding actions. By executing the adbreorgsystemdata command to reorganize the system table, you can secure free disk space. For details about reorganizing the system table, see 11.17 Reorganizing system tables.

(3) Checking the usage of a DB area file

Using the value output under `DBarea_filename` as the key, check the value output under `MB_Used_segments`. You can determine the size of the area (in megabytes) that is being used to store data in each DB area file.

10.9.7 Checking the size of archive files

By executing the `adbdbstatus` command to acquire the usage information of an archived chunk, you can check the size of archive files.

Example of the command to be executed

The usage information of an archived chunk for archivable multi-chunk table `ADBUSER01.T1` is output.

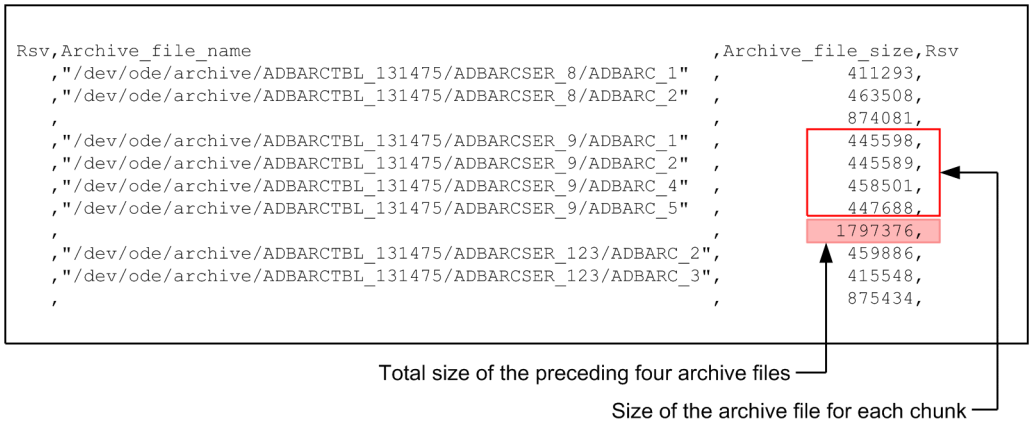
```
adbdbstatus -d used -c archivechunk -n ADBUSER01.T1
```

Check the following items in the output result.

▪ **Checking the size of archive files**

In the `Archive_file_size` column, the sizes of individual archive files are output. In a row in which nothing is output in the `Archive_file_name` column, the total size of all archive files that store data in the chunk is output.

Example:



10.10 Performing statistical analysis (checking HADB server operation information)

By executing the `adbstat` command, you can output the following types of HADB server operation information:

- The HADB server's statistical information
- Connection operation information
- Global buffer statistical information
- SQL statement statistical information

This section explains how to use these types of information to check the HADB server's operation information.

This section also provides details about the statistics log file into which the HADB server's operation information is output.

For details about the `adbstat` command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

10.10.1 Using the HADB server's statistical information

In the HADB server's statistical information, you can check the server's overall memory usage and its operational status.

For example, if you want to change the value of an operand specified in the server definition, you can use the HADB server's statistical information.

(1) Outputting the HADB server's statistical information and acquiring related information

When you execute the `adbstat` command, the HADB server's statistical information is output. This output information displays differences from previously output information. Therefore, if you want to check the HADB server's operational status in a given time period, we recommend that you output its statistical information on a regular basis.

■ Outputting the HADB server's statistical information

See *Checking the operational status of the entire system* in *Examples* under *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

We also recommend that you acquire the following types of related information when you output the HADB server's statistical information:

- **Server message log files**

  Use the OS's `cp` command or a similar method to copy and acquire these files. For the storage destination of server message log files, see 10.4.2 Viewing the message logs (message log output destination).

(2) Using the HADB server's statistical information and related information

By acquiring the HADB server's statistical information and related information on a regular basis, you can check the changes in the HADB server's operational status. You can also use the acquired HADB server's statistical information and related information to perform tuning.

■ **Using the HADB server's statistical information and related information**

1. Check the output date and time of the HADB server's statistical information.

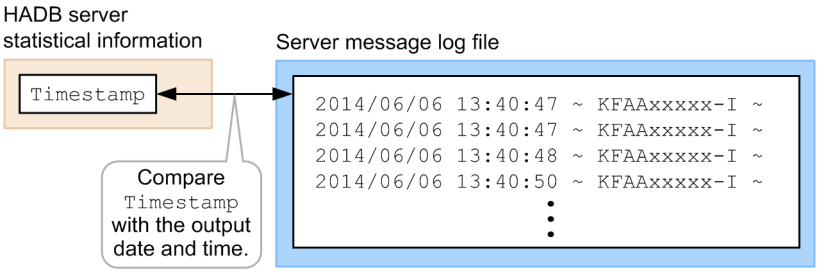
   Check the `Timestamp` date and time (the output date and time of the HADB server's statistical information) that is output under the HADB server's statistical information.

2. Compare the `Timestamp` date and time with the output dates and times in the server message log file.

   Compare the `Timestamp` date and time you checked in step 1 with the output dates and times of the messages in the server message log file. By comparing the `Timestamp` date and time with the output dates and times of the messages, you can determine what type of processing the HADB server was performing.

The following figure shows the concept of using the HADB server's statistical information and related information.

Figure 10-8: Using the HADB server's statistical information and related information



You can also use the HADB server's statistical information and related information when you perform the following types of tuning:

- 13.1.5 Preventing SQL statement execution wait status from occurring
- 13.1.11 Changing the file configuration of the work table DB area
- 13.4.1 Reducing the usage of shared memory to which HugePages is applied

10.10.2 Using the connection operation information

In the connection operation information, you can check the operational status of each connection made to the HADB server.

For example, if you want to change the value of an operand specified in the server definition, or if you want to change the execution method of an application, command, or SQL statement, you can use the connection operation information.

(1) Outputting the connection operation information and acquiring related information

When you execute the `adbstat` command, the connection operation information is output. The output information displays differences from previously output information. Therefore, if you want to check the operational status of each connection made to the HADB server in a given time period, we recommend that you output the connection operation information on a regular basis while you are executing applications, commands, or SQL statements.

■ **Outputting the connection operation information**

See *Checking the operational status of a specific application program* in *Examples* under *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

We also recommend that you acquire the following types of related information when you output the connection operation information:

- **Server message log files**

  Use the OS's `cp` command or a similar method to copy and acquire these files. For the storage destination of server message log files, see 10.4.2 Viewing the message logs (message log output destination).

- **Application identifier**

  Use the `adbls -d cnct` command to check the application identifier that is set for the application or command being executed. In the case of an application, the application identifier that was specified in the `adb_clt_ap_name` operand in the client definition is output.

  For details about the `adbls -d cnct` command, see *adbls -d cnct (Display the Connection Status)* in the manual *HADB Command Reference*.

  For details about the `adb_clt_ap_name` operand in the client definition, see *Operands related to application program status monitoring* in the *HADB Application Development Guide*.

  > **Note**
  >
  > If there are multiple applications, we recommend that you specify for each application a unique application identifier in the `adb_clt_ap_name` operand in the client definition.

(2) Using the connection operation information and related information

By acquiring the connection operation information and related information on a regular basis, you can check the operational status of each connection made to the HADB server. You can also use the acquired connection operation information and related information to perform tuning.

■ Using the connection operation information and related information

1. Check the connection operation information.

   In the connection operation information that is output, check the following information:

   - `Timestamp` (the output date and time of the connection operation information)
   - `AP_name` (application identifier)

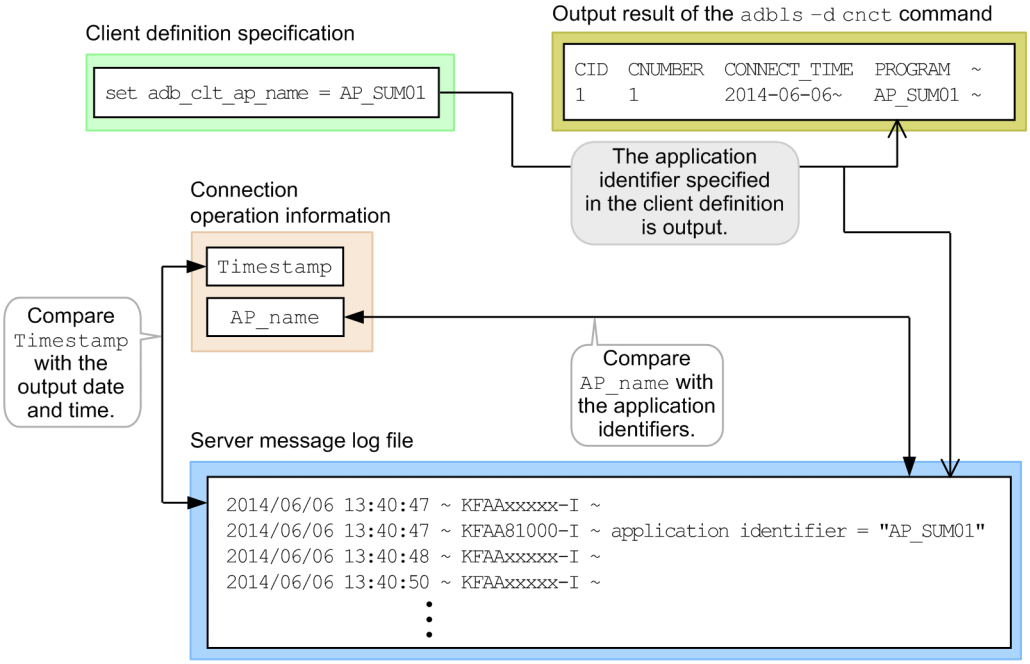
2. Compare the `Timestamp` date and time with the server message log file.

   Compare the `Timestamp` date and time that you checked in step 1 with the output dates and times of the messages in the server message log file. By comparing the `Timestamp` date and time with the output dates and times of the messages, you can determine what type of processing the target connection was performing.

   Also, compare the `AP_name` information, which you checked in step 1, with the application identifier in message `KFAA81000-I` in the server message log file. By comparing the `AP_name` information with the application identifier, you can check the target application and command.

The following figure shows the concept of using the connection operation information and related information.

Figure 10-9: Using the connection operation information and related information



You can also use the connection operation information and related information when you perform the following types of tuning:

- 13.1.6 Reducing the SQL statement processing time
- (1) Using the connection operation information to reduce the execution time of an SQL statement that creates a local work table in 13.2.4 Reducing the execution time of an SQL statement that creates a local work table.

10.10.3 Using the global buffer statistical information

In the global buffer statistical information, you can check the usage status of each global buffer.

For example, if you want to change the value of an operand specified in the server definition, or if you want to change a data DB area, table, or index, you can use the global buffer statistical information.

(1) Outputting the global buffer statistical information and acquiring related information

When you execute the `adbstat` command, the global buffer statistical information is output. The output information covers the entire period from the time the HADB server was started. If you want to check the usage status of the global buffer, we recommend that you output the global buffer statistical information.

■ Outputting the global buffer statistical information

See *Checking the operational status of a specific global buffer* in *Examples* under *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

We also recommend that you acquire the following types of related information when you output the global buffer statistical information:

- **Server message log files**

Use the OS's `cp` command or a similar method to copy and acquire these files. For the storage destination of server message log files, see 10.4.2 Viewing the message logs (message log output destination).

- **Global buffer name**

  Execute the `adbls -d gbuf` command to check the global buffer name. The global buffer name that was specified in the `-g` option of the `adbbuff` operand in the server definition is output.

  For details about the `adbls -d gbuf` command, see *adbls -d gbuf (Display Global Buffer Information)* in the manual *HADB Command Reference*. For details about the `adbbuff` operand in the server definition, see 7.2.11 Operands and options related to global buffers (command format).

  > **Note**
  >
  > You can also check the global buffer name using one of the following methods:
  >
  > - Check the message log when the HADB server starts, and determine the global buffer name from the server definition information in the output message `KFAA50027-I`.
  > - Reference the server definition file (`$ADBDIR/conf/server.def`) that was used when the HADB server was started, and determine the global buffer name.

(2) Using the global buffer statistical information and related information

By acquiring the global buffer statistical information and related information, you can check the usage status of each global buffer. You can also use the acquired global buffer statistical information and related information to perform tuning.

■ Using the global buffer statistical information and related information

1. Check the global buffer statistical information.

   In the global buffer statistical information that is output, check the following information:

   - `Timestamp` (the output date and time of the global buffer statistical information)
   - `DBbuff_name` (global buffer name)

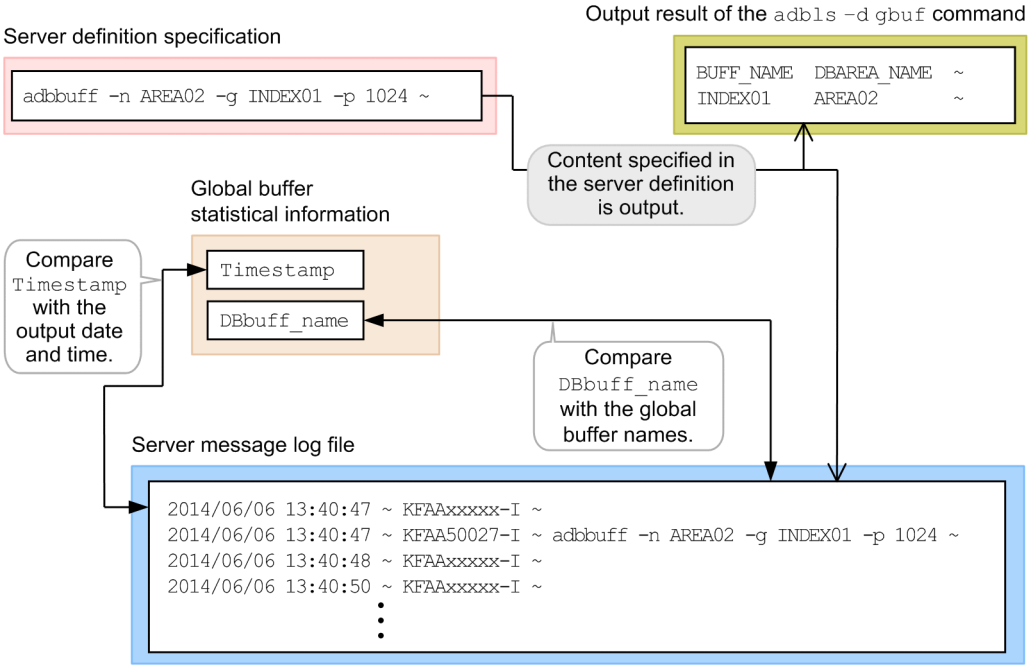
2. Compare the `Timestamp` date and time with the server message log file.

   Compare `Timestamp` checked in step 1 and the output date and time of the message in the server message log file. By comparing `Timestamp` and the output date and time, you can check how the applicable global buffer was used for processing.

   Also, compare `DBbuff_name` checked in step 1 and the global buffer name in the `KFAA50027-I` message in the server message log file. By comparing `DBbuff_name` and the global buffer name, you can check the applicable global buffer name.

The following figure shows the concept of using the global buffer statistical information and related information.

Figure 10-10: Using the global buffer statistical information and related information



You can also use the global buffer statistical information and related information when you perform the following types of tuning:

- (1) Using the global buffer statistical information to reduce SQL statement execution time in 13.2.2 Reducing the SQL statement execution time.
- (1) Using the global buffer statistical information to reduce the execution time of an SQL statement that creates a global work table in 13.2.3 Reducing the execution time of an SQL statement that creates a global work table.
- (1) Using the global buffer statistical information to reduce the execution time of an SQL statement that performs table scans in 13.2.5 Reducing the execution time of SQL statements that perform table scans.
- 13.3.2 Reducing the execution time of the adbidxrebuild command
- 13.3.3 Reducing the execution time of the adbgetcst command
- 13.3.4 Reducing the execution time of the adbexport command
- 13.3.5 Reducing the execution time of the adbmergechunk command
- 13.3.6 Reducing the execution time of the adbarchivechunk command

10.10.4 Using the SQL statement statistical information

In the SQL statement statistical information, you can check the operational status of each SQL statement.

For example, if you want to change the value of an operand specified in the server definition, or if you want to change the SQL statement to execute, or change a table or index, you can use the SQL statement statistical information.

Note

You can determine the operational status of SQL statements by checking the SQL trace information that has been output to SQL trace files, as well as by checking the SQL statement statistical information that

has been output by the adbstat command. For details about the SQL trace information, see 10.11 Running SQL tracing.

(1) Outputting the SQL statement statistical information and acquiring related information

When an SQL statement is finished executing, SQL statement statistical information is output to the statistics log file. By executing the adbstat command, you can display the SQL statement statistical information that was output to the statistics log file. Note that SQL statement statistical information might not be output for SQL statements that result in an error.

We recommend that you output the SQL statement statistical information on a regular basis.

■ Outputting the SQL statement statistical information

See *Checking the SQL statement statistical information in a specific date and time range* in *Examples* under *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

We also recommend that you acquire the following types of related information when you output the SQL statement statistical information:

- **Server message log files**

  Use the OS's cp command or a similar method to copy and acquire these files. For the storage destination of server message log files, see 10.4.2 Viewing the message logs (message log output destination).

  > **Tip**
  >
  > When Y is specified for the adb_sql_text_out operand in the server definition, all SQL statements that were accepted by the HADB server are output to the server message log file. To make it easier to identify a certain SQL statement when you check the SQL statement statistical information and server message log files, consider specifying Y for the adb_sql_text_out operand in the server definition. For details about the adb_sql_text_out operand in the server definition, see 7.2.5 Operands related to SQL statements (set format).

- **Application identifier**

  Use the adbls -d cnct command to check the application identifier that is set for the application or command being executed. In the case of an application, the application identifier that was specified in the adb_clt_ap_name operand in the client definition is output.

  For details about the adbls -d cnct command, see *adbls -d cnct (Display the Connection Status)* in the manual *HADB Command Reference*.

  For details about the adb_clt_ap_name operand in the client definition, see *Operands related to application program status monitoring* in the *HADB Application Development Guide*.

  > **Note**
  >
  > If there are multiple applications, we recommend that you specify for each application a unique application identifier in the adb_clt_ap_name operand in the client definition.

- **Application connection information**

  Use the adbls -d cnct command to check the application connection information.

For details about the `adbls -d cnct` command, see *adbls -d cnct (Display the Connection Status)* in the manual *HADB Command Reference*.

- **SQL trace files**

  Use a command such as the OS's `cp` command to copy SQL trace files. For the storage destination of SQL trace files, see 10.11.1 About SQL tracing.

> **Note**
>
> For details about how to output SQL trace information to SQL trace files, see 10.11.5 Preparations for outputting SQL trace information.

(2) Using the SQL statement statistical information and related information

By acquiring the SQL statement statistical information and related information on a regular basis, you can check the operational status of each SQL statement. You can also use the acquired SQL statement statistical information and related information to perform tuning.

■ Using the SQL statement statistical information and related information

1. Check the SQL statement statistical information.

   In the SQL statement statistical information that is output, check the following types of information:

   - `Timestamp` (output date and time of statistical log file information)
   - `AP_name` (application identifier)
   - `Connection_information` (connection information)
   - `SQL_serial_number` (sequential number assigned to SQL statements starting from when a connection is established)

2. Compare the information with the server message log file.

   Compare the information that you checked in step 1 with the information in the server message log file.

   - **`Timestamp`**

     Compare the `Timestamp` that you checked in step 1 with the output date and time of the message in the server message log file. By comparing the `Timestamp` with the output date and time, you can determine what type of processing the target SQL statement was performing.

   - **`AP_name`**

     Compare `AP_name`, which you checked in step 1, with the application identifier in message `KFAA81000-I` in the server message log file. By comparing `AP_name` with the application identifier, you can check the target application and command.

   - **`Connection_information`**

     Compare `Connection_information`, which you checked in step 1, with the connection information in message `KFAA81000-I` in the server message log file. By comparing `Connection_information` with the connection information, you can check the target application's connection.

   - **`SQL_serial_number`**

     When `Y` is specified for the `adb_sql_text_out` operand in the server definition, compare `SQL_serial_number`, which you checked in step 1, with the SQL statements' sequential numbers assigned starting from when the connection was established, which are indicated in message `KFAA51000-I` in the server

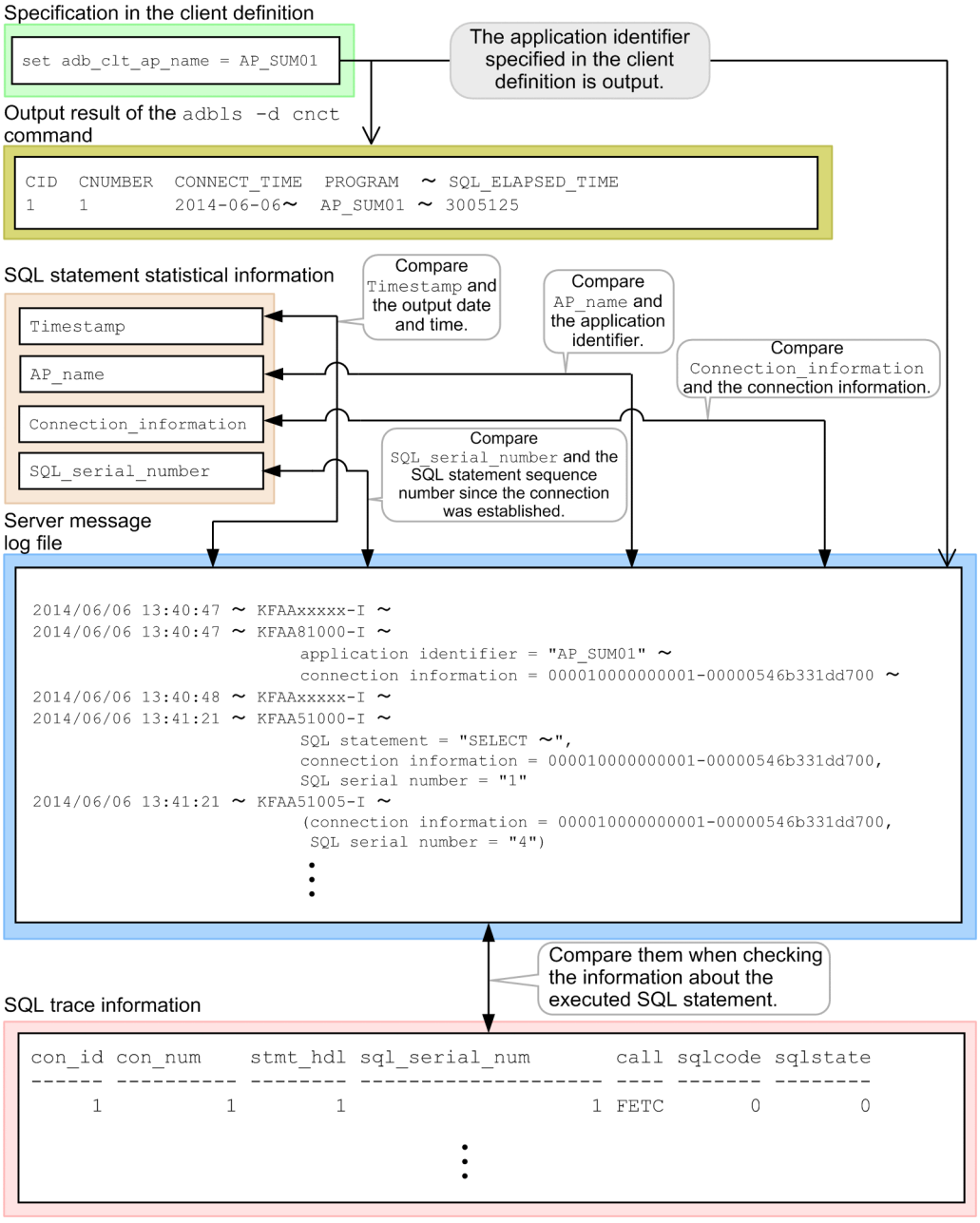
message log file. By comparing `SQL_serial_number` with the SQL statements' sequential numbers assigned starting from when the connection was established, you can check the target SQL statement.

For an SQL statement that terminated in an error, compare `SQL_serial_number`, which you checked in step 1, with the SQL statement's sequential number assigned starting from when the connection was established, which is indicated in message `KFAA51005-I` in the server message log file.

To obtain information about executed SQL statements, check the information in server message log files against the SQL trace information.

The following figure shows the concept of using the SQL statement statistical information and related information.

Figure 10-11: Using the SQL statement statistical information and related information



You can also use the SQL statement statistical information and related information to perform the following types of tuning:

- (2) Using the SQL statement statistical information to reduce SQL statement execution time in 13.2.2 Reducing the SQL statement execution time.
- (2) Using the SQL statement statistical information to reduce the execution time of an SQL statement that creates a global work table in 13.2.3 Reducing the execution time of an SQL statement that creates a global work table.
- (2) Using the SQL statement statistical information to reduce the execution time of an SQL statement that creates a local work table in 13.2.4 Reducing the execution time of an SQL statement that creates a local work table.
- (2) Using the SQL statement statistical information to reduce the execution time of an SQL statement that performs table scans in 13.2.5 Reducing the execution time of SQL statements that perform table scans.

10.10.5 Using the statistics log files

(1) Information that is output to statistics log files

The following HADB server operation information is output to statistics log files:

- SQL statement statistical information

You can check the information that is output to statistics log files by executing the `adbstat` command.

The HADB server automatically creates and uses four statistics log files.

> **Important**
>
> Do not manually update the created statistics log files. Otherwise, the information that is output to the statistics log files might become corrupt.

You can change the maximum size for one statistics log file by using the `adb_sta_log_max_size` and `adb_sta_log_size_unit` operands in the server definition. For details about the `adb_sta_log_max_size` and `adb_sta_log_size_unit` operands in the server definition, see the following sections in 7.2.7 Operands related to statistical information (set format):

- adb_sta_log_max_size operand
- adb_sta_log_size_unit operand

When a statistics log file reaches the maximum size, the HADB server stops outputting logs to that statistics log file and outputs them to the next statistics log file. During this process, any old information that was stored in the switching-destination statistics log file is overwritten. Therefore, you need to execute the `adbstat` command periodically before the old information is overwritten if you want to keep the information that has been output to the statistics log files.

> **Note**
>
> - You can check which statistics log file will be used by checking the message `KFAA80295-I`, which is output to the server message log file when the HADB server starts.
> - When a statistics log file reaches the maximum size and is about to be switched to another statistics log file, the message `KFAA80296-I` is output to the server message log file.

- To test a swap between statistics log files, you can change the unit of the maximum size of statistics log files to a smaller size (MB) by specifying the `adb_sta_log_size_unit` operand in the server definition.

(2) Guideline period during which statistical information is not overwritten

You can use the following formula to determine the guideline period during which statistical information in a statistics log file is not overwritten.

Formula (days)

```
guideline-period-during-which-statistical-information-is-not-overwritten = (log_max x 4,294,967,296 ÷ output_info) ÷ 1,440
```

Explanation of variables

- *log_max*: Value specified for the `adb_sta_log_max_size` operand in the server definition (4 is assumed when the operand is omitted)
- *output_info*: Guideline size (bytes) of the statistical information that is output per minute
  Use the following formula to determine its value.

  ```
  690 + 432 x adbbuff_num + 960 x connect_num + 4,376 x sql_num
  ```

- *adbbuff_num*: Number of global buffers defined in the `adbbuff` operand in the server definition (including the number of global buffers automatically defined by the HADB server)

  > **Note**
  >
  > For details about the global buffers automatically defined by the HADB server, see the explanation about the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

- *connect_num*: Number of concurrent connections to the HADB server
- *sql_num*: Number of executed SQL statements per minute

Calculation example

The following shows a calculation example of the guideline period during which statistical information in a statistics log file is not overwritten. In this example, the following conditions are assumed:

- Value specified for the `adb_sta_log_max_size` operand: 4
- Number of global buffers defined in the `adbbuff` operand: 128
- Number of concurrent connections to the HADB server: 128
- Number of executed SQL statements per minute: 120

Guideline size of the statistical information that is output per minute (*output_info*)

= 690 + 432 x 128 + 960 x 128 + 4,376 x 120

= 703,986 (bytes)

Guideline period during which statistical information is not overwritten

= (4 x 4,294,967,296 ÷ 703,986) ÷ 1,440

≈ approximately 17 days

Under the preceding conditions, the period during which statistical information is not overwritten is approximately 17 days.

(3) Changing the output destination of statistics log files

Statistics log files are output to the `$ADBDIR/spool` directory. However, you can change the output-destination directory of statistics log files by specifying the `adb_sta_log_path` operand in the server definition. For details about the `adb_sta_log_path` operand in the server definition, see 7.2.7 Operands related to statistical information (set format).

Important

If you change the output-destination directory of statistics log files, the statistical information stored in the old statistics log files (that were used before the output destination was changed) cannot be referred. Therefore, before you change the output-destination directory of statistics log files, execute the `adbstat` command to output statistical information.

(4) Deleting statistics log files

The following describes the procedure for deleting statistics log files.

Important

When you delete statistics log files, be sure to follow the following procedure. If you do not follow the procedure, the information that is output to the statistics log file might become corrupt.

Note

After you perform an operation test of the HADB server, the statistics log files might be unnecessary, and you might want to delete them. In such a case, make sure that you perform the following procedure and delete the statistics log files.

Procedure:

1. Execute the `adbstop` command to terminate the HADB server normally.

   If you are using the multi-node function, perform this step on all nodes on which you want to delete the statistics log files.

   If you are using a cold standby configuration, terminate the cold standby configuration normally.
2. Use the `rm` command of the OS to delete the files under the output-destination directory for statistics log files.

   Delete all the following files:

   - `adbstatlog`*XX*

     XX in the file name is a sequential number between `01` and `04`.
   - `.adbstatlog`

     This file is a hidden file.
3. Execute the `adbstart` command to start the HADB server normally.

10.10.6 Estimating the size of information that is output from the statistics log files when the adbstat command is executed

This subsection explains how to estimate the size of information that will be output from the statistics log files when the `adbstat` command is executed.

Use the following formula to determine the size of the CSV file that will be output as the SQL statement statistical information (variable *SQL_CSV_SIZE*).

Formula (kilobytes)

Determine the size by assuming that information in all statistics log files will be output using the `adbstat` command.

$$SQL\_CSV\_SIZE = \frac{STLOG\_SUM}{(0.81 + 0.22 \times (DBAREA\_NUM - 1)) \times 2} \times (1.37 + 0.69 \times (DBAREA\_NUM - 1))$$

Explanation of variables

STLOG_SUM

Combined total size of all statistics log files (kilobytes)

DBAREA_NUM

Number of DB areas in the entire HADB server

10.11 Running SQL tracing

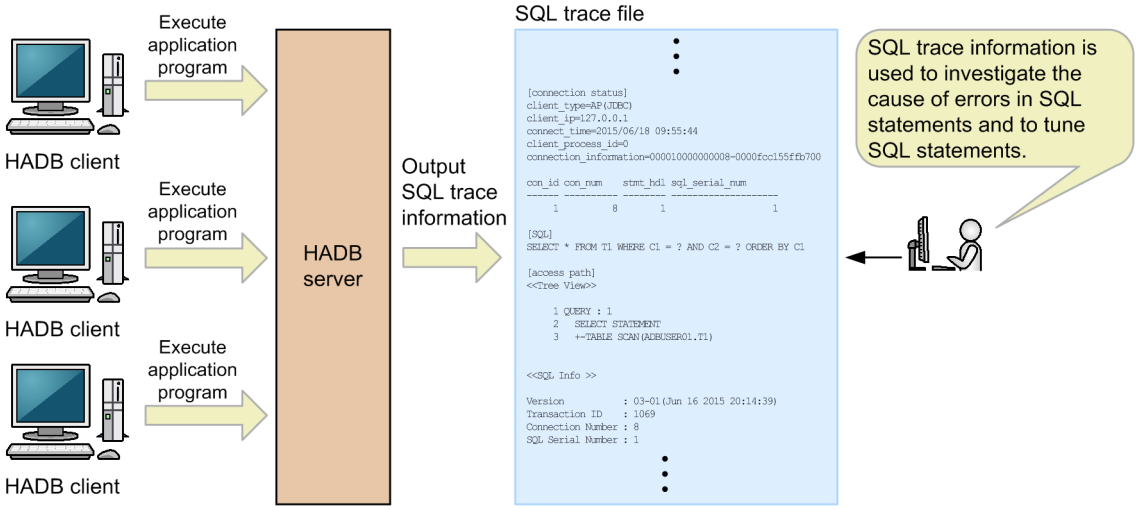
This section explains how to run SQL tracing and the SQL trace information that is output.

10.11.1 About SQL tracing

SQL tracing is a function that outputs to files (SQL trace files) log information (SQL trace information) about execution of SQL statements. The information that is output to the SQL trace files includes the SQL statements that were executed, the SQL statement execution times, the access paths and so on.

The following figure provides an overview of SQL tracing.

Figure 10-12: Overview of SQL tracing



Explanation

SQL trace information is output to SQL trace files on the HADB server. You use a text editor to view SQL trace information that has been output to SQL trace files.

Only HADB administrators and the OS users that belong to the HADB administrators group can view SQL trace files.

The SQL trace information that has been output is used for the following purposes:

- Determining the cause of SQL statement errors

  Check the `SQLCODE` number that has been output as SQL trace information to determine from the message that corresponds to the `SQLCODE` number the cause of an SQL statement error.

- Tuning SQL statements

  Check the SQL statement execution times that are output as SQL trace information to identify SQL statements whose execution time took too long. Use SQL statement statistical information to tune SQL statements based on the connection information and SQL statement sequence numbers that are output as SQL trace information.

■ SQL trace information output destination

- Name of output destination directory

  `$ADBDIR/spool`

- Names of SQL trace files

adbsqltrc01.log to adbsqltrc08.log

Eight SQL trace files are created. When one SQL trace file becomes full, the output destination is changed to the SQL trace file with the next sequence number. When adbsqltrc08.log becomes full, the output destination is changed back to adbsqltrc01.log.

Note

The troubleshooting information that is acquired by the adbinfoget command includes the SQL trace files.

10.11.2 Information that is output as SQL trace information

The following information is output as SQL trace information:

- HADB server version information
- SQL statement execution information
- Client-definition information
- Connection status information
- Executed SQL statement and access path information
- Dynamic parameter information
- Transaction information
- Authorization identifier information
- SQL statement statistical information
- Access path statistical information
- Line delimiting the establishment or termination of a connection
- SQL statement basic information

Note

If SQL statement processing stops due to an error, any SQL trace information that could not be acquired at that point is not output.

The following subsections explain the SQL trace information items that are output.

(1) HADB server version information

The following shows an example of HADB server version information that is output:

```
Hitachi Advanced Data Binder 03-01 Aug  5 2015 18:32:45
```

Explanation

- The underlined portion is the HADB server's version.
- Following the underlined portion, additional version information is output in the following format:

MMM DD YYYY hh:*mm*:*ss*

MMM: Month (three-letter abbreviation of the month name)

DD: Date

YYYY: Calendar year

hh:*mm*:*ss*: Time

> **Note**
>
> The HADB server version information is output at the beginning of an SQL trace file.

(2) SQL statement execution information

This subsection explains an example of the SQL statement execution information that is output and the items that are included in the output.

(a) Example of output of SQL statement execution information

The following shows an example of SQL statement execution information that is output:

```
con_id con_num    stmt_hdl sql_serial_num       call sqlcode sqlstate
------ ---------- -------- -------------------- ---- ------- --------
     1          1        1                    1 FETC       0 00000
start_time                 end_time                  exe_time(us)         rows
-------------------------- ------------------------- -------------------- --------------------
2015/07/31 21:48:46.736799 2015/07/31 21:48:46.737725                  926                    1
tran_id              trn_iso_lv      trn_access_mode sql_order_mode cursor_holdability       message_log_info
-------------------- --------------- --------------- -------------- ----------------------- --------------------
                4342 READ_COMMITTED  READ_WRITE      ISO            CLOSE_CURSORS_AT_COMMIT  00000054070000000001
```

(b) Items that are output as SQL statement execution information

The following table lists and describes the items that are output as SQL statement execution information.

Table 10-10: Items that are output as SQL statement execution information

No.	Item name	Description
1	`con_id`	Connection ID
2	`con_num`	Connection sequence number since the HADB server started.
3	`stmt_hdl`	Statement handle assigned to the SQL statement.
4	`sql_serial_num`	SQL statement sequence number since the connection was established.
5	`call`	Call type subject to SQL statement execution information. Table 10-11: Correspondence between call types and processes that were performed describes the correspondence between call types that are output and processes that were performed. A *call* is the unit of connection establishment or termination, transaction normal termination or cancellation, or database operation using an SQL statement. When SQL trace information is set to be output for each SQL statement, the SQL statement execution information is output when execution of an SQL statement (execution of `CLOS`, `EXEC`, and `EXDI` for each call) is completed. In such a case, `SQL` is displayed in the `call` column.
6	`sqlcode`	`SQLCODE`. For details about how `SQLCODE` corresponds to messages, see *Interpreting SQLCODEs* in the manual *HADB Messages*.
7	`sqlstate`	`SQLSTATE`.

No.	Item name	Description
		For details about SQLSTATE, see *List of SQLSTATE values* in the manual *HADB Messages*.
8	start_time	Processing start time. When SQL trace information is set to be output for each call, this item outputs the start time of each call. When SQL trace information is set to be output for each SQL statement, this item outputs the start time of the first call described in Table 10-12: Calls that are executed when the call type is SQL.
9	end_time	Processing termination time. When SQL trace information is set to be output for each call, the termination time of each call is output. When SQL trace information is set to be output for each SQL statement, the termination time of the first call executed is output.
10	exe_time(us)	Processing execution time in microseconds. • When SQL trace information is set to be output for each call The difference between the start time and the end time of each call is output. If an implicit commit or rollback occurs during the call processing, that processing time is included in this execution time. For the execution time of FETC for the second or any subsequent fetch operation, the sum of the execution times of all fetch operations is output, not the difference between the start time and the end time that is output as the same SQL statement execution information. • When SQL trace information is set to be output for each SQL statement The sum of the execution times of all calls is output.
11	rows	Number of rows subject to processing. This information is output when the call type is FETC, EXEC, EXDI, or SQL. • When the call type is FETC For the first fetch operation, 0 or 1 is output. For a subsequent fetch operation, the number of rows fetched during the second through the last fetch operations is output. • When the call type is EXEC or EXDI If the SQL statement executed is DELETE, INSERT, or UPDATE, the number of rows that were updated is output. If the SQL statement executed is other than these, 0 is output. • When the call type is SQL If the SQL statement executed is SELECT, DELETE, INSERT, or UPDATE, the number of rows subject to processing is output. If the SQL statement executed is other than these, 0 is output.
12	tran_id	Transaction ID, which is a sequence number assigned since the database was initialized
13	trn_iso_lv	Transaction isolation level. • READ_COMMITTED: Output when the transaction isolation level is READ COMMITTED. • REPEATABLE_READ: Output when the transaction isolation level is REPEATABLE READ.
14	trn_access_mode	Transaction access mode. • READ_WRITE: Output when the transaction access mode is the read/write mode. • READ_ONLY: Output when the transaction access mode is the read-only mode.
15	sql_order_mode	Sort order for character string data. • BYTE: Output when character string data is sorted by bytecode. • ISO: Output when character string data is sorted by sort code.
16	cursor_holdability	Cursor operation. • CLOSE_CURSORS_AT_COMMIT: Output when the cursor is closed during the commit operation. • HOLD_CURSORS_OVER_COMMIT: Output when the cursor is not closed during the commit operation. The cursor operation is output in the following cases: • The call type is OPEN, FETC, or CLOS. • The call type is SQL and the executed SQL statement is the SELECT statement.
17	message_log_info	Message log information.

No.	Item name	Description
		This is the client process ID and connection sequence number for a message that is output to the message log file. In the event of an SQL statement error, you can determine the cause of the error by checking the message log information against this client process ID and connection sequence number displayed in the message. For details, see 10.11.7 Using SQL trace information to determine the cause of errors in SQL statements.

Notes:

- When the call type is `CNCT`, `DISC`, or `DISP`, no values are output for `stmt_hdl`, `sql_serial_num`, `rows`, `tran_id`, `trn_iso_lv`, `trn_access_mode`, and `sql_order_mode`.
- When the call type is `CMIT` or `RLBK`, no values are output for `stmt_hdl`, `sql_serial_num`, and `rows`.
- If a statement handle satisfies any of the following conditions, no values are output for `call`, `sqlcode`, `sqlstate`, `start_time`, `end_time`, `exe_time`, and `rows` (this does not apply to a statement handle resulting in an error when the transaction was settled due to an error in the SQL statement):

  • The call immediately preceding the subject statement handle resulted in an error.

  • The type of the call immediately preceding the subject statement handle was `OPEN` (the unit of SQL trace information output is a call).

  • The type of the call immediately preceding the subject statement handle was `FETC` and a fetch operation was performed only once (the unit of SQL trace information output is a call).

Table 10-11: Correspondence between call types and processes that were performed

No.	Call type#1 (information output to call)	Processing that was performed	Unit of output#2	Additional information that is output#3
1	`PREP`	Preprocessing of SQL statements	Call	2, 4
2	`OPEN`	Cursor open processing		3, 4
3	`CLOS`	Cursor close processing		4, 6
4	`FETC`	First fetch operation or the second through the last fetch operations#4		4
5	`EXEC`	SQL statement execution#5		3, 4, 6
6	`EXDI`	Preprocessing and execution of SQL statements		2, 4, 6
7	`CMIT`	Commit processing	Call or SQL statement	4
8	`RLBK`	Rollback processing		4
9	`CNCT`	Connection establishment (for application programs and commands)		1, 4, 5, 7
10	`DISC`	Disconnection		4, 5, 7
11	`DISP`	Disconnection when using the connection pool function		4
12	`SQL`	Output by SQL statement#6, #7		2, 3, 4, 6

#1

For information about the correspondence between call types and executed JDBC methods, see (d) Correspondence between executed JDBC methods and the call types that are output.

For information about the correspondence between call types and executed ODBC functions, see (e) Correspondence between executed ODBC functions and the call types that are output.

For information about the correspondence between call types and executed CLI functions, see (f) Correspondence between executed CLI functions and the call types that are output.

#2

This is the output unit of the SQL trace information specified in the `adb_sql_trc_level` server definition operand or the `adbchgsqltrc` command.

#3

This is the information that is output in addition to the SQL statement execution information. The numbers in the table correspond to the following:

1: Client-definition information and connection status information

2: Executed SQL statement and access path information

3: Dynamic parameter information

4: Transaction information

5: Authorization identifier information

6: SQL statement statistical information and access path statistical information

7: Line delimiting the establishment or termination of a connection

#4

If the call type is `FETC`, this does not mean that as many sets of SQL trace information are output as there are fetches. If the fetch count is 2 or greater, only two sets of SQL statement execution information (one set for the first fetch operation and one set for all the subsequent fetch operations combined) are output regardless of how many times a fetch operation was performed. If the fetch count is 1, only one set of SQL statement execution information is output.

#5

If batch transfer of dynamic parameter values is performed, only one set of SQL statement execution information is output for the `EXEC` SQL statement. For details about batch transfer of dynamic parameter values, see *Batch transfer of dynamic parameter values* in the *HADB Application Development Guide*.

#6

If you execute the `adbexport` command by specifying an SQL statement file, information will be output as the execution information about a single SQL statement even if the output level is `CALL`. In this case, `SQL` is output as the call type.

#7

When SQL trace information is to be output for each SQL statement, one set of SQL statement execution information is output for each of the calls listed in the *Calls that are executed* column in Table 10-12: Calls that are executed when the call type is SQL. In this case, `SQL` is output as the call type.

Table 10-12: Calls that are executed when the call type is SQL

No.	Type of SQL statement	Calls that are executed[#1]
1	Update SQL statement	`PREP` → `EXEC`
2		`EXEC` (when `EXEC` is executed immediately after the call in No. 1 was executed)
3		`EXDI`
4	Definition SQL statement	`PREP` → `EXEC`
5		`EXDI`
6	Retrieval SQL statement	`PREP` → `OPEN` → `FETC` → [`FETC`] → `CLOS` (when the cursor is closed explicitly)
7		`PREP` → `OPEN` → `FETC` → [`FETC`] (when the cursor is not closed explicitly[#2])

Note:

A call type enclosed in square brackets ([]) might not be executed.

#1

If an error occurs during execution of an SQL statement, SQL statement execution information is output as shown in (c) Output of SQL statement execution information in the event of an error during execution of an SQL statement (when the call type is SQL).

#2

This applies when a statement handle was released or a transaction was settled without closing the cursor. This does not apply when the `cursor_holdability` (cursor operation) explained in Table 10-10: Items that are output as SQL statement execution information is `HOLD_CURSORS_OVER_COMMIT`, because the cursor is not closed even when commit processing is performed.

(c) Output of SQL statement execution information in the event of an error during execution of an SQL statement (when the call type is SQL)

If an error occurs during execution of an SQL statement, one set of SQL statement execution information is output for each of the following calls:

- Call resulting in the error
- Of the calls shown in Table 10-12: Calls that are executed when the call type is SQL, all the calls that were executed before the call that resulted in the error

If the only type of call resulting in an error is `FETC`, the processing depends on whether implicit rollback occurred when the error occurred:

- Implicit rollback occurred when the error occurred.

  Among the calls shown in Table 10-12: Calls that are executed when the call type is SQL, a set of SQL statement execution information is output for the `FETC` that caused the error and for the calls that preceded it.

- Implicit rollback did not occur.

  `FETC` resulting in the error is not treated as the completion of the retrieval SQL statement execution. Execution of the retrieval SQL statement is regarded as having been completed when the corresponding `CLOS` processing is completed. In this case, a set of SQL statement execution information is output for each call shown as a retrieval SQL statement in Table 10-12: Calls that are executed when the call type is SQL.

The following are examples of when SQL statements result in an error:

- When retrieval SQL statements were executed, `OPEN` resulted in an error.

  SQL statement execution information for `PREP` and `OPEN` is combined into one set, and one set of SQL statement execution information is output for each SQL statement.

- When retrieval SQL statements were executed, `FETC` resulted in an error and implicit rollback occurred.

  SQL statement execution information for `PREP`, `OPEN`, [`FETC`], and `FETC` is combined into one set, and one set of SQL statement execution information is output for each SQL statement.

- When update SQL statements were executed, `PREP` resulted in an error.

  SQL statement execution information for `PREP` is output as a set of SQL statement execution information for each SQL statement.

(d) Correspondence between executed JDBC methods and the call types that are output

The following table shows the correspondence between executed JDBC methods and the call types that are output.

Table 10-13: Correspondence between executed JDBC methods and the call types that are output

No.	Executed JDBC method	Call type that is output
1	`connect(String url, Properties info)` of the `Driver` interface	`CNCT`
2	`close()` of the `Connection` interface	`CMIT`, `DISC`, `DISP`
3	`commit()` of the `Connection` interface	`CMIT`
4	`prepareStatement(String sql)` of the `Connection` interface	`PREP`
5	`prepareStatement(String sql, int resultSetType, int resultSetConcurrency)` of the `Connection` interface	
6	`prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)` of the `Connection` interface	
7	`rollback()` of the `Connection` interface	`RLBK`
8	`close()` of the `Statement` interface	`CLOS`, `CMIT`
9	`execute(String sql)` of the `Statement` interface	`PREP`, `OPEN`, `EXEC`, `EXDI`, `CLOS`, `CMIT`, `RLBK`
10	`executeBatch() or executeLargeBatch()` of the `Statement` interface	`PREP`, `EXEC`, `EXDI`, `CMIT`, `RLBK`
11	`executeQuery(String sql)` of the `Statement` interface	`PREP`, `OPEN`, `CLOS`, `CMIT`
12	`executeUpdate(String sql)` or `executeLargeUpdate(String sql)` of the `Statement` interface	`PREP`, `EXEC`, `EXDI`, `CMIT`, `RLBK`
13	`getMoreResults()` of the `Statement` interface	`CLOS`
14	`execute()` of the `PreparedStatement` interface	`PREP`, `OPEN`, `EXEC`, `CLOS`, `CMIT`, `RLBK`
15	`executeQuery()` of the `PreparedStatement` interface	`PREP`, `OPEN`, `CLOS`, `CMIT`, `RLBK`
16	`executeUpdate()` or `executeLargeUpdate()` of the `PreparedStatement` interface	`PREP`, `EXEC`, `CMIT`, `RLBK`
17	`absolute(int row)` of the `ResultSet` interface	`FETC`
18	`afterLast()` of the `ResultSet` interface	
19	`beforeFirst()` of the `ResultSet` interface	
20	`first()` of the `ResultSet` interface	
21	`isAfterLast()` of the `ResultSet` interface	
22	`isBeforeFirst()` of the `ResultSet` interface	
23	`ilast()` of the `ResultSet` interface	
24	`irelative(int rows)` of the `ResultSet` interface	
25	`isLast()` of the `ResultSet` interface	
26	`next()` of the `ResultSet` interface	

No.	Executed JDBC method	Call type that is output
27	close() of the ResultSet interface	CLOS, CMIT
28	getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern) of the DatabaseMetaData interface	PREP, OPEN
29	getCrossReference(String primaryCatalog, String primarySchema, String primaryTable, String foreignCatalog, String foreignSchema, String foreignTable) of the DatabaseMetaData interface	
30	getExportedKeys(String catalog, String schema, String table) of the DatabaseMetaData interface	
31	getImportedKeys(String catalog, String schema, String table) of the DatabaseMetaData interface	
32	getIndexInfo(String catalog, String schema, String table, boolean unique, boolean approximate) of the DatabaseMetaData interface	
33	getPrimaryKeys(String catalog, String schema, String table) of the DatabaseMetaData interface	
34	getSchemas(String catalog, String schemaPattern) of the DatabaseMetaData interface	
35	getTablePrivileges(String catalog, String schemaPattern, String tableNamePattern) of the DatabaseMetaData interface	
36	getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types) of the DatabaseMetaData interface	
37	getSchemas() of the DatabaseMetaData interface	
38	getConnection() of the DataSource interface	CNCT
39	getConnection(String username, String password) of the DataSource interface	
40	close() of the PooledConnection interface	DISC
41	getConnection() of the PooledConnection interface	CNCT

(e) Correspondence between executed ODBC functions and the call types that are output

The following table shows the correspondence between executed ODBC functions and the call types that are output.

Table 10-14: Correspondence between executed ODBC functions and the call types that are output

No.	Executed ODBC function	Call type that is output
1	SQLPrepare(W)	PREP
2	SQLExecute	PREP, EXEC, OPEN, CMIT
3	SQLExecDirect(W)	PREP, EXEC, CMIT
4	SQLParamData	EXEC, OPEN, CMIT
5	SQLFetch	FETC

No.	Executed ODBC function	Call type that is output
6	SQLColumns(W)	PREP, OPEN
7	SQLForeignKeys(W)	
8	SQLPrimaryKeys(W)	
9	SQLStatistics(W)	
10	SQLTablePrivileges(W)	
11	SQLTables(W)	
12	SQLFreeStmt	CLOS, CMIT
13	SQLCloseCursor	
14	SQLEndTran	CMIT, RLBK
15	SQLDisconnect	CMIT, DISC
16	SQLFreeHandle	CMIT
17	SQLSetConnectAttr(W)	
18	SQLConnect(W)	CNCT, DISC
19	SQLDriverConnect(W)	
20	SQLBrowseConnect(W)	

(f) Correspondence between executed CLI functions and the call types that are output

The following table shows the correspondence between executed CLI functions and the call types that are output.

Table 10-15: Correspondence between executed CLI functions and the call types that are output

No.	Executed CLI function	Call type that is output
1	a_rdb_SQLPrepare()	PREP
2	a_rdb_SQLExecute()	OPEN, EXEC
3	a_rdb_SQLCloseCursor()	CLOS
4	a_rdb_SQLFetch()	FETC
5	a_rdb_SQLExecDirect()	EXDI
6	a_rdb_SQLEndTran()	CMIT, RLBK
7	a_rdb_SQLConnect()	CNCT
8	a_rdb_SQLDisconnect()	DISC

(3) Client-definition information

The following shows an example of client-definition information that is output.

■ Example of output of client-definition information

```
[client definitions]
adb_clt_ap_name="SAMPLE"
adb_clt_fetch_size=1024
adb_clt_group_name="test_group"
```

```
adb_clt_sql_order_mode="ISO"
adb_clt_sql_text_out="Y"
adb_clt_trn_access_mode="READ_WRITE"
adb_clt_trn_iso_lv="READ_COMMITTED"
adb_dbbuff_wrktbl_clt_blk_num=256
adb_sql_exe_hashflt_area_size=200
adb_sql_exe_hashgrp_area_size=4800
adb_sql_exe_hashtbl_area_size=2000
adb_sql_exe_max_rthd_num=0
adb_sql_prep_dec_div_rs_prior="FRACTIONAL_PART"
adb_sql_prep_delrsvd_use_srvdef="Y"
```

If the connection was established by an application program, the client-definition information that has been applied to that application program is output. If the connection was established by a command, no client-definition information is output.

Each operand value output to the client-definition information is the value applied to the application program, not the value specified in the client definition. For example, if the `adb_sql_exe_max_rthd_num` operand value specified in the client definition was changed to the default value by the HADB server because the specified value was invalid, the value assumed by the HADB server is output.

Note

Client-definition information is output in the following cases:

- The call type is `CNCT` and `CNCT` terminated normally.
- The first transaction was started after the `adbchgsqltrc -s` command was executed and SQL trace information was output (the client-definition information for the connection that started the transaction is output).

(4) Connection status information

The following shows an example of the connection status information that is output.

■ Example of output of the connection status information (when the connection was established by an application program)

```
[connection status]
client_type=AP(JDBC)
client_ip=127.0.0.1
connect_time=2015/07/31 21:54:41
client_process_id=0
connection_information=000010000000001-0000181584fb6700
```

■ Example of output of connection status information (when the connection was established by a command)

```
[connection status]
client_type=command(adbimport)
client_ip=127.0.0.1
connect_time=2015/07/31 16:10:37
client_process_id=30422
connection_information=000010000000003-000073b601e46700
```

The following table lists and describes the items that are output as connection status information.

Table 10-16: Items that are output as connection status information

No.	Item name	Description
1	client_type	Type of HADB client. • For accesses from JDBC drivers: AP(JDBC) • For accesses from programs other than the JDBC drivers: AP(C library) • For commands: command(*command-name*) Example: command(adbimport)
2	client_ip	IP address of the HADB client
3	connect_time	Time when the connection was established
4	client_process_id	Process ID of the HADB client. If a JDBC drive is used, 0 is output.
5	connection_information	Connection information (the information that is output for connection information in the KFAA81000-I message)

Note

Connection status information is output in the following cases:

- The call type is CNCT and CNCT terminated normally.
- The first transaction was started after the adbchgsqltrc -s command was executed and SQL trace information was output (the connection status information for the connection that started the transaction is output).

(5) Executed SQL statement and access path information

The following shows an example of executed SQL statement and access path information that is output.

■ Example of output of executed SQL statement and access path information

```
[SQL]
SELECT * FROM "T1" WHERE "C1"=? AND "C2"=? AND "C3"=?

[access path]
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T1)


<<SQL Info >>

Version           : 03-01(Jul 31 2015 21:31:44)
Transaction ID    : 4342
Connection Number : 1
SQL Serial Number : 1
```

Explanation

- [SQL]

  Outputs the executed SQL statement.

Asterisks (*) are output for the password portion of the ALTER USER or CREATE USER statement.

The executed SQL statement is output as is even if it contains unprintable special characters.

- [access path]

  Outputs the access path information. For details about the items that are output as access path information, see *How to use access paths (how to use SQL statement execution plans)* in the *HADB Application Development Guide*.

> **Note**
>
> The executed SQL statement and the access path information are output in the following cases:
>
> - The call type is PREP.#
> - The call type is EXDI.
> - The call type is SQL.
>
> #
>
> The access path information is output when PREP terminates normally. If PREP did not terminate normally, only the executed SQL statement is output.

(6) Dynamic parameter information

The following shows an example of dynamic parameter information that is output.

■ Example of output of dynamic parameter information

```
[param]
param_no type      len1  len2  data
-------- --------- ----- ----- ----
       1 int           8       1
       2 char          2       'ab'
       3 dec           7     3 1234.567
```

The following table lists and describes the items that are output as dynamic parameter information.

Table 10-17: Items that are output as dynamic parameter information

No.	Item name	Description
1	param_no	Sequence number of the dynamic parameter in the SQL statement
2	type	Data type of the dynamic parameter
3	len1	Parameter length
4	len2	Parameter length attribute. If there is no parameter length attribute, this item is not output. For details about parameter lengths and parameter length attributes, see *a_rdb_SQLDescribeParams() (acquire dynamic parameter information)* in the *HADB Application Development Guide*.
5	data	Data bound to the dynamic parameter

The following table shows the output format for the dynamic parameter information.

Table 10-18: Output format for the dynamic parameter information

No.	Data type	type	len1	len2	Output format
1	INTEGER	int	8	None	Decimal
2	DECIMAL(*m*,*n*) #1	dec	*m*	*n*	Decimal
3	SMALLINT	smallint	4	None	Decimal
4	DOUBLE PRECISION	double	8	None	*Mantissa*E*exponent* The mantissa and exponent are both output in decimal.
5	CHARACTER(*n*)	char	*n*	None	'*character-string*'
6	VARCHAR(*n*)	varchar	*n*	None	'*character-string*' If the actual length is 0, '' is output.
7	DATE	date	4	None	Predefined output representation for date data#2
8	TIME(*p*) #3	time	3 + ↑*p* ÷ 2↑	*p*	Predefined output representation for time data#2
9	TIMESTAMP(*p*) #3	timestamp	7 + ↑*p* ÷ 2↑	*p*	Predefined output representation for time stamp data#2
10	BINARY(*n*)	binary	*n*	None	X'*hexadecimal-value*'
11	VARBINARY(*n*)	varbinary	*n*	None	X'*hexadecimal-value*' If the actual length is 0, X'' is output.
12	ROW	row	Row length	None	X'*hexadecimal-value*'

Legend:

m, *n*, and *p*: Positive integer

#1

If the data type is DECIMAL, len1 acquires the precision. len2 acquires scaling.

#2

For details about the predefined output representations, see *Predefined character-string representations* in the manual *HADB SQL Reference*.

#3

If the data type is TIME or TIMESTAMP, len1 acquires the data length. len2 acquires the number of digits in the fractional seconds.

■ Output rules for dynamic parameter information

- If bound data is the null value, NULL is output.
- The dynamic parameter information is output as is even when it contains unprintable special characters.
- If only a dynamic parameter is specified in the NULL predicate and data that is not the null value is bound, nothing is output in the data column. If the null value is bound, NULL is output in the data column.
- If batch transfer of dynamic parameter values is performed, the headers (param_no, type, len1, len2, data, and separator line) and data are output for each dynamic parameter set. Note that [param] is output only once per batch transfer.

  For details about batch transfer of dynamic parameter values, see *Batch transfer of dynamic parameter values* in the *HADB Application Development Guide*.

> **Note**
>
> Dynamic parameter information is output when both the following conditions are satisfied:
>
> - A dynamic parameter is specified in the SQL statement.
> - The call type is `OPEN`, `EXEC`, or `SQL`.

(7) Transaction information

The following shows an example of transaction information that is output.

■ Example of output of transaction information

```
tran_id              start_time                 end_time                   exe_time(us)
-------------------- -------------------------- -------------------------- --------------------
                4342 2015/07/31 21:48:40.537485 2015/07/31 21:48:49.711265              9173780
trn_iso_lv      trn_access_mode sql_order_mode
--------------- --------------- --------------
READ_COMMITTED  READ_WRITE      ISO
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

The following table lists and describes the items that are output as transaction information.

Table 10-19: Items that are output as transaction information

No.	Item name	Description
1	`tran_id`	Transaction ID, which is a sequence number assigned since the database was initialized
2	`start_time`	Transaction start time
3	`end_time`	Transaction end time
4	`exe_time(us)`	Transaction execution time in microseconds. This is the difference between the transaction start time and the transaction end time.
5	`trn_iso_lv`	Transaction isolation level. • `READ_COMMITTED`: Output when the transaction isolation level is `READ COMMITTED`. • `REPEATABLE_READ`: Output when the transaction isolation level is `REPEATABLE READ`.
6	`trn_access_mode`	Transaction access mode. • `READ_WRITE`: Output when the transaction access mode is read/write mode. • `READ_ONLY`: Output when the transaction access mode is read-only mode.
7	`sql_order_mode`	Sort order for character string data. • `BYTE`: Output when character string data is sorted by bytecode. • `ISO`: Output when character string data is sorted by sort code.
8	~	The swung dash (~) is output as the separator line character for transaction information.

> **Note**
>
> Transaction information is output in the following cases:
>
> - The call type is `CMIT` or `RLBK`.

- An implicit transaction settlement occurred.#

#

An implicit commit is executed when a definition SQL statement, TRUNCATE TABLE statement, or PURGE CHUNK statement terminates normally.

If the SQL statement results in an error, an implicit rollback might be executed. If this happens, the KFAA51001-E message will be output to the message log file.

An implicit commit is also executed when the connection is terminated during transaction processing.

(8) Authorization identifier information

The following shows an example of authorization identifier information that is output.

■ Example of output of authorization identifier information

When the call type is CNCT:

```
ADBUSER01 has been connected.
```

When the call type is DISC:

```
ADBUSER01 has been disconnected.
```

Explanation

The underlined portion is the authorization identifier that was specified when the connection was established by the application program.

If the connection was established by a command, the authorization identifier specified in the -u option of the command is output. If the -u option was omitted, no authorization identifier information is output even if the command connects to the HADB server.

Note

Authorization identifier information is output in the following cases:

- The call type is CNCT and CNCT terminated normally.
- The call type is DISC and DISC terminated normally.

(9) SQL statement statistical information

The following shows an example of SQL statement statistical information that is output. The example is in two parts because it consists of a large amount of data.

■ Example of output of SQL statement statistical information (part 1)

```
[SQL statistics]
Type HADB_system_version Timestamp                  AP_name                         Connection_information
---- ------------------- ------------------------- ------------------------------- ------------------------------
SQL  03-01               2015-07-31 21:48:46.738029 SAMPLE                          000010000000001-000014f70d8e1700
SQL  03-01               2015-07-31 21:48:46.738029 SAMPLE                          000010000000001-000014f70d8e1700
SQL  03-01               2015-07-31 21:48:46.738029 SAMPLE                          000010000000001-000014f70d8e1700
Connect_time              SQL_serial_number    SQL_type       SQL_total_time       Fetch_row_cnt        Update_row_cnt
------------------------- -------------------- -------------- -------------------- -------------------- --------------------
2015-07-31 21:48:37.590351                    1 SELECT                       1007                    1                    0
2015-07-31 21:48:37.590351                    1 SELECT
2015-07-31 21:48:37.590351                    1 SELECT
Hashgrp_area_max_size Hashgrp_area_get_cnt Hashtbl_area_max_size Lock_dbarea_request_cnt Lock_dbarea_wait_cnt Lock_dbarea_wait_time
--------------------- -------------------- --------------------- ----------------------- -------------------- ---------------------
                    0                    0                     0                       0                    0                     0


Lock_table_request_cnt Lock_table_wait_cnt  Lock_table_wait_time DBbuff_dbarea_information_num DBbuff_information_num
---------------------- -------------------- -------------------- ------------------------------ ----------------------
                    13                    0                    0                              1                      1
                                                                                                2                      1
                                                                                                3                      2
DBbuff_dbarea_name             DBbuff_name                                 DBbuff_page_request_cnt DBbuff_page_hit_cnt
------------------------------ ------------------------------------------- ----------------------- --------------------
ADBDIC                         ##ADBOTHER#0000004096                                            38                   23
ADBSTBL                        ##ADBOTHER#0000004096                                             8                    4
ADBUTBL01                      TBLBUF01                                                          5                    3
DBbuff_page_hit_rate DBbuff_page_put_cnt  DBbuff_page_read_cnt DBbuff_page_write_cnt DBbuff_page_rng_request_cnt
-------------------- -------------------- -------------------- --------------------- ----------------------------
                  61                    0                   15                     0                            0
                  50                    0                    4                     0                            0
                  60                    0                    2                     0                            0
DBbuff_page_rng_hit_cnt DBbuff_page_rng_hit_rate DBbuff_page_rng_put_cnt DBbuff_page_rng_read_cnt DBbuff_page_rng_write_cnt
----------------------- ------------------------ ------------------------ ------------------------ -------------------------
                      0                        0                        0                        0                         0
                      0                        0                        0                        0                         0
                      0                        0                        0                        0                         0
DBbuff_tblscan_request_cnt DBbuff_tblscan_hit_cnt DBbuff_tblscan_hit_rate DBbuff_tblscan_read_cnt DBbuff_tblscan_failed_cnt
-------------------------- ---------------------- ---------------------- ---------------------- -------------------------
                         0                      0                      0                      0                         0
                         0                      0                      0                      0                         0
                         0                      0                      0                      0                         0
DBbuff_tblscan_insufficient_buff_num DBbuff_tblscan_reset_cnt DBbuff_wrktbl_clt_request_cnt DBbuff_wrktbl_clt_hit_cnt
------------------------------------ ------------------------ ------------------------------ -------------------------
                                   0                        0                              0                        0
                                   0                        0
                                   0                        0
DBbuff_wrktbl_clt_hit_rate DBbuff_wrktbl_clt_put_cnt DBbuff_wrktbl_clt_read_cnt DBbuff_wrktbl_clt_write_cnt DBbuff_wrktbl_clt_tbl_cnt
-------------------------- ------------------------- ------------------------- -------------------------- -------------------------
                         0                         0                         0                          0                         0


DBbuff_wrktbl_clt_sort_merge_cnt DBbuff_wrktbl_clt_sort_page_max_cnt Log_usrbuf_out_cnt   Log_usrfile_max_size DBarea_extension_cnt
-------------------------------- ---------------------------------- -------------------- -------------------- --------------------
                               0                                  0                    0                    0                    0


Log_usrfile_write_cnt DBbuff_wrktbl_clt_request_fix_cnt DBbuff_wrktbl_clt_pagein_fix_cnt DBbuff_wrktbl_clt_pageout_fix_cnt
--------------------- --------------------------------- --------------------------------- ---------------------------------
                    0                                 0                                 0                                 0


DBbuff_wrktbl_clt_max_used_blk_num Directory_send_num   Directory_recv_num   Log_send_num         Log_recv_num
---------------------------------- -------------------- -------------------- -------------------- --------------------
                                 0                    0                    0                    0                    0
```

■ Example of output of SQL statement statistical information (part 2)

```
DBbuff_send_num      DBbuff_recv_num      Node_com_num         Node_com_time        Bidx_page_split_cnt  Tidx_page_split_cnt
-------------------- -------------------- -------------------- -------------------- -------------------- --------------------
                   0                    0                    0                    0                    0                    0


Bidx_validation_check_cnt Ridx_sgmt_skip_cnt   Ridx_chunk_skip_cnt  Ridx_chunk_read_cnt  Ridx_chunk_judge_cnt Ridx_sgmt_read_cnt
------------------------- -------------------- -------------------- -------------------- -------------------- --------------------
                       21                    0                    0                    0                    0                    0


Ridx_sgmt_judge_cnt  SegmentRel_rthd_max_num Csvread_file_cnt     Csvread_file_read_size Hashgrp_area_shortage
-------------------- ----------------------- -------------------- ---------------------- --------------------
                   0                       0                    0                      0 N


Hashgrp_area_sufficient_size Hashtbl_area_shortage Syndict_file_access_time Auditread_file_cnt   Auditread_file_read_size
---------------------------- -------------------- ------------------------ -------------------- ------------------------
                           0 N                                            0                    0                        0


DBbuff_tblscan_read_size DBbuff_tblscan_rthd_min_size DBbuff_tblscan_rthd_max_size DBbuff_tblscan_use_size
------------------------ ---------------------------- ---------------------------- ------------------------
                       0                            0                            0                        0
                       0                            0                            0                        0
                       0                            0                            0                        0


DBbuff_tblscan_insufficient_buff_size Hashflt_disabled Tbldef_req_cnt       Tbldef_access_cnt    Tbldef_cache_access_cnt
------------------------------------- ---------------- -------------------- -------------------- ----------------------
                                    0 N                                    1                    1                      0
                                    0
                                    0


Tbldef_cache_register_cnt Tbldef_cache_sweep_cnt DBbuff_page_wait_cnt Max_sql_rthd_num     Hashtbl_area_size    Hashflt_area_size
------------------------- ------------------------ -------------------- -------------------- -------------------- --------------------
                        1                        0                    0                    4                 2000                  200
                                                                      0
                                                                      0
```

Explanation

The items that are output as SQL statement statistical information are the same as for the SQL statement statistical information for the `adbstat` command. For details about the items that are output, see *Items that are output as SQL statement statistical information* in the manual *HADB Command Reference*.

> **Note**
>
> SQL statement statistical information is output in the following cases:
>
> - The call type is `CLOS` (including when the statement handle was released or the transaction was concluded without closing the cursor).
> - The call type is `EXEC`, `EXDI`, or `SQL`.

(10) Access path statistical information

For an example of output of and output items for access path statistical information, see 10.11.3 Examples of output of and output items for access path statistical information.

(11) Line delimiting the establishment or termination of a connection

255 asterisks (`*`) are output as a line indicating where the connection was established or terminated.

■ Example of a line denoting connection establishment (when the call type is `CNCT`)

```
********************************************************************************************************************************
********************************************************************************************************************************
*****************************
con_id con_num    stmt_hdl sql_serial_num       call sqlcode sqlstate
------ ---------- -------- -------------------- ---- ------- --------
     1          1                               CNCT       0 00000
start_time                 end_time                   exe_time(us)         rows
-------------------------- -------------------------- -------------------- --------------------
2015/07/31 21:48:37.501670 2015/07/31 21:48:37.612007               110337
tran_id            trn_iso_lv      trn_access_mode sql_order_mode cursor_holdability       message_log_info
------------------ --------------- --------------- -------------- ------------------------ --------------------
                                                                                            0000005407000000001
```

■ Example of a line denoting connection termination (when the call type is DISC)

```
con_id con_num    stmt_hdl sql_serial_num       call sqlcode sqlstate
------ ---------- -------- -------------------- ---- ------- --------
     1          1                               DISC       0 00000
start_time                 end_time                   exe_time(us)         rows
-------------------------- -------------------------- -------------------- --------------------
2015/07/31 21:48:49.688406 2015/07/31 21:48:49.713685                25279
tran_id            trn_iso_lv      trn_access_mode sql_order_mode cursor_holdability       message_log_info
------------------ --------------- --------------- -------------- ------------------------ --------------------
                                                                                            0000005407000000001

********************************************************************************************************************************
********************************************************************************************************************************
*****************************
```

Note

A line delimiting the establishment or termination of a connection is output in the following cases:

- The call type is CNCT.
- The call type is DISC and DISC terminated normally.

(12) SQL statement basic information

When SQL trace information is output for an SQL statement, basic information about the SQL statement is output. The following shows an example of SQL statement basic information that is output.

■ Example of output of SQL statement basic information

```
con_id con_num    stmt_hdl sql_serial_num
------ ---------- -------- --------------------
     1          1        1                    1

[SQL]
SELECT * FROM T1 WHERE C1=? AND C2=? AND C3=?

[access path]
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   +-TABLE SCAN(ADBUSER01.T1)


<<SQL Info >>

Version           : 03-01(Jul 31 2015 21:31:44)
Transaction ID    : 4346
Connection Number : 1
SQL Serial Number : 1
```

SQL statement basic information

Explanation

- `con_id`
  Outputs the connection ID.
- `con_num`
  Outputs the connection sequence number since the HADB server started.
- `stmt_hdl`
  Outputs the statement handle assigned to the SQL statement.
- `sql_serial_num`
  Outputs the SQL statement sequence number since the connection was established.

10.11.3 Examples of output of and output items for access path statistical information

This subsection shows examples of the output and explains the output items for access path statistical information.

Access path statistical information is output as part of SQL trace information when executed SQL statements terminate. Access path statistical information might not be output for SQL statements that result in an error.

You can check the SQL trace information against the access path information by using as a search key the *corresponding access path's tree row number* that is output as access path statistical information. For details about access path information, see (5) Executed SQL statement and access path information in 10.11.2 Information that is output as SQL trace information.

> **Note**
>
> Access path statistical information is output in the following cases:
>
> - The call type is `CLOS` (including when the statement handle was released or the transaction was concluded without closing the cursor).
> - The call type is `EXEC`, `EXDI`, or `SQL`.

(1) Example of output of access path statistical information

The following shows an example of access path statistical information that is output. The example is in two parts because it consists of a large amount of data.

■ Example of output of access path statistical information (part 1)

```
 [access path statistics]
<<Data access information>>
Data_path#           Data_dbbuff_page_request_cnt(table) Data_dbbuff_page_hit_cnt(table)
-------------------- ----------------------------------- -------------------------------
                   3                                   5                               3
Data_dbbuff_page_read_cnt(table) Data_dbbuff_page_write_cnt(table) Data_dbbuff_tblscan_request_cnt
-------------------------------- --------------------------------- -------------------------------
                               2                                 0                               0
Data_dbbuff_tblscan_hit_cnt Data_dbbuff_tblscan_read_cnt Data_dbbuff_tblscan_read_size Data_dbbuff_tblscan_failed_cnt
--------------------------- -------------------------- ---------------------------- ------------------------------
                          0                          0                            0                              0
Data_dbbuff_page_request_cnt(btree) Data_dbbuff_page_hit_cnt(btree) Data_dbbuff_page_read_cnt(btree)
----------------------------------- ---------------------------------- ----------------------------------
                                  0                                  0                                  0
Data_dbbuff_page_write_cnt(btree) Data_dbbuff_page_request_cnt(text) Data_dbbuff_page_hit_cnt(text)
--------------------------------- ---------------------------------- -------------------------------
                                0                                  0                              0
Data_dbbuff_page_read_cnt(text) Data_dbbuff_page_write_cnt(text) Data_dbbuff_page_request_cnt(range)
------------------------------- -------------------------------- ----------------------------------
                              0                                0                                  0
Data_dbbuff_page_hit_cnt(range) Data_dbbuff_page_read_cnt(range) Data_dbbuff_page_write_cnt(range)
------------------------------- -------------------------------- ----------------------------------
                              0                                0                                  0
Data_dbbuff_wrktbl_clt_tbl_cnt Data_dbbuff_wrktbl_clt_request_cnt Data_dbbuff_wrktbl_clt_page_hit_cnt
------------------------------ ---------------------------------- -----------------------------------
                             0                                  0                                   0
Data_dbbuff_wrktbl_clt_read_cnt Data_dbbuff_wrktbl_clt_write_cnt Data_dbbuff_page_request_cnt(glb_wrk)
------------------------------- --------------------------------- ---------------------------------------
                              0                                 0                                      0
Data_dbbuff_page_hit_cnt(glb_wrk) Data_dbbuff_page_read_cnt(glb_wrk) Data_dbbuff_page_write_cnt(glb_wrk)
--------------------------------- ---------------------------------- ----------------------------------
                                0                                  0                                  0
Data_bidx_page_split_cnt Data_bidx_validation_check_cnt Data_ridx_chunk_judge_cnt Data_ridx_chunk_skip_cnt
------------------------ ------------------------------ ------------------------ ------------------------
                       0                              0                        0                        0
Data_ridx_sgmt_judge_cnt Data_ridx_sgmt_skip_cnt Data_tidx_page_split_cnt Data_wrktbl_create_cnt
------------------------ ----------------------- ------------------------ ----------------------
                       0                       0                        0                      0
Data_wrktbl_drop_cnt Data_wrktbl_page_assign_num Data_wrktbl_page_free_num Data_wrktbl_page_use_max
-------------------- ---------------------------- --------------------------- ------------------------
                   0                            0                           0                        0
Data_wrktbl_sort_merge_cnt Data_wrktbl_sort_page_max_cnt Data_dbarea_extension_cnt Data_segmentrel_rthd_max_num
-------------------------- ---------------------------- -------------------------- ----------------------------
                         0                            0                          0                            0
Data_log_usrfile_write_cnt Data_log_usrbuf_out_cnt Data_log_usrfile_max_size Data_directory_send_num
-------------------------- ----------------------- --------------------------- ------------------------
                         0                       0                           0                        0
Data_access_info_type    Data_csvread_file_cnt Data_csvread_file_read_size Data_deleted_rows_cnt Data_tidx_all_search_cnt
------------------------ --------------------- ---------------------------- --------------------- ------------------------
Access                                       0                            0                     0                        0
Data_auditread_file_cnt Data_auditread_file_read_size Data_grpagg_mem_short_cnt Data_grpagg_mem_exceed_limit_cnt
----------------------- ----------------------------- ---------------------------- ---------------------------------
                      0                             0                            0                                 0
Data_grpagg_mem_try_get_cnt Data_grpagg_mem_max_size Data_grpagg_preagg_row_cnt Data_grpagg_rslttbl_reuse_cnt
--------------------------- ------------------------ --------------------------- -----------------------------
                          0                        0                           0                             0
Data_cs_column_invalid_rows_cnt Data_cs_rowstore_fetch_rows_cnt Data_cs_invalid_info_page_request_cnt Data_cs_invalid_false_positive_cnt
------------------------------- ------------------------------- ------------------------------------- ------------------------------------
                              0                               0                                     0                                    0
```

■ Example of output of access path statistical information (part 2)

```
<<Scan information>>
Scan_path#           Scan_row_cnt         Scan_start_time            Scan_end_time
-------------------- -------------------- -------------------------- --------------------------
                   3                    1 2015-07-31 21:48:46.736068 2015-07-31 21:48:46.737075
Scan_info_data_max_num Scan_info_data_max_free_num Scan_info_data_size  Scan_info_inpred_max_num
---------------------- -------------------------- -------------------- ------------------------
                     0                          0                    0                        0
Scan_info_inpred_max_free_num Scan_info_inpred_size Scan_info_quantpred_max_num Scan_info_quantpred_max_free_num
----------------------------- -------------------- ------------------------------ -------------------------------
                            0                    0                              0                               0
Scan_info_quantpred_size Scan_info_drvtbl_max_num Scan_info_drvtbl_max_free_num Scan_info_drvtbl_size Scan_info_hash_max_num
------------------------ ------------------------ ------------------------------ --------------------- ----------------------
                       0                        0                              0                     0                      0
Scan_info_hash_max_free_num Scan_info_hash_size  Scan_info_hashbckt_max_num Scan_info_hashbckt_size
--------------------------- -------------------- --------------------------- ----------------------
                          0                    0                           0                      0
Scan_outerjoin_info_max_num Scan_outerjoin_info_size Scan_info_csvread_file_max_num Scan_info_csvread_file_max_free_num
--------------------------- ------------------------ ------------------------------ -----------------------------------
                          0                        0                              0                                   0
Scan_info_csvread_file_size Scan_info_csvread_data_max_num Scan_info_csvread_data_max_free_num Scan_info_csvread_data_size
--------------------------- ------------------------------ ---------------------------------- ---------------------------
                          0                              0                                  0                           0

<<Set operation information>>

<<Hash grouping area information>>

<<Hash table area information>>

<<Correlated subqueries information>>

<<Correlated subqueries result cache information>>

<<Recursive queries information>>

<<SQL exec information>>
SQLexec_result_wrk_size SQLexec_result_wrk_max_num SQLexec_result_wrk_init_num SQLexec_mem_size     SQLexec_mem_max_num
----------------------- -------------------------- --------------------------- -------------------- --------------------
                     48                          0                           0                    0                    0
SQLexec_mem_init_num SQLexec_dynamic_mem_alloc_unit SQLexec_delayed_free_mem_max_size SQLexec_stack_size
-------------------- ------------------------------ ------------------------------- -------------------
                   0                              0                               0              163840
```

(2) Items that are output as access path statistical information

The following table lists the items that are output as access path statistical information and the types of access paths for which the items are output.

Table 10-20: List of items that are output as access path statistical information

No.	Output item	Type of access path subject to output operation		Details of information that is output
1	Data access information	Work table creation information	CREATE GLOBAL WORK TABLE	See (a) Data access information.
2			CREATE LOCAL WORK TABLE	
3		Table retrieval method information	TABLE SCAN	
4			KEY SCAN	
5			INDEX SCAN	
6		Work table retrieval information	WORK TABLE SCAN	
7		Derived table information	DERIVED TABLE	
8		Table-function derived table information	TABLE FUNCTION DERIVED TABLE	
9		Join method information	HASH JOIN	
10			NESTED LOOP JOIN	

No.	Output item	Type of access path subject to output operation		Details of information that is output
11		Subquery type information	SUBQUERY HASH	
12		Grouping set information	GROUPING SET	
13	Information related to retrieval processing[#1, #2, #3]	Subquery type information	SUBQUERY HASH	See (b) Information related to retrieval processing.
14		Derived table information	DERIVED TABLE	
15		Table-function derived table information	TABLE FUNCTION DERIVED TABLE	
16		Table value constructor retrieval information	TABLE VALUE CONSTRUCTOR SCAN	
17		Grouping processing method information	GLOBAL HASH GROUPING	
18		Duplication elimination method information	GLOBAL HASH UNIQUE	
19		Table retrieval method information	TABLE SCAN	
20			KEY SCAN	
21			INDEX SCAN	
22		Work table retrieval information	WORK TABLE SCAN	
23		Join method information	NESTED LOOP JOIN	
24			HASH JOIN	
25	Information related to set operations[#4]	Set operation type information	SET OPERATION	See (c) Information related to set operations.
26	Information related to hash grouping areas[#5]	Grouping processing method information	LOCAL HASH GROUPING	See (d) Information related to hash grouping areas.
27	Information related to hash table areas	Duplication elimination method information	GLOBAL HASH UNIQUE	See (e) Information related to hash table areas.
28		Subquery type information	SUBQUERY HASH	
29		Grouping processing method information	GLOBAL HASH GROUPING	
30		Join method information	HASH JOIN	
31	Information related to subqueries containing external reference columns[#6,#7]	Subquery type information	SUBQUERY LOOP	See (f) Information related to subqueries containing external reference columns.
32	Information related to caches for storing subquery requests	Subquery cache information	USING CACHE	See (g) Information related to caches for storing subquery requests.
33	Information related to recursive queries	Recursive query information	RECURSIVE	See (h) Information related to recursive queries.

No.	Output item	Type of access path subject to output operation		Details of information that is output
34	Information related to execution of SQL statements	--	--	See (i) Information related to execution of SQL statements.

Legend:

--: When an SQL statement subject to output of access path statistical information is executed, the information is output regardless of the type of access path.

#1

If the retrieval processing is performed within a subquery containing an external reference column whose processing method is nested loops work table execution or nested loops row value execution, the target access path is not output as part of the access path statistical information.

However, the access paths in the following processing that is executed in a subquery that includes an external reference column are output to the access path statistical information:

- Retrieval performed in a subquery that does not include an external reference column

#2

If the retrieval processing is performed on a work table that contains the results of a subquery whose processing method is work table execution, work table row value execution, or nested loops work table execution, the target access path is not output as part of the access path statistical information.

#3

If the retrieval processing is performed on an outer table or an inner table for nested loop join during table join processing to which nested loop join is applied, the target access path is not output as part of the access path statistical information. However, when the outermost table for nested loop join is a derived table, if the retrieval processing is performed for the derived query, the target access path is output as part of the access path statistical information.

#4

If the set operation is performed within a subquery containing an external reference column whose processing method is nested loops work table execution or nested loops row value execution, the target access path is not output as part of the access path statistical information.

However, the access paths in the following processing that is executed in a subquery that includes an external reference column are output to the access path statistical information:

- Set operation performed in a subquery that does not include an external reference column

#5

If local hash grouping is performed within a subquery containing an external reference column whose processing method is nested loops work table execution or nested loops row value execution, the target access path is not output as part of the access path statistical information.

However, the access paths in the following processing that is executed in a subquery that includes an external reference column are output to the access path statistical information:

- Local hash grouping performed in a subquery that does not include an external reference column

#6

If there are multiple incidences of `SUBQUERY LOOP` in the part where subquery results are evaluated, this information is output on the first line.

#7

If `SUBQUERY LOOP` is nested in the tree view of access path information, access path statistical information might not be output for the nested `SUBQUERY LOOP`.

(a) Data access information

The table below explains the data access information that is output as access path statistical information.

In the case of access path statistical information related to update SQL statements, only information about retrieving the data to be updated or deleted is output for the tables subject to update processing. Information about adding, updating, and deleting data is not output.

Multiple lines of data access information might be output for an access path. For details, see Table 10-22: Number of lines and information that are output as data access information for each access path.

Table 10-21: Items that are output as access path statistical information (Data access information)

No.	Column header	Information that is output	Output format	In the event of overflow
1	Data_path#	Corresponding access path's tree row number	UINT8	--
2	Data_dbbuff_page_request_cnt(table)	Page request count of global buffers that are used when base tables are accessed	UINT8	Maximum value
3	Data_dbbuff_page_hit_cnt(table)	Hit count of global buffers that are used when base tables are accessed	UINT8	Maximum value
4	Data_dbbuff_page_read_cnt(table)	Number of read operations from files to global buffers that are used when base tables are accessed	UINT8	Maximum value
5	Data_dbbuff_page_write_cnt(table)	Number of write operations to files from global buffers that are used when base tables are accessed	UINT8	Maximum value
6	Data_dbbuff_tblscan_request_cnt	Page request count of table scan buffers that are used when base tables are accessed	UINT8	Maximum value
7	Data_dbbuff_tblscan_hit_cnt	Page hit count of table scan buffers that are used when base tables are accessed	UINT8	Maximum value
8	Data_dbbuff_tblscan_read_cnt	Number of read operations from files to table scan buffer blocks that are used when base tables are accessed	UINT8	Maximum value
9	Data_dbbuff_tblscan_read_size	Size of data read from files to the table scan buffer that is used when base tables are accessed (kilobytes)	UINT8	Maximum value
10	Data_dbbuff_tblscan_failed_cnt	Allocation failure count of table scan buffer blocks that are used when base tables are accessed When allocation fails, the global buffer is used.	UINT8	Maximum value
11	Data_dbbuff_page_request_cnt(btree)	Page request count of global buffers that are used when B-tree indexes are accessed	UINT8	Maximum value
12	Data_dbbuff_page_hit_cnt(btree)	Hit count of global buffers that are used when B-tree indexes are accessed	UINT8	Maximum value
13	Data_dbbuff_page_read_cnt(btree)	Number of read operations from files to global buffers that are used when B-tree indexes are accessed	UINT8	Maximum value
14	Data_dbbuff_page_write_cnt(btree)	Number of write operations to files from global buffers that are used when B-tree indexes are accessed	UINT8	Maximum value
15	Data_dbbuff_page_request_cnt(text)	Page request count of global buffers that are used when text indexes are accessed	UINT8	Maximum value
16	Data_dbbuff_page_hit_cnt(text)	Hit count of global buffers that are used when text indexes are accessed	UINT8	Maximum value
17	Data_dbbuff_page_read_cnt(text)	Number of read operations from files to global buffers that are used when text indexes are accessed	UINT8	Maximum value

No.	Column header	Information that is output	Output format	In the event of overflow
18	Data_dbbuff_page_write_cnt(text)	Number of write operations to files from global buffers that are used when text indexes are accessed	UINT8	Maximum value
19	Data_dbbuff_page_request_cnt(range)	Page request count of global buffers that are used when range indexes are accessed	UINT8	Maximum value
20	Data_dbbuff_page_hit_cnt(range)	Hit count of global buffers that are used when range indexes are accessed	UINT8	Maximum value
21	Data_dbbuff_page_read_cnt(range)	Number of read operations from files to global buffers that are used when range indexes are accessed	UINT8	Maximum value
22	Data_dbbuff_page_write_cnt(range)	Number of write operations to files from global buffers that are used when range indexes are accessed	UINT8	Maximum value
23	Data_dbbuff_wrktbl_clt_tbl_cnt	Number of times buffers for local work tables were created	UINT8	Maximum value
24	Data_dbbuff_wrktbl_clt_request_cnt	Page request count of buffers for local work tables	UINT8	Maximum value
25	Data_dbbuff_wrktbl_clt_page_hit_cnt	Hit count of buffers for local work tables	UINT8	Maximum value
26	Data_dbbuff_wrktbl_clt_read_cnt	Number of read operations from files to buffers for local work tables	UINT8	Maximum value
27	Data_dbbuff_wrktbl_clt_write_cnt	Number of write operations to files from buffers for local work tables	UINT8	Maximum value
28	Data_dbbuff_page_request_cnt(glb_wrk)	Page request count of global buffers for global work tables	UINT8	Maximum value
29	Data_dbbuff_page_hit_cnt(glb_wrk)	Hit count of global buffers for global work tables	UINT8	Maximum value
30	Data_dbbuff_page_read_cnt(glb_wrk)	Number of read operations from files to global buffers for global work tables	UINT8	Maximum value
31	Data_dbbuff_page_write_cnt(glb_wrk)	Number of write operations to files from global buffers for global work tables	UINT8	Maximum value
32	Data_bidx_page_split_cnt	Value used by the system This is always 0.	UINT8	Maximum value
33	Data_bidx_validation_check_cnt	Number of times whether rows were valid was checked during retrieval using B-tree indexes	UINT8	Maximum value
34	Data_ridx_chunk_judge_cnt	Number of times table chunks were checked during retrieval using range indexes	UINT8	Maximum value
35	Data_ridx_chunk_skip_cnt	Number of times table chunks were skipped during retrieval using range indexes	UINT8	Maximum value
36	Data_ridx_sgmt_judge_cnt	Number of times table segments were checked during retrieval using range indexes	UINT8	Maximum value
37	Data_ridx_sgmt_skip_cnt	Number of times table segments were skipped during retrieval using range indexes	UINT8	Maximum value
38	Data_tidx_page_split_cnt	Value used by the system This is always 0.	UINT8	Maximum value
39	Data_wrktbl_create_cnt	Number of times work tables were created	UINT8	Maximum value
40	Data_wrktbl_drop_cnt	Number of times work tables were deleted	UINT8	Maximum value

No.	Column header	Information that is output	Output format	In the event of overflow
41	Data_wrktbl_page_assign_num	Number of work table pages allocated	UINT8	Maximum value
42	Data_wrktbl_page_free_num	Number of work table pages freed	UINT8	Maximum value
43	Data_wrktbl_page_use_max	Number of pages in the largest work table that was allocated	UINT8	Maximum value
44	Data_wrktbl_sort_merge_cnt	Maximum number of sorting stages in sorting of the work table	UINT8	Maximum value
45	Data_wrktbl_sort_page_max_cnt	Number of pages allocated to the largest work table for sorting work tables	UINT8	Maximum value
46	Data_dbarea_extension_cnt	Value used by the system This is always 0.	UINT8	Maximum value
47	Data_segmentrel_rthd_max_num	Value used by the system This is always 0.	UINT8	Maximum value
48	Data_log_usrfile_write_cnt	Write count of user log files	UINT8	Maximum value
49	Data_log_usrbuf_out_cnt	Flush count resulting from full user log buffers	UINT8	Maximum value
50	Data_log_usrfile_max_size	Largest size of user log files used	UINT8	Maximum value
51	Data_directory_send_num	Value used by the system This is always 0.	UINT8	Maximum value
52	Data_access_info_type	Type of data access • Access Indicates data access information when schema objects subject to access paths are accessed. • Create_hash_wrktbl Indicates data access information when work tables are created due to a shortage of hash table areas. • Scan_hash_wrktbl Indicates data access information when work tables created due to a shortage of hash table areas are retrieved. • Create_split_hash_wrktbl Indicates data access information when a work table# is newly created due to a shortage of hash table areas. • Scan_split_hash_wrktbl Indicates data access information when a work table# newly created due to a shortage of hash table areas is retrieved. • Rowid_fetch Indicates data access information when retrieval processing is performed to fetch data from data pages by using row IDs. • Create_grpset_wrktbl Indicates that the data access information pertains to creation of a work table that is used when grouping is performed multiple times.	CHAR	--

No.	Column header	Information that is output	Output format	In the event of overflow
		• Scan_grpset_wrktbl Indicates that the data access information pertains to retrieval of a work table that is used when grouping is performed multiple times. For the relationship between data access types and access paths, see Table 10-22: Number of lines and information that are output as data access information for each access path.		
53	Data_csvread_file_cnt	Number of files opened by using the ADB_CSVREAD function These files include those opened by using the ADB_CSVREAD function, which is generated by equivalent exchange of the SQL statement used to access the archivable multi-chunk table.	UINT8	Maximum value
54	Data_csvread_file_read_size	Total size of data read from files by using the ADB_CSVREAD function (bytes) This data size includes the size of data read from files by using the ADB_CSVREAD function that is generated by equivalent exchange of the archivable multi-chunk table.	UINT8	Maximum value
55	Data_deleted_rows_cnt	Number of rows that are determined to be invalid rows during retrieval For details about invalid rows, see the following sections: • 11.1.9 Checking whether a single-chunk table needs to be reorganized • 11.4.13 Checking whether a multi-chunk table needs to be reorganized • 11.17.1 Reason for reorganizing a system table	UINT8	Maximum value
56	Data_tidx_all_search_cnt	Number of times when the entire data in a chunk is retrieved by using text index	UINT8	Maximum value
57	Data_auditread_file_cnt	Number of files opened by using the ADB_AUDITREAD function	UINT8	Maximum value
58	Data_auditread_file_read_size	Total size of data read from files by using the ADB_AUDITREAD function (bytes)	UINT8	Maximum value
59	Data_cs_column_invalid_rows_cnt	Number of rows that are determined to be invalid rows during retrieval for column-data segments in a column store table	UINT8	Maximum value
60	Data_cs_rowstore_fetch_rows_cnt	Number of rows found when data in row store format (rows added by using the INSERT or UPDATE statement) is accessed during retrieval in a column store table	UINT8	Maximum value
61	Data_cs_invalid_info_page_request_cnt	Number of page requests for invalid row information pages in a column store table	UINT8	Maximum value

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

CHAR: The character string indicated in *Information that is output* is output. If the target access path was not executed or was being executed, blanks might be output.

Maximum value: In the event of an overflow, 18,446,744,073,709,551,615 is output.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

#

If a shortage of hash table areas occurs while a hash table is created, multiple work tables are created, and data to be stored in the hash table are divided and stored into individual work tables. If another shortage of hash table areas occurs while hash tables for individual work tables are created, a new work table is created. The work table (with hash mark) indicates this type of work table.

The following table shows the number of lines and the information that are output as data access information for each access path.

Table 10-22: Number of lines and information that are output as data access information for each access path

No.	Type of access path	Number of lines and information that are output as data access information	Information that is output as Data_access_info_type
1	CREATE GLOBAL WORK TABLE	**Number of lines output** One line of information is output for the access path. **Output information** Data access information when creating a global work table (including sort processing) is output. This includes the information when creating a local work table at the stage prior to creation of the global work table.	Access
2	CREATE LOCAL WORK TABLE	**Number of lines output** One line of information is output for the access path. **Output information** Data access information when creating a local work table (including sort processing) is output.	Access
3	TABLE SCAN	**Output information (consisting of one line)** Normally, one line of information is output for the access path. The data access information when retrieving target base tables, work tables, or table-function derived tables is output. **Output information (consisting of three lines)** If the access path is one of the following, three lines are output: • Inner table of hash join • Target of the outer query of a subquery to which hash execution is applied The first line displays the data access information when retrieving a target base table, work table, or table-function derived table. The second line displays the data access information when a work table was created due to a shortage of hash table areas. The third line displays the data access information when a work table was retrieved due to a shortage of hash table areas.	**When one line is output** Access **When three lines are output** • Access (Line 1) • Create_hash_wrktbl (Line 2) • Scan_hash_wrktbl (Line 3) **When five lines are output** • Access (Line 1) • Create_hash_wrktbl (Line 2) • Scan_hash_wrktbl (Line 3) • Create_split_hash_wrktbl (Line 4) • Scan_split_hash_wrktbl (Line 5)
4	INDEX SCAN		
5	KEY SCAN		

No.	Type of access path	Number of lines and information that are output as data access information	Information that is output as Data_access_info_type
		Output information (consisting of five lines) If the access path is one of the following, five lines are output: • Outer table of hash join • Target of a subquery to which hash execution is applied • Target of global hash grouping • Target of `SELECT DISTINCT` to which hash execution is applied The first line displays the data access information when retrieving a target base table, work table, or table-function derived table. The second line displays the data access information when a work table was created due to a shortage of hash table areas. The third line displays the data access information when a work table was retrieved due to a shortage of hash table areas. The fourth line displays the data access information when the work table was created due to a shortage of hash table areas after data in a hash table was divided into work tables. The fifth line displays the data access information when the work table was retrieved due to a shortage of hash table areas after data in a hash table was divided into work tables.	
6	`WORK TABLE SCAN`	**Output information (consisting of one line)** Normally, one line of information is output for the access path. The data access information when retrieving target work tables is output. **Output information (consisting of two lines)** If the following applies to the access path, two lines are output: • Storing the row IDs of a table specified in the `FROM` clause for a work table created by an SQL statement with the `ORDER BY` clause specified The first line displays the data access information when retrieving from the target work table. The second line displays the data access information when retrieval processing was performed to fetch data from data pages by using row IDs. If data is retrieved from multiple tables, the data access information is the sum of the values when multiple tables are accessed. **Output information (consisting of three lines)** If the access path is one of the following, three lines are output: • Inner table of hash join • Target of the outer query of a subquery to which hash execution is applied The first line displays the data access information when retrieving from the target work table.	**When one line is output** `Access` **When two lines are output** • `Access` (Line 1) • `Rowid_fetch` (Line 2) **When three lines are output** • `Access` (Line 1) • `Create_hash_wrktbl` (Line 2) • `Scan_hash_wrktbl` (Line 3) **When five lines are output** • `Access` (Line 1) • `Create_hash_wrktbl` (Line 2) • `Scan_hash_wrktbl` (Line 3) • `Create_split_hash_wrktbl` (Line 4) • `Scan_split_hash_wrktbl` (Line 5)

No.	Type of access path	Number of lines and information that are output as data access information	Information that is output as Data_access_info_type
		The second line displays the data access information when a work table was created due to a shortage of hash table areas. The third line displays the data access information when a work table was retrieved due to a shortage of hash table areas. **Output information (consisting of five lines)** If the access path is one of the following, five lines are output: • Outer table of hash join • Target of a subquery to which hash execution is applied • Target of global hash grouping • Target of `SELECT DISTINCT` to which hash execution is applied The first line displays the data access information when retrieving from the target work table. The second line displays the data access information when a work table was created due to a shortage of hash table areas. The third line displays the data access information when a work table was retrieved due to a shortage of hash table areas. The fourth line displays the data access information when the work table was created due to a shortage of hash table areas after data in a hash table was divided into work tables. The fifth line displays the data access information when the work table was retrieved due to a shortage of hash table areas after data in a hash table was divided into work tables.	
7	`DERIVED TABLE`	**Output information (consisting of two lines)** If the access path is one of the following, two lines are output: • Inner table of hash join • Target of the outer query of a subquery to which hash execution is applied The first line displays the data access information when the work table was created due to a shortage of hash table areas. The second line displays the data access information when the work table was retrieved due to a shortage of hash table areas. **Output information (consisting of four lines)** If the access path is one of the following, four lines are output: • Outer table of hash join • Target of a subquery to which hash execution is applied • Target of global hash grouping • Target of `SELECT DISTINCT` to which hash execution is applied • Derived table created for a set operation to which hash execution is applied	**When two lines are output** • `Create_hash_wrktbl` (Line 1) • `Scan_hash_wrktbl` (Line 2) **When four lines are output** • `Create_hash_wrktbl` (Line 1) • `Scan_hash_wrktbl` (Line 2) • `Create_split_hash_wrktbl` (Line 3) • `Scan_split_hash_wrktbl` (Line 4)
8	`TABLE FUNCTION DERIVED TABLE`		
9	`HASH JOIN`		
10	`NESTED LOOP JOIN`		
11	`SUBQUERY HASH`		

No.	Type of access path	Number of lines and information that are output as data access information	Information that is output as Data_access_info_type
		The first line displays the data access information when the work table was created due to a shortage of hash table areas. The second line displays the data access information when the work table was retrieved due to a shortage of hash table areas. The third line displays the data access information when the work table was created due to a shortage of hash table areas after data in a hash table was divided into work tables. The fourth line displays the data access information when the work table was retrieved due to a shortage of hash table areas after data in a hash table was divided into work tables.	
12	GROUPING SET	**Number of lines output** Two lines are output per access path. **Output information** The first line displays the data access information that pertains to creation of a work table that is used when grouping is performed multiple times. The second line displays the data access information that pertains to retrieval of a work table that is used when grouping is performed multiple times.	• Create_grpset_wrktbl (Line 1) • Scan_grpset_wrktbl (Line 2)

(b) Information related to retrieval processing

The table below explains the information related to retrieval processing that is output as access path statistical information.

Table 10-23: Items that are output as access path statistical information (Information related to retrieval processing)

No.	Column header	Information that is output	Output format	In the event of overflow
1	Scan_path#	Corresponding access path's tree row number	UINT8	--
2	Scan_row_cnt	Number of rows retrieved[#1, #5]	UNIT8	WRAP
3	Scan_start_time	Retrieval processing start date and time[#2, #6] The date and time the retrieval processing started are output in the following format: *YYYY-MM-DD*△*hh*:*mm*:*ss*.*nnnnnn*[#4]	CHAR	--
4	Scan_end_time	Retrieval processing end date and time[#3, #7] The date and time the retrieval processing ended are output in the following format: *YYYY-MM-DD*△*hh*:*mm*:*ss*.*nnnnnn*[#4]	CHAR	--

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

CHAR: The character string indicated in *Information that is output* is output. If the target access path was not executed or was being executed, blanks might be output.

WRAP: When the value exceeds 18,446,744,073,709,551,615, it is reset to the minimum value in wraparound mode.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

#1

If the retrieval processing references the recursive query names in recursive queries, the total value for all recursive queries is output.

#2

If the retrieval processing references the recursive query names in recursive queries, the start time of the first recursive query is output.

#3

If the retrieval processing references the recursive query names in recursive queries, the end time of the last recursive query is output.

#4

Four digits are displayed for year (*YYYY*) and two digits are displayed for month (*MM*), date (*DD*), hour (*hh*), minute (*mm*), and second (*ss*). Six digits are displayed for microsecond (*nnnnnn*). A value that consists of fewer digits is padded with leading zeros. Δ indicates a single-byte space.

#5

If multiple DISTINCT set functions are specified with different arguments in the same query specification, grouping is performed multiple times to determine the result of each DISTINCT set function. In this case, the total number of rows retrieved during all grouping processes is output.

#6

If multiple DISTINCT set functions are specified with different arguments in the same query specification, grouping is performed multiple times to determine the result of each DISTINCT set function. In this case, the start date and time for the first grouping process is output.

#7

If multiple DISTINCT set functions are specified with different arguments in the same query specification, grouping is performed multiple times to determine the result of each DISTINCT set function. In this case, the end date and time for the last grouping process is output.

(c) Information related to set operations

The table below explains the information related to set operations that is output as access path statistical information.

Table 10-24: Items that are output as access path statistical information (Information related to set operations)

No.	Column header	Information that is output	Output format	In the event of overflow
1	Setop_path#	Corresponding access path's tree row number	UINT8	--
2	Setop_type	Type of set operation: • UNION ALL • UNION DISTINCT • EXCEPT ALL • EXCEPT DISTINCT • INTERSECT ALL	CHAR	--

No.	Column header	Information that is output	Output format	In the event of overflow
		• INTERSECT DISTINCT This might differ from the set operation type specified in SQL statements.		
3	Setop_sort	Whether sort processing is performed during the set operation[#1]: • Y: Sort processing is performed. • N: Sort processing is not performed.	CHAR	--
4	Setop_row_cnt	Number of rows processed by the set operation	UINT8	WRAP
5	Setop_start_time	Date and time the set operation processing started The date and time the set operation processing started are output in the following format[#2]: *YYYY-MM-DD*Δ*hh*:*mm*:*ss*.*nnnnnn*[#4]	CHAR	--
6	Setop_end_time	Date and time the set operation processing ended The date and time the set operation processing ended are output in the following format[#3]: *YYYY-MM-DD*Δ*hh*:*mm*:*ss*.*nnnnnn*[#4]	CHAR	--

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

CHAR: The character string indicated in *Information that is output* is output. If the target access path was not executed or was being executed, blanks might be output.

WRAP: When the value exceeds 18,446,744,073,709,551,615, it is reset to the minimum value in wraparound mode.

--: Overflow does not occur. Otherwise, overflow is not applicable.

#1

If the processing is performed by the set operation that references the recursive query names in recursive queries, the total value for all recursive queries is output.

#2

If the processing is performed by the set operation that references the recursive query names in recursive queries, the start time of the first recursive query is output.

#3

If the processing is performed by the set operation that references the recursive query names in recursive queries, the end time of the last recursive query is output.

#4

Four digits are displayed for the year (*YYYY*), and two digits are displayed for the month (*MM*), date (*DD*), hour (*hh*), minute (*mm*), and second (*ss*). Six digits are displayed for the microsecond (*nnnnnn*). A value that consists of fewer digits is padded with leading zeros. Δ indicates a single-byte space.

(d) Information related to hash grouping areas

The table below explains the information related to hash grouping areas that is output as access path statistical information.

Table 10-25: Items that are output as access path statistical information (Information related to hash grouping areas)

No.	Column header	Information that is output	Output format	In the event of overflow
1	Hashgrp_path#	Corresponding access path's tree row number	UINT8	--
2	Hashgrp_area_shortage	Whether a space shortage has occurred in the hash grouping area: • Y: A space shortage has occurred. • N: A space shortage has not occurred.	CHAR	--
3	Hashgrp_area_sufficient_size	Sufficient size (in kilobytes) of hash grouping area that will not result in a shortage of space. If this sufficient size exceeds the maximum value for the adb_sql_exe_hashgrp_area_size operand in the server definition or the client definition, the maximum value for each operand is displayed.	UINT8	--
4	Hashgrp_group_num	Number of rows processed by grouping (before the HAVING clause is evaluated)	UINT8	WRAP
5	Hashgrp_wrktbl_max_row_cnt	Maximum number of rows inserted into a work table (maximum value per real thread)	UINT8	WRAP

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

CHAR: The character string indicated in *Information that is output* is output. If the target access path was not executed or was being executed, blanks might be output.

WRAP: When the value exceeds 18,446,744,073,709,551,615, it is reset to the minimum value in wraparound mode.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

(e) Information related to hash table areas

The table below explains the information related to hash table areas that is output as access path statistical information.

Information about hash filters is also included in the information related to hash table areas.

Table 10-26: Items that are output as access path statistical information (Information related to hash table areas)

No.	Column header	Information that is output	Output format	In the event of overflow
1	Hashtbl_path#	Corresponding access path's tree row number	UINT8	--
2	Hashtbl_area_shortage	Whether a space shortage has occurred in the hash table area[#1, #6]: • Y: A space shortage has occurred. • N: A space shortage has not occurred.	CHAR	--

No.	Column header	Information that is output	Output format	In the event of overflow
3	Hashtbl_wrktbl_bckt_max_row_cnt	Maximum number of rows inserted into a work table (maximum value per real thread) in the event of a space shortage in the hash table area[#2, #7]	UINT8	WRAP
4	Hashtbl_filter_num	Number of hash filters[#3]	UINT8	--
5	Hashtbl_filter_disabled_num	Number of invalidated hash filters[#4, #5]	UINT8	--
6	Hashtbl_sum_filter_check_cnt	Total number of times a hash value was checked by using hash filters[#4, #5] The total value for all hash filters is output.	UINT8	--
7	Hashtbl_max_filter_check_cnt	Largest number of times a hash value was checked by using a specific hash filter[#4, #5] After the number of time a hash value was checked is obtained for each hash filter, the largest one of the obtained numbers is output.	UINT8	--
8	Hashtbl_min_filter_check_cnt	Smallest number of times a hash value was checked by using a specific hash filter[#4, #5] After the number of time a hash value was checked is obtained for each hash filter, the smallest one of the obtained numbers is output.	UINT8	--
9	Hashtbl_sum_filtering_cnt	Total number of times a hash value was excluded by hash filters[#4, #5] The total value for all hash filters is output.	UINT8	--
10	Hashtbl_max_filtering_cnt	Largest number of times a hash value was excluded by a specific hash filter[#4, #5] After the number of time a hash value was excluded is obtained for each hash filter, the largest one of the obtained numbers is output.	UINT8	--
11	Hashtbl_min_filtering_cnt	Smallest number of times a hash value was excluded by a specific hash filter[#4, #5] After the number of time a hash value was excluded is obtained for each hash filter, the smallest one of the obtained numbers is output.	UINT8	--

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

CHAR: The character string indicated in *Information that is output* is output. If the target access path was not executed or was being executed, blanks might be output.

WRAP: When the value exceeds 18,446,744,073,709,551,615, it is reset to the minimum value in wraparound mode.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

#1

If the hash table area is used by the retrieval processing that references the recursive query names in recursive queries, whether a space shortage has occurred is output after all recursive queries are completed.

#2

If the hash table area is used by the retrieval processing that references the recursive query names in recursive queries, of the values for individual recursive queries, the largest value is output.

#3

For an SQL statement to which no hash filter is applied, `0` is output. `0` is also output if the value of the `adb_sql_exe_hashflt_area_size` operand is `0`.

#4

For an SQL statement to which no hash filter is applied, `0` is output. `0` is also output if the value of the `adb_sql_exe_hashflt_area_size` operand is `0` and if the hash filter size is insufficient.

#5

For a hash filter area used for retrieval processing that references the recursive query names in recursive queries, information about the summation results for each hash filter through all recursive queries is output.

#6

If multiple `DISTINCT` set functions are specified with different arguments in the same query specification, grouping is performed multiple times to determine the result of each `DISTINCT` set function. In this case, the information about all grouping processes is output.

#7

If multiple `DISTINCT` set functions are specified with different arguments in the same query specification, grouping is performed multiple times to determine the result of each `DISTINCT` set function. In this case, the largest one of the values resulting from all grouping processes is output.

(f) Information related to subqueries containing external reference columns

The table below explains the information related to subqueries containing external reference columns that is output as access path statistical information.

Table 10-27: Items that are output as access path statistical information (Information related to subqueries containing external reference columns)

No.	Column header	Information that is output	Output format	In the event of overflow
1	`Corsubq_path#`	Corresponding access path's tree row number	`UINT8`	--
2	`Corsubq_request_cnt`	Number of times subqueries containing an external reference column were requested[#]	`UINT8`	`WRAP`

Legend:

`UINT8`: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, `0` or an intermediate value might be output.

`WRAP`: When the value exceeds `18,446,744,073,709,551,615`, it is reset to the minimum value in wraparound mode.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

#

If the subquery contains an external reference column and is under retrieval processing that references the recursive query names in recursive queries, the total value for all recursive queries is output.

(g) Information related to caches for storing subquery requests

The table below explains the information related to caches for storing subquery requests that is output as access path statistical information.

Table 10-28: Items that are output as access path statistical information (Information related to caches for storing subquery requests)

No.	Column header	Information that is output	Output format	In the event of overflow
1	Corsubqc_path#	Corresponding access path's tree row number	UINT8	--
2	Corsubqc_request_cnt	Number of times caches were requested#	UINT8	WRAP
3	Corsubqc_exec_cnt	Number of times subqueries were executed with no cache hits#	UINT8	WRAP

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

WRAP: When the value exceeds 18,446,744,073,709,551,615, it is reset to the minimum value in wraparound mode.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

#

If the subquery is under retrieval processing that references the recursive query names in recursive queries, the total value for all recursive queries is output.

(h) Information related to recursive queries

The following table describes the information related to recursive queries that is output as access path statistical information.

Table 10-29: Items that are output as access path statistical information (Information related to recursive queries)

No.	Column header	Information that is output	Output format	In the event of overflow
1	Recurq_path#	Corresponding access path's tree row number	UINT8	--
2	Recurq_row_cnt	Number of rows processed for recursive queries	UINT8	WRAP
3	Recurq_start_time	Date and time the recursive query processing started The date and time the recursive query processing started are output in the following format: *YYYY-MM-DD*△*hh*:*mm*:*ss.nnnnnn*#	CHAR	--
4	Recurq_end_time	Date and time the recursive query processing ended The date and time the recursive query processing ended are output in the following format: *YYYY-MM-DD*△*hh*:*mm*:*ss.nnnnnn*#	CHAR	--
5	Recurq_max_recursion	Value for the maximum-number-of-recursions specification	UINT8	--

No.	Column header	Information that is output	Output format	In the event of overflow
6	Recurq_recursion_cnt	Number of recursions	UINT8	WRAP

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

CHAR: The character string indicated in *Information that is output* is output. If the target access path was not executed or was being executed, blanks might be output.

WRAP: When the value exceeds 18,446,744,073,709,551,615, it is reset to the minimum value in wraparound mode.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

#

Four digits are displayed for the year (*YYYY*). Two digits are displayed for the month (*MM*), date (*DD*), hour (*hh*), minute (*mm*), and second (*ss*). Six digits are displayed for the microsecond (*nnnnnn*). A value that consists of fewer digits is padded with leading zeros. Δ indicates a single-byte space.

(i) Information related to execution of SQL statements

The table below explains the information related to execution of SQL statements that is output as access path statistical information.

Table 10-30: Items that are output as access path statistical information (Information related to execution of SQL statements)

No.	Column header	Information that is output	Output format	In the event of overflow
1	SQLexec_result_wrk_size	Size of the area for storing SQL statement execution results (bytes)	UINT8	--

Legend:

UINT8: Information is output as an 8-byte unsigned integer. If the target access path was not executed or was being executed, 0 or an intermediate value might be output.

--: Overflow does not occur. Otherwise, overflow is not applicable.

Note

The output items that are not listed in the table are all maintenance information.

(3) Notes about access path statistical information

- Not all access paths are subject to output of access path statistical information. Therefore, the contents of the access path statistical information might not match the SQL statement statistical information. For details about the items that are output as access path statistical information, see (2) Items that are output as access path statistical information.
- For details about the SQL statements subject to output of access path statistical information, see *SQL statements for which access paths are output* under *About access paths* under *How to use access paths (how to use SQL statement execution plans)* in *Tuning Application Programs* in the *HADB Application Development Guide*.

- Normally, access path statistical information is not output when batch transfer of dynamic parameter values is used to execute SQL statements. However, access path statistical information is output when batch transfer of dynamic parameter values is used to execute SQL statements when only one set of dynamic parameters is used. For details about batch transfer of dynamic parameter values, see *Batch transfer of dynamic parameter values* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

10.11.4 Examples of output of SQL trace information and how to interpret the information

When there are multiple connections, SQL trace information for all the connections is output at one time. This subsection explains examples of the SQL trace information that is output and how to interpret the information when there are multiple connections.

Prerequisites

The following SQL statements are executed from multiple connections, and SQL trace information is output for each of the SQL statements.

- `SELECT * FROM T2`

  This SQL statement is executed using the connection whose connection ID is `1`.

  This is referred to in the examples as *SQL statement [1]*.

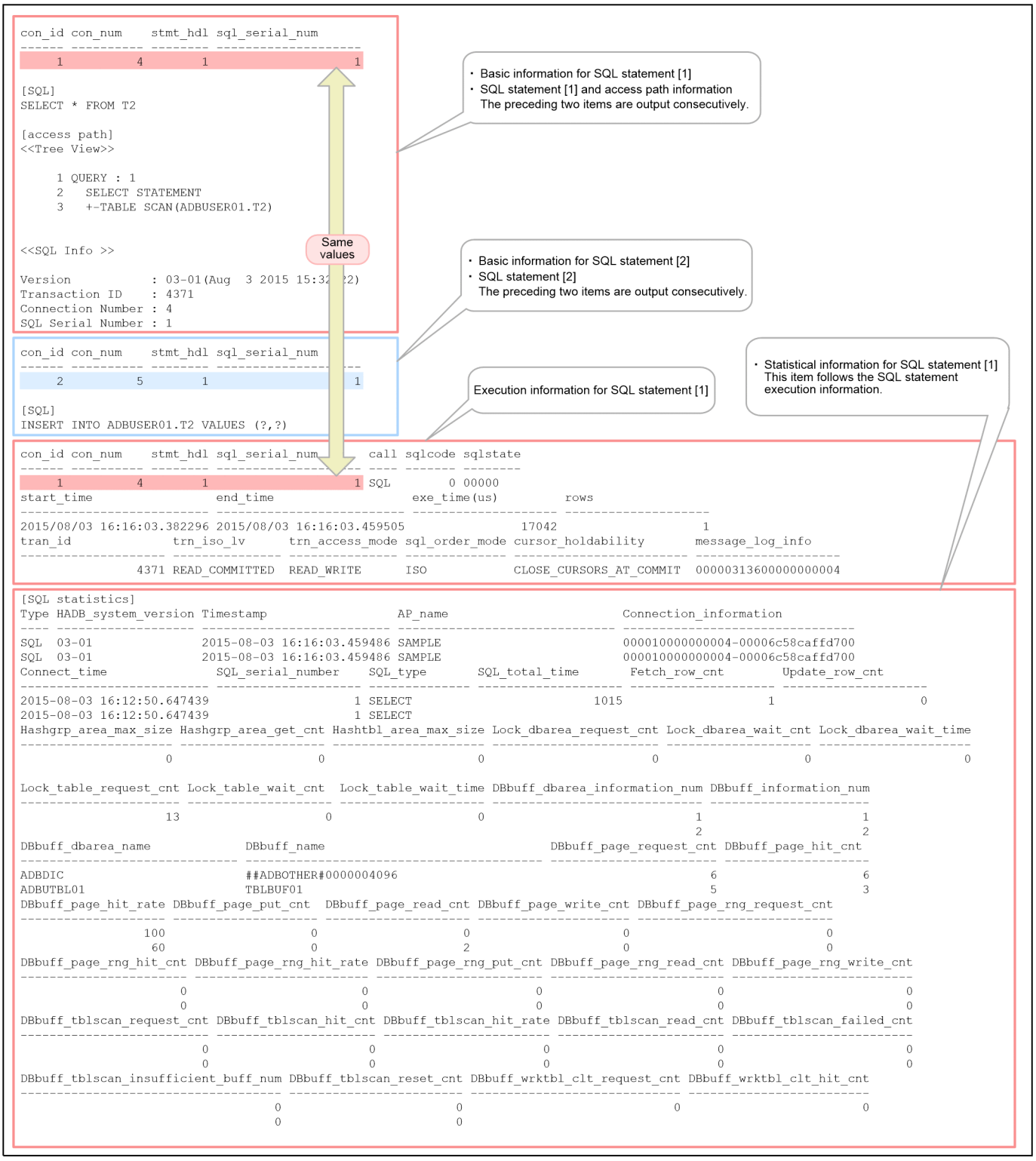
- `INSERT INTO ADBUSER01.T2 VALUES (?,?)`

  This SQL statement is executed using the connection whose connection ID is `2`.

  This is referred to in the examples as *SQL statement [2]*.

> **Note**
>
> The SQL statement basic information and the SQL statement execution information for one connection and the related information# are output consecutively. Information about other connections is never output within this set of information. Therefore, you can search for information related to an executed SQL statement by using the `con_id`, `con_num`, `stmt_hdl`, and `sql_serial_num` values that are output as the SQL statement basic information or the SQL statement execution information.
>
> #
>
> Related information means the information explained in (3) Client-definition information to (10) Access path statistical information in 10.11.2 Information that is output as SQL trace information.

■ Example of output of SQL trace information (part 1)



■ Example of output of SQL trace information (part 2)

```
DBbuff_wrktbl_clt_hit_rate DBbuff_wrktbl_clt_put_cnt DBbuff_wrktbl_clt_read_cnt DBbuff_wrktbl_clt_write_cnt DBbuff_wrktbl_clt_tbl_cnt
-------------------------- ------------------------- -------------------------- --------------------------- -------------------------
                         0                         0                          0                           0                         0

DBbuff_wrktbl_clt_sort_merge_cnt DBbuff_wrktbl_clt_sort_page_max_cnt Log_usrbuf_out_cnt   Log_usrfile_max_size DBarea_extension_cnt
-------------------------------- ---------------------------------- -------------------- -------------------- --------------------
                               0                                  0                    0                    0                    0

Log_usrfile_write_cnt DBbuff_wrktbl_clt_request_fix_cnt DBbuff_wrktbl_clt_pagein_fix_cnt DBbuff_wrktbl_clt_pageout_fix_cnt
--------------------- ---------------------------------- --------------------------------- ----------------------------------
                    0                                  0                                 0                                  0


DBbuff_wrktbl_clt_max_used_blk_num Directory_send_num   Directory_recv_num   Log_send_num         Log_recv_num
---------------------------------- -------------------- -------------------- -------------------- --------------------
                                 0                    0                    0                    0                    0

DBbuff_send_num      DBbuff_recv_num      Node_com_num         Node_com_time        Bidx_page_split_cnt  Tidx_page_split_cnt
-------------------- -------------------- -------------------- -------------------- -------------------- --------------------
                   0                    0                    0                    0                    0                    0

Bidx_validation_check_cnt Ridx_sgmt_skip_cnt   Ridx_chunk_skip_cnt  Ridx_chunk_read_cnt  Ridx_chunk_judge_cnt Ridx_sgmt_read_cnt
------------------------- -------------------- -------------------- -------------------- -------------------- --------------------
                        1                    0                    0                    0                    0                    0

Ridx_sgmt_judge_cnt  SegmentRel_rthd_max_num Csvread_file_cnt     Csvread_file_read_size Hashgrp_area_shortage
-------------------- ------------------------ -------------------- ---------------------- ----------------------
                   0                        0                    0                      0 N

Hashgrp_area_sufficient_size Hashtbl_area_shortage Syndict_file_access_time Auditread_file_cnt   Auditread_file_read_size
---------------------------- --------------------- ------------------------ -------------------- ------------------------
                           0 N                                             0                    0                        0


DBbuff_tblscan_read_size DBbuff_tblscan_rthd_min_size DBbuff_tblscan_rthd_max_size DBbuff_tblscan_use_size
------------------------ ---------------------------- ---------------------------- ------------------------
                       0                            0                            0                        0
                       0                            0                            0                        0

DBbuff_tblscan_insufficient_buff_size Hashflt_disabled Tbldef_req_cnt       Tbldef_access_cnt    Tbldef_cache_access_cnt
------------------------------------- ---------------- -------------------- -------------------- -----------------------
                                    0 N                                   1                    1                       0
                                    0


Tbldef_cache_register_cnt Tbldef_cache_sweep_cnt DBbuff_page_wait_cnt Max_sql_rthd_num     Hashtbl_area_size    Hashflt_area_size
------------------------- ------------------------ ---------------------- -------------------- -------------------- --------------------
                        1                        0                      0                    4                 2000                  200
                                                                        0
```

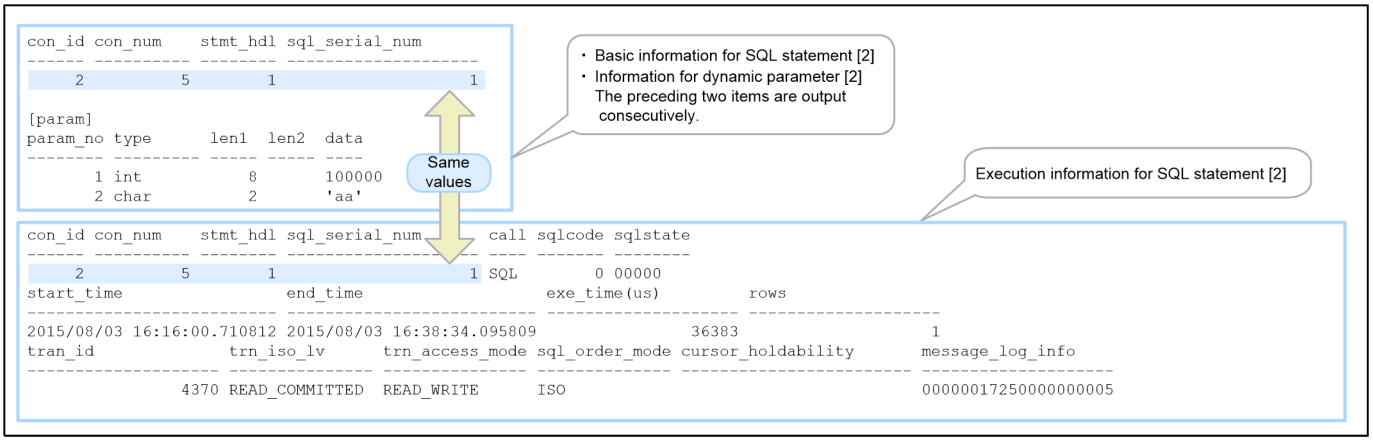
■ Example of output of SQL trace information (part 3)


```
[access path statistics]
<<Data access information>>
Data_path#           Data_dbbuff_page_request_cnt(table) Data_dbbuff_page_hit_cnt(table)
-------------------- ----------------------------------- -------------------------------
                   3                                   5                               3
Data_dbbuff_page_read_cnt(table) Data_dbbuff_page_write_cnt(table) Data_dbbuff_tblscan_request_cnt
-------------------------------- --------------------------------- -----------------------------
                               2                                 0                             0
Data_dbbuff_tblscan_hit_cnt Data_dbbuff_tblscan_read_cnt Data_dbbuff_tblscan_read_size Data_dbbuff_tblscan_failed_cnt
--------------------------- --------------------------- ---------------------------- ----------------------------
                          0                           0                            0                            0
Data_dbbuff_page_request_cnt(btree) Data_dbbuff_page_hit_cnt(btree) Data_dbbuff_page_read_cnt(btree)
----------------------------------- ------------------------------- --------------------------------
                                  0                               0                              0
Data_dbbuff_page_write_cnt(btree) Data_dbbuff_page_request_cnt(text) Data_dbbuff_page_hit_cnt(text)
--------------------------------- ---------------------------------- -----------------------------
                                0                                  0                             0
Data_dbbuff_page_read_cnt(text) Data_dbbuff_page_write_cnt(text) Data_dbbuff_page_request_cnt(range)
------------------------------- -------------------------------- ----------------------------------
                              0                                0                                  0
Data_dbbuff_page_hit_cnt(range) Data_dbbuff_page_read_cnt(range) Data_dbbuff_page_write_cnt(range)
------------------------------- -------------------------------- ----------------------------------
                              0                                0                                  0
Data_dbbuff_wrktbl_clt_tbl_cnt Data_dbbuff_wrktbl_clt_request_cnt Data_dbbuff_wrktbl_clt_page_hit_cnt
------------------------------ ---------------------------------- -----------------------------------
                             0                                  0                                   0
Data_dbbuff_wrktbl_clt_read_cnt Data_dbbuff_wrktbl_clt_write_cnt Data_dbbuff_page_request_cnt(glb_wrk)
------------------------------- -------------------------------- ------------------------------------
                              0                                0                                    0
Data_dbbuff_page_hit_cnt(glb_wrk) Data_dbbuff_page_read_cnt(glb_wrk) Data_dbbuff_page_write_cnt(glb_wrk)
--------------------------------- ---------------------------------- -----------------------------------
                                0                                  0                                   0
Data_bidx_page_split_cnt Data_bidx_validation_check_cnt Data_ridx_chunk_judge_cnt Data_ridx_chunk_skip_cnt
------------------------ ------------------------------ -------------------------- ------------------------
                       0                              0                          0                        0
Data_ridx_sgmt_judge_cnt Data_ridx_sgmt_skip_cnt Data_tidx_page_split_cnt Data_wrktbl_create_cnt
------------------------ ----------------------- ------------------------ ----------------------
                       0                       0                        0                      0
Data_wrktbl_drop_cnt Data_wrktbl_page_assign_num Data_wrktbl_page_free_num Data_wrktbl_page_use_max
-------------------- ---------------------------- --------------------------- -----------------------
                   0                            0                           0                       0
```

■ Example of output of SQL trace information (part 4)

```
Data_wrktbl_sort_merge_cnt Data_wrktbl_sort_page_max_cnt Data_dbarea_extension_cnt Data_segmentrel_rthd_max_num
-------------------------- ------------------------------ -------------------------- ----------------------------
                         0                              0                          0                            0
Data_log_usrfile_write_cnt Data_log_usrbuf_out_cnt Data_log_usrfile_max_size Data_directory_send_num
-------------------------- ----------------------- -------------------------- ----------------------
                         0                       0                          0                      0
Data_access_info_type    Data_csvread_file_cnt Data_csvread_file_read_size Data_deleted_rows_cnt Data_tidx_all_search_cnt
------------------------ --------------------- ---------------------------- --------------------- -------------------------
Access                                       0                            0                     0                         0
Data_auditread_file_cnt Data_auditread_file_read_size Data_grpagg_mem_short_cnt Data_grpagg_mem_exceed_limit_cnt
----------------------- ------------------------------ -------------------------- ---------------------------------
                      0                              0                          0                                 0
Data_grpagg_mem_try_get_cnt Data_grpagg_mem_max_size Data_grpagg_preagg_row_cnt Data_grpagg_rslttbl_reuse_cnt
--------------------------- ------------------------ ---------------------------- -------------------------------
                          0                        0                            0                               0
Data_cs_column_invalid_rows_cnt Data_cs_rowstore_fetch_rows_cnt Data_cs_invalid_info_page_request_cnt Data_cs_invalid_false_positive_cnt
------------------------------- ------------------------------- ------------------------------------- -------------------------------------
                              0                               0                                     0                                     0

<<Scan information>>
Scan_path#           Scan_row_cnt         Scan_start_time            Scan_end_time
-------------------- -------------------- -------------------------- --------------------------
                   3                    1 2015-08-03 16:16:03.458122 2015-08-03 16:16:03.459137
Scan_info_data_max_num Scan_info_data_max_free_num Scan_info_data_size  Scan_info_inpred_max_num
---------------------- -------------------------- --------------------- -----------------------
                     0                          0                     0                       0
Scan_info_inpred_max_free_num Scan_info_inpred_size Scan_info_quantpred_max_num Scan_info_quantpred_max_free_num
----------------------------- --------------------- ------------------------- -----------------------------
                            0                     0                         0                             0
Scan_info_quantpred_size Scan_info_drvtbl_max_num Scan_info_drvtbl_max_free_num Scan_info_drvtbl_size Scan_info_hash_max_num
------------------------ ------------------------ ---------------------------- --------------------- ----------------------
                       0                        0                            0                     0                      0
Scan_info_hash_max_free_num Scan_info_hash_size  Scan_info_hashbckt_max_num Scan_info_hashbckt_size
--------------------------- -------------------- -------------------------- ----------------------
                          0                    0                          0                      0
Scan_outerjoin_info_max_num Scan_outerjoin_info_size Scan_info_csvread_file_max_num Scan_info_csvread_file_max_free_num
--------------------------- ------------------------ ------------------------------ ---------------------------------
                          0                        0                              0                                 0
Scan_info_csvread_file_size Scan_info_csvread_data_max_num Scan_info_csvread_data_max_free_num Scan_info_csvread_data_size
--------------------------- ------------------------------ ----------------------------------- ----------------------------
                          0                              0                                   0                            0

<<Set operation information>>

<<Hash grouping area information>>

<<Hash table area information>>

<<Correlated subqueries information>>

<<Correlated subqueries result cache information>>

<<Recursive queries information>>

<<SQL exec information>>
SQLexec_result_wrk_size SQLexec_result_wrk_max_num SQLexec_result_wrk_init_num SQLexec_mem_size     SQLexec_mem_max_num
----------------------- -------------------------- -------------------------- -------------------- --------------------
                     48                          0                          0                    0                    0
SQLexec_mem_init_num SQLexec_dynamic_mem_alloc_unit SQLexec_delayed_free_mem_max_size SQLexec_stack_size
-------------------- ------------------------------ ------------------------------------ --------------------
                   0                              0                                    0               163840
```

■ Example of output of SQL trace information (part 5)



■ Example of output of SQL trace information (part 6)

・ Statistical information for SQL statement [2]
This item follows the SQL statement execution information.

```
[SQL statistics]
Type HADB_system_version Timestamp                  AP_name                         Connection_information
---- ------------------- ------------------------- ------------------------------- ------------------------------
SQL  03-01               2015-08-03 16:38:34.095803 PERFORMACEMEASURE               000020000000005-00006c58c95fa700
SQL  03-01               2015-08-03 16:38:34.095803 PERFORMACEMEASURE               000020000000005-00006c58c95fa700
SQL  03-01               2015-08-03 16:38:34.095803 PERFORMACEMEASURE               000020000000005-00006c58c95fa700
Connect_time              SQL_serial_number    SQL_type       SQL_total_time       Fetch_row_cnt        Update_row_cnt
------------------------- -------------------- -------------- -------------------- -------------------- --------------------
2015-08-03 16:13:16.011371                   1 INSERT                         18982                    0                    1
2015-08-03 16:13:16.011371                   1 INSERT
2015-08-03 16:13:16.011371                   1 INSERT
Hashgrp_area_max_size Hashgrp_area_get_cnt Hashtbl_area_max_size Lock_dbarea_request_cnt Lock_dbarea_wait_cnt Lock_dbarea_wait_time
--------------------- -------------------- --------------------- ----------------------- -------------------- ---------------------
                    0                    0                     0                       1                    0                     0


Lock_table_request_cnt Lock_table_wait_cnt  Lock_table_wait_time DBbuff_dbarea_information_num DBbuff_information_num
---------------------- -------------------- -------------------- ---------------------------- ----------------------
                    13                    0                    0                            1                      1
                                                                                            2                      1
                                                                                            3                      2
DBbuff_dbarea_name             DBbuff_name                                DBbuff_page_request_cnt DBbuff_page_hit_cnt
------------------------------ ------------------------------------------ ----------------------- --------------------
ADBDIC                         ##ADBOTHER#0000004096                                           31                   22
ADBSTBL                        ##ADBOTHER#0000004096                                            6                    2
ADBUTBL01                      TBLBUF01                                                         3                    3
DBbuff_page_hit_rate DBbuff_page_put_cnt  DBbuff_page_read_cnt DBbuff_page_write_cnt DBbuff_page_rng_request_cnt
-------------------- -------------------- -------------------- --------------------- ----------------------------
                  71                    0                    9                     0                            0
                  34                    0                    4                     0                            0
                 100                    1                    0                     1                            0
DBbuff_page_rng_hit_cnt DBbuff_page_rng_hit_rate DBbuff_page_rng_put_cnt DBbuff_page_rng_read_cnt DBbuff_page_rng_write_cnt
----------------------- ------------------------ ------------------------ ------------------------ -------------------------
                      0                        0                        0                        0                         0
                      0                        0                        0                        0                         0
                      0                        0                        0                        0                         0
DBbuff_tblscan_request_cnt DBbuff_tblscan_hit_cnt DBbuff_tblscan_hit_rate DBbuff_tblscan_read_cnt DBbuff_tblscan_failed_cnt
-------------------------- ---------------------- ------------------------ ------------------------ -------------------------
                         0                      0                        0                        0                         0
                         0                      0                        0                        0                         0
                         0                      0                        0                        0                         0
DBbuff_tblscan_insufficient_buff_num DBbuff_tblscan_reset_cnt DBbuff_wrktbl_clt_request_cnt DBbuff_wrktbl_clt_hit_cnt
------------------------------------ ------------------------ ------------------------------ -------------------------
                                   0                        0                              0                         0
                                   0                        0
                                   0                        0
DBbuff_wrktbl_clt_hit_rate DBbuff_wrktbl_clt_put_cnt DBbuff_wrktbl_clt_read_cnt DBbuff_wrktbl_clt_write_cnt DBbuff_wrktbl_clt_tbl_cnt
-------------------------- ------------------------- ------------------------- -------------------------- -------------------------
                         0                         0                         0                          0                         0


DBbuff_wrktbl_clt_sort_merge_cnt DBbuff_wrktbl_clt_sort_page_max_cnt Log_usrbuf_out_cnt   Log_usrfile_max_size DBarea_extension_cnt
-------------------------------- ---------------------------------- -------------------- -------------------- --------------------
                               0                                  0                    0                    0                    0
```

■ Example of output of SQL trace information (part 7)

```
Log_usrfile_write_cnt DBbuff_wrktbl_clt_request_fix_cnt DBbuff_wrktbl_clt_pagein_fix_cnt DBbuff_wrktbl_clt_pageout_fix_cnt
--------------------- --------------------------------- --------------------------------- ---------------------------------
                    1                                 0                                 0                                 0

DBbuff_wrktbl_clt_max_used_blk_num Directory_send_num   Directory_recv_num   Log_send_num         Log_recv_num
---------------------------------- -------------------- -------------------- -------------------- --------------------
                                 0                    0                    0                    0                    0

DBbuff_send_num      DBbuff_recv_num      Node_com_num         Node_com_time        Bidx_page_split_cnt  Tidx_page_split_cnt
-------------------- -------------------- -------------------- -------------------- -------------------- --------------------
                   0                    0                    0                    0                    0                    0

Bidx_validation_check_cnt Ridx_sgmt_skip_cnt   Ridx_chunk_skip_cnt  Ridx_chunk_read_cnt  Ridx_chunk_judge_cnt Ridx_sgmt_read_cnt
------------------------- -------------------- -------------------- -------------------- -------------------- --------------------
                       15                    0                    0                    0                    0                    0

Ridx_sgmt_judge_cnt  SegmentRel_rthd_max_num Csvread_file_cnt     Csvread_file_read_size Hashgrp_area_shortage
-------------------- ------------------------ -------------------- ---------------------- ---------------------
                   0                        0                    0                      0 N

Hashgrp_area_sufficient_size Hashtbl_area_shortage Syndict_file_access_time Auditread_file_cnt   Auditread_file_read_size
---------------------------- -------------------- ------------------------- -------------------- ------------------------
                           0 N                                             0                    0                        0

DBbuff_tblscan_read_size DBbuff_tblscan_rthd_min_size DBbuff_tblscan_rthd_max_size DBbuff_tblscan_use_size
------------------------ ---------------------------- ---------------------------- ------------------------
                       0                            0                            0                        0
                       0                            0                            0                        0
                       0                            0                            0                        0

DBbuff_tblscan_insufficient_buff_size Hashflt_disabled Tbldef_req_cnt       Tbldef_access_cnt    Tbldef_cache_access_cnt
------------------------------------- ---------------- -------------------- -------------------- ----------------------
                                    0 N                                   1                    0                      1
                                    0
                                    0

Tbldef_cache_register_cnt Tbldef_cache_sweep_cnt DBbuff_page_wait_cnt Max_sql_rthd_num     Hashtbl_area_size    Hashflt_area_size
------------------------- ------------------------ ---------------------- -------------------- -------------------- --------------------
                        0                        0                      0                    4                 2000                  200
                                                                        0
                                                                        0
```

Note

- The SQL statement execution information might not be output in chronological order of the end times of the SQL statements or calls. However, this information is sorted in chronological order within each connection.
- If the output destination files for SQL trace information are swapped, the related information might be output to a different file from the file for the SQL statement basic information and SQL statement execution information. In such a case, you can locate the corresponding SQL statement basic information and SQL statement execution information in the SQL trace file whose number falls immediately before that of the SQL trace file to which the related information has been output (if the trace file number is the smallest value, the information is the last file number).

10.11.5 Preparations for outputting SQL trace information

Before you output SQL trace information, complete the following tasks.

(1) Checking the disk capacity

The SQL trace information is output to SQL trace files. The maximum size of one SQL trace file is specified in the `adb_sql_trc_txtfile_size` operand in the server definition (the default is 256 megabytes). A maximum of eight SQL trace files are created. Make sure that you have sufficient disk capacity for the trace files that are created.

(2) Specifying server definitions

You use the `adb_sql_trc_out` operand in the server definition to specify whether the SQL trace information is to be output. The default is `Y` (SQL trace information is output).

In general, we recommend that you collect all SQL trace information because the information is useful for troubleshooting and tuning purposes. However, because the SQL trace information is output to files, the performance of SQL statement processing performance might be adversely affected. As the amount of information to be output increases, the effects on processing performance might also increase. If the effects on SQL statement processing performance are too significant, you can use the following server definition operands to reduce the amount of SQL trace information that is output:

- `adb_sql_trc_param`
  This operand specifies whether dynamic parameter information is to be output. The default is `Y` (dynamic parameter information is output).
- `adb_sql_trc_accesspath`
  This operand specifies whether access path information and access path statistical information are to be output. The default is `Y` (access path information and access path statistical information are output).
- `adb_sql_trc_level`
  This operand specifies the unit for outputting SQL trace information. The default is `SQL` (SQL trace information is output for each SQL statement).

> **Note**
>
> After the HADB server has started, you can use the `adbchgsqltrc` command to stop output of SQL trace information or to change the SQL trace information that is to be output.

10.11.6 Operation when SQL trace information is set to be output

This subsection explains the operations when SQL trace information is set to be output.

(1) Making a backup of SQL trace files explains the operations that need to be performed regularly when SQL trace information is set to be output. The subsections starting with (2) Checking the SQL trace information that has been output explain the operations that are to be performed only as needed.

(1) Making a backup of SQL trace files

When one SQL trace file becomes full as a result of output of SQL trace information, another SQL trace file is used. There are a total of eight SQL trace files. Once all SQL trace files become full, output of SQL trace information continues by overwriting the oldest file. Therefore, if you want to save the collected SQL trace information, back up the SQL trace files.

When output to one SQL trace file is switched to another SQL trace file, the KFAA81004-I message is output to the message log file. When this message is output, back up the SQL trace file that is being swapped out.

(2) Checking the SQL trace information that has been output

If you want to check what kind of information has been output as the SQL trace information, execute the adbchgsqltrc command.

■ Command execution example

```
adbchgsqltrc -d

sqltrace param    accesspath level
ACTIVE   ACTIVE   INACTIVE   SQL
```

Explanation

- sqltrace

  Displays whether SQL trace information has been output. In this example, ACTIVE is displayed (SQL trace information has been output).

- param

  Displays whether dynamic parameter information has been output. In this example, ACTIVE is displayed (dynamic parameter information has been output).

- accesspath

  Displays whether access path information and access path statistical information have been output. In this example, INACTIVE is displayed (access path information and access path statistical information have not been output).

- level

  Displays the unit of SQL trace information output. In this example, SQL is displayed (SQL trace information was output by SQL statement).

For details about the execution results of the adbchgsqltrc command, see *adbchgsqltrc (Start or Stop Output of SQL Trace Information)* in the manual *HADB Command Reference*.

(3) Changing the SQL trace information to be output

If you want to change the SQL trace information to be output, execute the adbchgsqltrc command.

■ Command execution example

Example 1: Stopping output of dynamic parameter information

```
adbchgsqltrc -n param
```

Explanation

- -n param

  Specifies that output of dynamic parameter information is to be stopped.

Example 2: Starting output of access path information and access path statistical information by call

```
adbchgsqltrc -y accesspath -l call
```

Explanation

- `-y accesspath`
  Specifies that output of access path information and access path statistical information is to be started.
- `-l call`
  Specifies that the output is to be by call.

For details about the options that can be specified when the `adbchgsqltrc` command is executed, see *adbchgsqltrc (Start or Stop Output of SQL Trace Information)* in the manual *HADB Command Reference*.

Note that changes made with the `adbchgsqltrc` command to the SQL trace information to be output do not take effect the next time the HADB server starts. The next time the HADB server starts, the information specified in the server definition is applied.

(4) Stopping output of SQL trace information

If you want to stop output of SQL trace information, execute the `adbchgsqltrc` command.

■ Command execution example

```
adbchgsqltrc -e
```

For details about the options that can be specified when the `adbchgsqltrc` command is executed, see *adbchgsqltrc (Start or Stop Output of SQL Trace Information)* in the manual *HADB Command Reference*.

Note that the setting to stop output of SQL trace information that is specified with the `adbchgsqltrc` command does not take effect the next time the HADB server starts. The next time the HADB server starts, the information specified in the server definition is applied.

(5) Starting output of SQL trace information

If you want to start output of SQL trace information, execute the `adbchgsqltrc` command.

■ Command execution example

```
adbchgsqltrc -s
```

For details about the options that can be specified when the `adbchgsqltrc` command is executed, see *adbchgsqltrc (Start or Stop Output of SQL Trace Information)* in the manual *HADB Command Reference*.

Note that the setting to start output of SQL trace information that is specified with the `adbchgsqltrc` command does not take effect the next time the HADB server starts. The next time the HADB server starts, the information specified in the server definition is applied.

(6) Checking the output destination files for the SQL trace information

You use the update dates and times of the SQL trace files to determine the SQL trace file to which SQL trace information has been output. The current SQL trace information is being output to the SQL trace file with the most recent update date and time. You can use the OS's `ls` command, for example, to check the update dates and times of the SQL trace files.

If the update date and time of an SQL trace file has been changed because you updated the SQL trace file by, for example, using a text editor, the current output destination cannot be determined from the update dates and times of the SQL trace files. Check the end date and time of the last SQL statement output to each SQL trace file. The SQL trace file containing the SQL statement with the most recent date and time is the current output destination.

10.11.7 Using SQL trace information to determine the cause of errors in SQL statements

(1) How to determine the cause of errors in SQL statements

The following procedure explains how to use SQL trace information to determine the cause of errors in SQL statements.

Procedure

1. Check SQLCODE that has been output to the SQL statement execution information.

   SQLCODE is displayed in the sqlcode column. Check for an SQLCODE that is a negative value.

   Example:

```
con_id con_num    stmt_hdl sql_serial_num       call sqlcode sqlstate
------ ---------- -------- -------------------- ---- ------- --------
     1          1        1                    1 SQL     -559 42K07
start_time                end_time                  exe_time(us)         rows
------------------------- ------------------------- -------------------- --------------------
2015/08/03 19:02:01.185562 2015/08/03 19:02:01.246807                61245                    0
tran_id             trn_iso_lv      trn_access_mode sql_order_mode cursor_holdability       message_log_info
------------------- --------------- --------------- -------------- ------------------------ --------------------
               4375 READ_COMMITTED  READ_WRITE      ISO                                     00000692530000000001
```

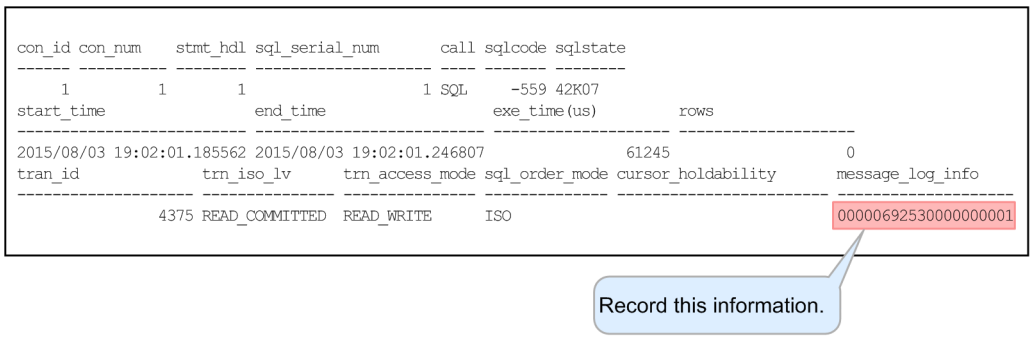
2. Check the message corresponding to the output SQLCODE.

   For details about how SQLCODEs correspond to messages, see *Interpreting SQLCODEs* in the manual *HADB Messages*.

   In this example, SQLCODE is -559, which means that the corresponding message is KFAA30559-E.

3. Make a note of the message log information that is displayed under message_log_info.

   Example:

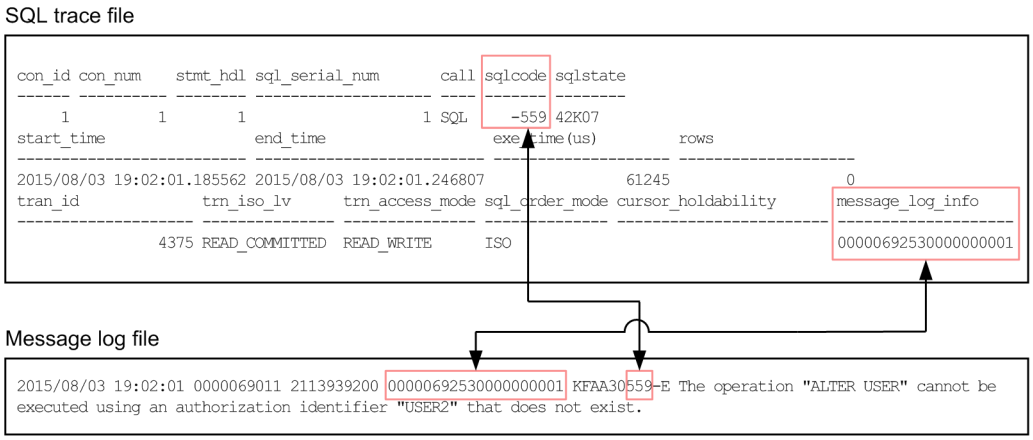


4. Retrieve the message that has been output to the message log file.

   Retrieve from the message log file the message that was checked in step 2. In this example, retrieve the KFAA30559-E message from the message log file.

   Then check if the message log information obtained in step 3 matches the message log information contained in the message. If they match, the KFAA30559-E message is the target message.

   Example:



5. Check the corrective action given in the message.
   See the corrective action given in the message and identify the cause of the error in the SQL statement.
   In this example, take the corrective action given in the Action column for the `KFAA30559-E` message.

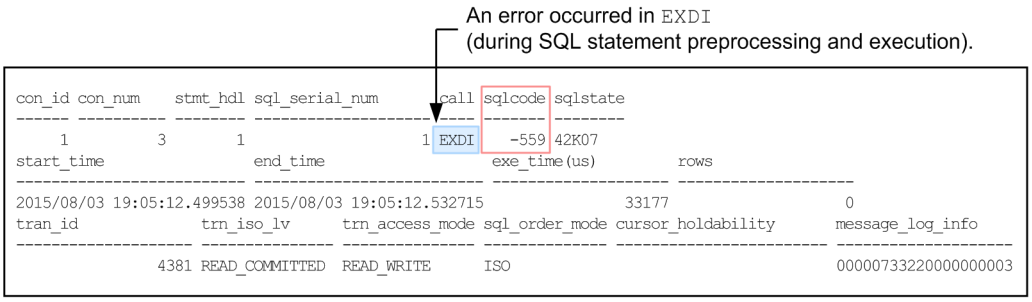
(2) How to identify how far the SQL statement was executed

The method depends on the unit of SQL trace information output.

(a) If SQL trace information was set to be output for each call

An error occurred in the call whose `SQLCODE` in the `sqlcode` column is a negative value.

Example:



(b) If SQL trace information was set to be output for each SQL statement

Identify the items that have been output and the items that have not been output to determine the call resulting in the error. The following shows an example.

- Example 1: The executed SQL statement has been output, but access path information has not been output.
  An error occurred during processing of `PREP` (preprocessing of SQL statement) or `EXDI` (preprocessing and execution of SQL statement).

```
con_id con_num    stmt_hdl sql_serial_num
------ ---------- -------- --------------------
     1          1        1                    9

[SQL]
SELECT * FROM T1

con_id con_num                                call sqlcode sqlstate
------ ---------- --------                    ---- ------- --------
     1          1        1                    SQL     -204 42741
start_time                                         exe_time(us)         rows
--------------------------                    ---- -------------------- --------------------
2015/08/21 14:40:56.931828 2015/08/21 14:40:56.957078                25250                    0
tran_id              trn_iso_lv      trn_access_mode sql_order_mode cursor_holdability       message_log_info
-------------------- --------------- --------------- -------------- ------------------------ --------------------
               77132 READ_COMMITTED  READ_WRITE      BYTE                                    00000187840000000001
```

Access path information has not been output.

- Example 2: The executed SQL statement, access path information, and dynamic parameter information have been output, but the SQL statement statistical information and access path statistical information have not been output.
  An error occurred during processing of `OPEN` (cursor open processing).

```
con_id con_num    stmt_hdl sql_serial_num
------ ---------- -------- --------------------
     1          2        1                    1

[SQL]
SELECT * FROM T1 LIMIT ?

[access path]
<<Tree View>>

     1 QUERY : 1
     2   SELECT STATEMENT
     3   |-TABLE SCAN(ADBUSER01.T1)
     4   +-LIMIT ? PARAMETER


<<SQL Info >>

Version           : 03-01(Aug  3 2015 15:32:22)
Transaction ID    : 4378
Connection Number : 2
SQL Serial Number : 1

con_id con_num    stmt_hdl sql_serial_num
------ ---------- -------- --------------------
     1          2        1                    1

[param]
param_no type      len1  len2  data
-------- --------- ----- ----- ----
       1 int           8    -1

con_id con_num    stmt_hdl sql_serial_num       call sqlcode sqlstate
------ ---------- -------- -------------------- ---- ------- --------
     1          2        1                    1 SQL    -1301 22510
start_time                 end_time                   exe_time(us)         rows
-------------------------- -------------------------- -------------------- --------------------
2015/08/03 19:02:29.905449 2015/08/03 19:02:32.148798               155743                    0
tran_id              trn_iso_lv      trn_access_mode sql_order_mode cursor_holdability       message_log_info
-------------------- --------------- --------------- -------------- ------------------------ --------------------
                4378 READ_COMMITTED  READ_WRITE      ISO            CLOSE_CURSORS_AT_COMMIT  00000701170000000002

tran_id              start_time                 end_time                   exe_time(us)
-------------------- -------------------------- -------------------------- --------------------
                4378 2015/08/03 19:02:29.904799 2015/08/03 19:02:34.837262              4932463
trn_iso_lv      trn_access_mode sql_order_mode
--------------- --------------- --------------
READ_COMMITTED  READ_WRITE      ISO
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~
```

SQL statement statistical information and access path statistical information have not been output.

10.11.8 Using SQL trace information to tune SQL statements

This subsection explains how to use SQL trace information to tune SQL statements.

(1) When the SQL trace information is output by SQL statement

This subsection explains how to tune SQL statements when the SQL trace information is output by SQL statement.

Procedure

1. Prepare the information that will be used for tuning.
   Prepare the SQL trace information so that you can view the information.
2. Identify an SQL statement that took too long to execute.
   Check the execution durations of the SQL statements that are displayed under `exe_time` in the SQL statement execution information that is output as SQL trace information. Identify an SQL statement that took too long to execute.
3. Make a note of the values displayed in the SQL statement execution information.
   Make a note of the following values in the SQL statement execution information that was identified in step 2:
   - `con_id` (connection ID)
   - `con_num` (connection sequence number)
   - `stmt_hdl` (statement handle allocated to the SQL statement)
   - `sql_serial_num` (SQL statement sequence number)
4. Check the basic information for the SQL statements.
   Locate the set of SQL statement basic information that satisfies the following condition:
   - The values of `con_id`, `con_num`, `stmt_hdl`, and `sql_serial_num` are the same as those obtained in step 3.

   For details about how to locate the target SQL statement basic information from multiple sets of SQL statement basic information, see 10.11.4 Examples of output of SQL trace information and how to interpret the information.
5. Check the executed SQL statement and access path information and the dynamic parameter information.
   Check the *executed SQL statement and access path information* and the *dynamic parameter information* that were output immediately after the SQL statement basic information identified in step 4.
6. Check the statistical information.
   See the following statistical information that was output immediately after the SQL statement execution information, identified in step 2, and use it to tune the SQL statement:
   - SQL statement statistical information
   - Access path statistical information

(2) When the SQL trace information is output by call

This subsection explains how to tune SQL statements when the SQL trace information is output by call.

Procedure

1. Prepare the information that will be used for tuning.
   Prepare the SQL trace information so that you can view the information.
2. Identify an SQL statement that took too long to execute.
   Check the execution durations of the SQL statements that are displayed under `exe_time` in the SQL statement execution information that is output as SQL trace information. Identify an SQL statement that took too long to execute.

3. Make a note of the values displayed in the SQL statement execution information.
   Make a note of the following values in the SQL statement execution information that was identified in step 2:
   - `con_id` (connection ID)
   - `con_num` (connection sequence number)
   - `stmt_hdl` (statement handle allocated to the SQL statement)
   - `sql_serial_num` (SQL statement sequence number)
4. Check the SQL statement execution information.
   Locate the SQL statement execution information that satisfies the following conditions:
   - The values of `con_id`, `con_num`, `stmt_hdl`, and `sql_serial_num` are the same as those obtained in step 3.
   - If a retrieval SQL statement was executed, the call type is `PREP` or `OPEN`.
   - If an update SQL statement was executed, the call type is `PREP`, `EXEC`, or `EXDI`.
5. Check the executed SQL statement and access path information and the dynamic parameter information.
   Check the following information that is output immediately after the SQL statement execution information identified in step 4:
   - If the call type is `PREP` or `EXDI`, check the *executed SQL statement and access path information*.
   - If the call type is `OPEN` or `EXEC`, check the *dynamic parameter information*.
6. Check the SQL statement execution information.
   Locate the SQL statement execution information that satisfies both of the following conditions:
   - The values of `con_id`, `con_num`, `stmt_hdl`, and `sql_serial_num` are the same as those obtained in step 3.
   - The call type is `CLOS`, `EXEC`, or `EXDI`.
7. Check the statistical information.
   See the following statistical information that has been output immediately after the SQL statement execution information, identified in step 6, and use it to tune the SQL statement:
   - SQL statement statistical information
   - Access path statistical information

10.11.9 Corrective action to take if SQL trace information was not output due to an error

The SQL trace information output processing stops if any of the following errors occurs:

- Shortage of disk space (`KFAA51010-W` message is output)
- Lack of privilege to access SQL trace files (`KFAA51010-W` message is output)
- Insufficient file descriptors (`KFAA51010-W` message is output)
- Memory shortage (`KFAA51011-W` message is output)

The following procedure explains how to restart output of SQL trace information.

Procedure

1. Check the SQL trace information output status.
   Execute the `adbchgsqltrc` command:

```
adbchgsqltrc -d

sqltrace param    accesspath level
INACTIVE ACTIVE   ACTIVE     CALL
```

   If the SQL trace information output processing has stopped, `INACTIVE` is displayed in the `sqltrace` column.
2. Eliminate the cause of the error.
   Check the error message that has been output to the message log file and eliminate the cause of the error.
3. Restart the SQL trace information output processing.
   Execute the `adbchgsqltrc` command:

```
adbchgsqltrc -s
```

Note

If the disk storing the SQL trace files has become full, a warning message is output to the message log file and SQL trace information output processing stops. Even when SQL trace information output processing has stopped, SQL statement processing continues.

10.11.10 Stopping using SQL tracing

If you want to stop using SQL tracing, specify `N` in the `adb_sql_trc_out` operand in the server definition. After the HADB server has stopped, manually delete unneeded SQL trace files.

Note

If you want to stop using SQL tracing while the HADB server is running, execute the `adbchgsqltrc` command:

```
adbchgsqltrc -e
```

10.11.11 Notes about using the multi-node function

When the multi-node function is running, SQL trace files are created for each node. The SQL trace information is output to the SQL trace files on the node where the corresponding transactions have executed. Therefore, if you want to output SQL trace information for all transactions, you will have to output the SQL trace information on all HADB servers in the multi-node configuration.

Note

The operand values output to the client-definition information might differ from one node to another. Therefore, the client-definition information and the information related to the connection status of the application programs that use that client-definition information are output on all nodes.

10.12 Working with the system log files

This section explains how to work with the system log files.

10.12.1 System log file configuration and server definition specification

System log files consist of two types of files: a master log file and a set of user log files. In the master log file, history information (event logs) regarding system control (such as start and end of transactions) and management information of user log files are output. When the HADB server starts, only one master log file is automatically created.

In user log files, operation history information (user logs) of a database is output. One user log file is created for each transaction that updates the database or each real thread. When the HADB server starts, user log files are automatically created. The number and initial size of user log files to be created are determined by the definitions in the server definition.

Server definition operands related to user log files

- `adb_log_usrfile_num`

  This operand specifies the number of user log files that are created when the HADB server starts.

- `adb_log_usrfile_size`

  This operand specifies the initial size of user log files and the trigger for reducing the size of user log files.

  If the size of a user log file exceeds the initial size specified here for user log files, the user log file is expanded automatically. Note that, however, expansion of user log files will have a significant impact on update processing performance. For details, see 13.1.8 Expanding the initial size of user log files.

- `adb_log_usrbuf_num`

  This operand specifies the number of user log buffer files.

For details about each operand, see 7.2.3 Operands related to system logs (set format).

Note

- For details about system log files, see (2) System log in 2.11.1 Recovery flow based on a restart.
- For details about how to determine the number of user log files, see 6.12.15 Determining the number of user log files.

10.12.2 Trigger for reducing the size of the system log files

As a database is updated, system logs are output, which increases the size of system log files. Especially, when a database is massively updated (for example, by an unscheduled operation), the size of some user log files might increase considerably. A problem, such as DB directory suppression, might occur as a result. To prevent the size of the DB directory from being suppressed, the HADB server automatically reduces the size of system log files triggered by the following conditions.

Trigger for reducing the size of the master log file

If the size of the master log file is larger than 2 megabytes when the transaction that updated the database was settled, the size of the master log file is reduced. However, if either of the following conditions is satisfied, the processing of reducing the size of the master log file is skipped because the system logs being used exist in the master log file:

- Another transaction that updates the database exists.

- The transaction that caused processing to reduce master log file size is running processing to delete data in the archive directory.

> **Important**
>
> If multiple transactions that update the database continue running, processing to reduce master log file size is not executed. Therefore, we recommend that you regularly (for example, once a day) and intentionally set the time period during which update transactions are not concurrently executed.

> **Note**
>
> If you execute the following SQL statements or command for the archivable multi-chunk table, the processing for deleting data in the archive directory is executed:
>
> - `DROP SCHEMA` statement
> - `REVOKE` statement with `SCHEMA` specified
> - `DROP TABLE` statement
> - `TRUNCATE TABLE` statement
> - `ALTER TABLE` statement (when executed to change an archivable multi-chunk table to a regular multi-chunk table)
> - `PURGE CHUNK` statement
> - `adbunarchivechunk` command

Trigger for reducing the size of user log files

When the update transaction was settled, the size of a user log file used by that update transaction might be larger than the size specified (as the trigger value for reducing the size of user log files) in the `adb_log_usrfile_size` operand in the server definition. In this case, the size of the user log file is reduced to the initial size#.

#

This is the size specified as the initial size of user log files in the `adb_log_usrfile_size` operand.

Consider the following related to reduction of user log file size.

- To prevent the impact (on update performance) of processing to reduce user log file size:
  If there is enough free space on the disk, in the `adb_log_usrfile_size` operand, specify `0` as the trigger value for reducing user log file size. If `0` is specified, processing to reduce user log file size is not performed.
- To prevent a shortage of disk space due to increased user log file size:
  Omit specifying the `adb_log_usrfile_size` operand. Alternatively, in the `adb_log_usrfile_size` operand, specify a value other than `0` for the trigger value for reducing user log file size. In this case, processing to reduce user log file size is performed. However, during normal operation, an increased number of processes for reducing user log file size might impact update performance. Therefore, regularly check the number of processes for reducing user log file size. For details about how to check the number of processes for reducing user log file size, see 13.1.9 Re-evaluating the trigger size for reducing user log files.

11 Unscheduled Operations

This chapter explains operational items that are performed on an unscheduled basis to maintain the HADB server.

11.1 Base table operations

This section explains base table operations.

For details about operations on multi-chunk tables of base tables, see 11.4 Performing operations on multi-chunk tables.

11.1.1 Defining a base table

To define a base table, execute the CREATE TABLE definition SQL statement.

For details about the CREATE TABLE statement, see *CREATE TABLE (define a table)* in the manual *HADB SQL Reference*.

Note that the specification of STORAGE FORMAT in the CREATE TABLE statement differs depending on whether the base table is to be defined as a row store table or column store table.

- **To define a base table as a row store table**
  Specify ROW for STORAGE FORMAT in the CREATE TABLE statement. Alternatively, omit the STORAGE FORMAT specification.
- **To define a base table as a column store table**
  Specify COLUMN for STORAGE FORMAT in the CREATE TABLE statement.

If you define a base table as a multi-chunk table, see also 11.4.1 Defining a multi-chunk table.

Important

- In HADB, a user (the HADB user with the authorization identifier that was used for the current connection to the HADB server) can define base tables only in a schema which that user owns.
- Before defining a base table, make sure that you have designed the base table. For details about designing a base table, see 5.2 Designing a table.

11.1.2 Adding a column to a base table

To add a column to a base table, execute the ALTER TABLE definition SQL statement.

The HADB user who defined the base table can use the ALTER TABLE statement to add a column.

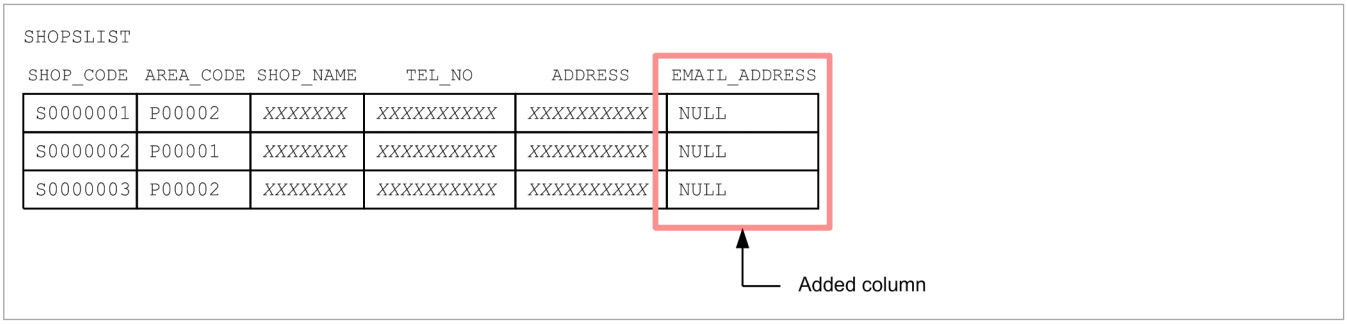
Only a single column can be added, and it is added after the last column in the base table. Null values are stored in the added column.

The following shows a specification example of adding a column to a base table.

Specification example

The column name EMAIL_ADDRESS is added to the shop table (SHOPSLIST). The data type of the column name EMAIL_ADDRESS is set to VARCHAR(100).

```
ALTER TABLE "SHOPSLIST"
    ADD COLUMN "EMAIL_ADDRESS" VARCHAR(100)
```



When you execute the ALTER TABLE statement, you can also specify NOT NULL or BRANCH for the column definition. For details about the specification format and rules for the ALTER TABLE statement, see *ALTER TABLE (alter table definition)* in *Definition SQL* in the manual *HADB SQL Reference*.

You cannot use the ALTER TABLE statement to add a column in the following situations:

- The target base table becomes non-updatable.
- The target base table is a FIX table, and the segments for storing rows have been assigned to the table.
- For the target base table, for which segments for storing rows have been assigned, BRANCH ALL was specified when the CREATE TABLE statement was executed.
- For the target base table, for which segments for storing rows have been assigned, NOT NULL is specified in the column definition of the ALTER TABLE statement.

If a column cannot be added because one of the preceding cases applies, see 15.8.2 When a column cannot be added to a base table.

For details about the status in which segments for storing rows are assigned, see 5.3.1 Notes on defining B-tree indexes (unfinished status of B-tree indexes).

11.1.3 Changing the column name of a base table

To change the column name of a base table, execute the ALTER TABLE definition SQL statement.

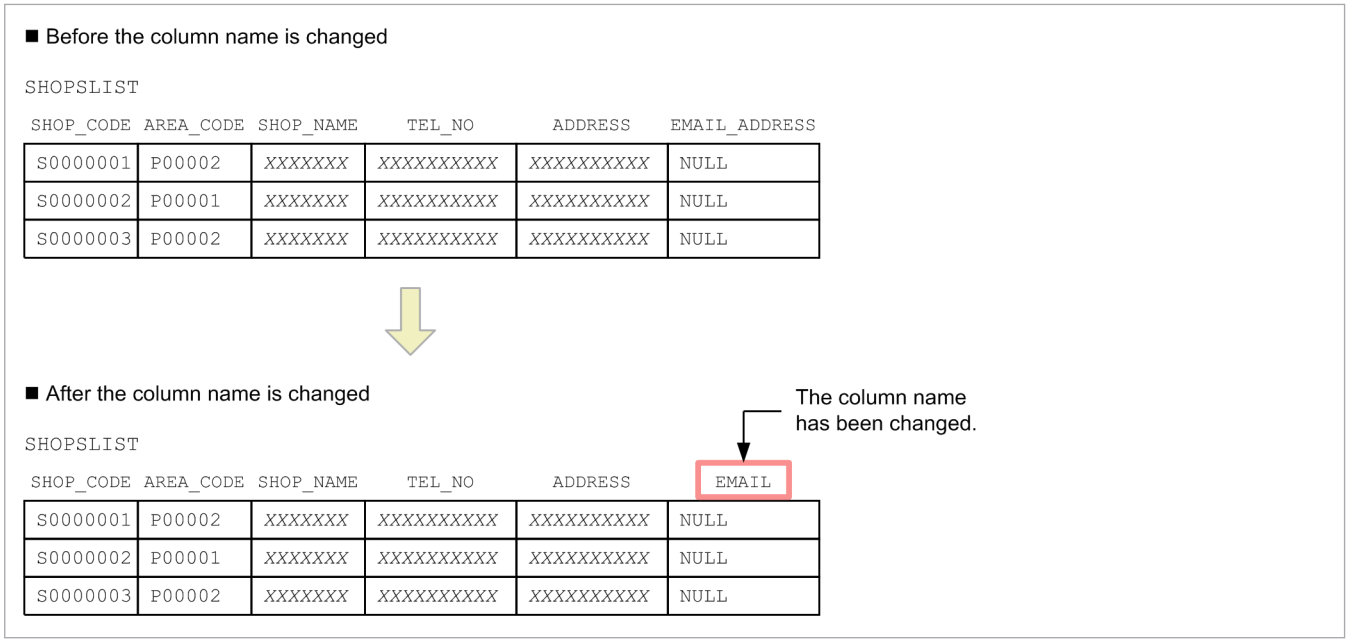
The HADB user who defined the base table can use the ALTER TABLE statement to change the column name.

The following shows a specification example of changing the column name of a base table.

Specification example

The name of the EMAIL_ADDRESS column of the shop table (SHOPSLIST) is changed to EMAIL.

```
ALTER TABLE "SHOPSLIST"
    RENAME COLUMN FROM "EMAIL_ADDRESS" TO "EMAIL"
```



For details about the specification format and rules for the `ALTER TABLE` statement, see *ALTER TABLE (alter table definition)* in the manual *HADB SQL Reference*.

Note that changing the column name of a base table might result in the following adverse affects. Check whether there is any adverse effect.

- When there are viewed tables that depend on the base table whose column name was changed

  The viewed tables that depend on the base table whose column name was changed are invalidated. Such viewed tables are invalidated regardless of whether the old column name (that was used before the column name was changed) is explicitly specified when viewed tables are defined.

  To check the viewed tables that are to be invalidated (dependent viewed tables), see 11.2.11 Checking dependent viewed tables. If there is a viewed table that is invalidated, after using the `ALTER TABLE` statement to change the column name, you need to release the invalid status of the viewed table.

  For details about how to release a viewed table from invalid status, see (2) When viewed tables are invalidated by using an ALTER TABLE statement to change a column name in a table in 11.2.8 Releasing a viewed table from invalidation.

- When the old column name is explicitly specified in an SQL statement

  You need to modify the old column name specified in an SQL statement to the new column name.

- When the column structure information file is specified for the `-r` option of the `adbimport` command

  If the old column name is specified by the column structure information option for the column structure information file, you need to modify the column name to the new column name.

You cannot use the `ALTER TABLE` statement to rename a column of the base table in the following situations:

- When the target base table is non-updatable

  In this case, after releasing the non-updatable status of the base table, execute the `ALTER TABLE` statement to change the column name. For details about how to release a base table from non-updatable status, see (1) Releasing a base table from non-updatable status in 15.8.1 Steps to take when a base table becomes non-updatable.

11.1.4 Retrieving and updating data in a base table

This subsection describes how to retrieve and update data in a base table by using SQL statements.

To retrieve data in a base table

To retrieve data from a base table, execute the `SELECT` data manipulation SQL statement.

For details about the `SELECT` statement, see *SELECT (retrieve rows)* in the manual *HADB SQL Reference*.

Adding data to a base table (by inserting rows)

To add data to a base table, execute the `INSERT` data manipulation SQL statement.

For details about the `INSERT` statement in *INSERT (insert rows)* in the manual *HADB SQL Reference*.

> **Important**
>
> If you add a large amount of data to a column store table, use the `adbimport` command rather than the `INSERT` statement. If you use the `INSERT` statement to add data to a column store table, the added data is stored in row store format. This might degrade the retrieval performance for the column store table. This is why we recommend that you use the `adbimport` command when adding a large amount of data to a column store table. For details about how to add data by using the `adbimport` command, see 11.1.7 Storing data in a base table (data import).

To update data in a base table (by updating rows)

To update data stored in a base table, execute the `UPDATE` data manipulation SQL statement.

For details about the `UPDATE` statement, see *UPDATE (update rows)* in the manual *HADB SQL Reference*.

Deleting data in a base table (by deleting rows)

To delete data stored in a base table, execute the `DELETE` data manipulation SQL statement.

For details about the `DELETE` statement, see *DELETE (delete rows)* in the manual *HADB SQL Reference*.

> **Note**
>
> When you delete all the data stored in a base table, we recommend that you use the `TRUNCATE TABLE` statement rather than using the `DELETE` statement.
>
> For details about the `TRUNCATE TABLE` statement, see 11.1.5 Deleting all rows from a base table.

> **Important**
>
> If you use an SQL statement (`INSERT`, `UPDATE`, or `DELETE` statement) to repeatedly perform data addition, update, or deletion for a column store table, the retrieval performance for the column store table might be degraded. To prevent degradation of retrieval performance, we recommend that you take the following actions:
>
> - Enable the updated-row columnizing facility.
> - Periodically obtain the *information about the need for reorganization* of the table or chunk to check whether the table or chunk needs to be reorganized.
>
> For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables).

For details about how to check whether the table or chunk needs to be reorganized, see 11.1.9 Checking whether a single-chunk table needs to be reorganized or 11.4.13 Checking whether a multi-chunk table needs to be reorganized.

11.1.5 Deleting all rows from a base table

If you want to delete all rows from a base table, we recommend that as a rule, you execute the TRUNCATE TABLE data manipulation SQL statement.

Executing the TRUNCATE TABLE statement allows you to delete all rows more quickly than using the DELETE statement (with the WHERE clause omitted). Moreover, since segments are released, you can reuse the area where the deleted row data was stored.

The following table shows the differences between the TRUNCATE TABLE and DELETE statements.

Table 11-1: Differences between TRUNCATE TABLE and DELETE statements

No.	Compared item	TRUNCATE TABLE statement	DELETE statement (with the WHERE clause omitted)
1	Processing performance	High-speed Can delete all rows more quickly than the DELETE statement.	Low-speed Deleting all rows takes longer compared with the TRUNCATE TABLE statement.
2	Releasing of segments	Released The area that stored the deleted row data can be reused.	Not released Although the deleted rows become invalid, they are not deleted from the disk. Consequently, the area that stored the deleted row data cannot be reused. If this statement is executed for a column store table, the number of used segments might increase because invalid row information pages need to be reserved.
3	Need to execute the COMMIT statement	Automatically committed When the processing terminates normally, it is committed as soon as it is completed. Execution of the COMMIT statement is unnecessary.	Not automatically committed The COMMIT statement must be executed after processing is complete.
4	Locking (whether the statement can be concurrently executed on the same table with the SELECT statement) #	Cannot be concurrently executed When the TRUNCATE TABLE statement is executed, it locks the table to be processed in exclusive mode. Therefore, the SELECT and TRUNCATE TABLE statements cannot be concurrently executed on the same table.	Can be concurrently executed The SELECT and DELETE statements can be concurrently executed on the same table.
5	Recommended execution timing	• When you want to delete all rows quickly • When you want to reuse the area that was allocated to the deleted row data	• When you want to execute deletion and retrieval concurrently on the same table • When you want to use the statement as an element of a transaction

#

For details about the locked resources that are reserved when the TRUNCATE TABLE and DELETE statements are executed, and their lock modes, see 2.10.4 Locked resources that are reserved and their lock modes.

The following shows a specification example of deleting all rows from a base table.

Specification example

All rows are deleted from the sales history table (SALESLIST).

```
TRUNCATE TABLE "SALESLIST"
```

For details about the specification format and rules for the TRUNCATE TABLE statement, see *TRUNCATE TABLE (delete all rows in a base table)* in *Data Manipulation SQL* in the manual *HADB SQL Reference*.

Important

Before you delete all rows, check whether you need to re-execute the command for the base table for which the target rows are defined (whether the base table is non-updatable). For details about the check method, see (1) Checking whether a base table is non-updatable in 10.9.2 Checking the status and usage of a base table.

If re-execution of the command is necessary

Re-execute the command. Then, delete all rows.

If you deleted all rows when command re-execution was necessary (when the base table was non-updatable)

Temporary work files created by the interrupted command might remain. To delete the temporary work files, you must release all base tables from the non-updatable status.

For details about deleting temporary work files, see (2) When there are unneeded temporary work files on the disk in 15.2.5 Steps to take in the event of a shortage of disk space for storing temporary work files during command execution.

11.1.6 Changing a single-chunk table to a multi-chunk table

To change a single-chunk table to a multi-chunk table, first, delete the single-chunk table. Then, redefine the table as either of the following multi-chunk tables:

- Regular multi-chunk table
- Archivable multi-chunk table

Important

- You cannot change a single-chunk table to an archivable multi-chunk table if the single-chunk table is a column store table.
- Before changing a single-chunk table to a multi-chunk table, check the notes by referring to 5.2.4 Points to consider in defining a multi-chunk table and 5.2.5 Points to consider in defining an archivable multi-chunk table [Row store table].

The following shows the procedure for changing a single-chunk table to a multi-chunk table.

Procedure:

1. Check the table definition of the single-chunk table that you want to change to a multi-chunk table.

   Check the specification content of the CREATE TABLE statement when the single-chunk table was defined. This information is used in step 4 for defining a multi-chunk table.

If the content that was specified for the CREATE TABLE statement when the single-chunk table was defined is unknown, see (28) Finding out base table definition information in B.22 Searching a dictionary table. By searching the dictionary table, you can check the specification content of the CREATE TABLE statement when a single-chunk table was defined.

2. Output all data in the single-chunk table.

   Use the adbexport command to output all data from the single-chunk table to an output data file.

3. Delete the single-chunk table.

   Use the DROP TABLE statement without specifying *drop-behavior* to delete the single-chunk table.

4. Define a multi-chunk table.

   Use the CREATE TABLE statement to define a multi-chunk table. Note the following points when you create the specification content to be used in the CREATE TABLE statement:

   - Use the specification content (checked in step 1) used in the CREATE TABLE statement when the single-chunk table was defined, and create a CREATE TABLE statement.
   - Add a chunk specification to the CREATE TABLE statement. Note that the contents of the chunk specification differ depending on whether a regular multi-chunk table or archivable multi-chunk table is to be defined. For details about the contents of a chunk specification, see 11.4.1 Defining a multi-chunk table.

5. Define an index for the multi-chunk table.

   If an index was defined for the single-chunk table that was deleted, use the CREATE INDEX statement to redefine the index for the multi-chunk table defined in step 4. Note the following points when you create a CREATE INDEX statement:

   - If the content that was specified for the CREATE INDEX statement when the index for the single-chunk table was defined is unknown, see (29) Finding out index definition information in B.22 Searching a dictionary table. By searching the dictionary table, you can check the contents of the CREATE INDEX statement that were used when the index for a single-chunk table was defined.
   - You cannot define a unique index for a multi-chunk table.

6. Store data in the multi-chunk table.

   Use the adbimport command to store the data output in step 2 in the multi-chunk table that was redefined in step 4. Do not specify the -d or -b option for the adbimport command.

7. Re-validate viewed tables.

   The viewed tables whose underlying table is the single-chunk table that was deleted in step 3 are invalidated. Therefore, after defining a multi-chunk table, you must re-validate the viewed tables. When you re-validate the viewed tables, see (4) When viewed tables are invalidated due to erroneous deletion of a table in 11.2.8 Releasing a viewed table from invalidation.

Note

For details about the adbexport and adbimport commands, see the manual *HADB Command Reference*.

For details about the DROP TABLE, CREATE TABLE, and CREATE INDEX statements, see the manual *HADB SQL Reference*.

11.1.7 Storing data in a base table (data import)

To store data in a base table, use the `adbimport` command to perform data import. For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

Note that the data import method to be used differs depending on the option specified in the `adbimport` command. The following describes the data import methods that you can use when executing the `adbimport` command:

- **Data import in creation mode**

  Data import in creation mode is a data import method that deletes all the existing data in a base table, and then stores new data in the base table. If the target table is a multi-chunk table, this method deletes all the existing chunks, creates a new chunk, and then imports data into the chunk. Therefore, use data import in creation mode when you want to replace all the data in a base table with new data.

  To perform data import in creation mode, for the target base table, execute the `adbimport` command with the `-d` specified.

  > **Note**
  >
  > If you execute the command for a newly defined table, the command executes data import in creation mode regardless of whether the `-d` option is specified.

- **Background import**

  This is a data import method that can be used for multi-chunk tables. Background import creates a new chunk, and then imports data into the chunk. While data is being imported, you can retrieve data from the existing chunks. Therefore, use background import if you want to perform data import and retrieval at the same time.

  To perform background import, for the target multi-chunk table, execute the `adbimport` command by specifying the `-b` option. For details about how to perform background import, see 11.4.2 Storing data in a multi-chunk table (background import).

- **Data import in addition mode**

  Data import in addition mode is a method that adds data to a base table without deleting the data existing in the base table. If the target table is a multi-chunk table, this method adds new data to the current chunk without deleting the data in the existing chunks. At this time, if there are segments containing free pages, data is first added to those segments. However, if the target table is a column store table, data is always stored in a new segment even when there are segments that contain free pages. Note that while data import in addition mode is being performed, you cannot retrieve data from the target table.

  To perform data import in addition mode, for the target base table, execute the `adbimport` command without specifying the `-d` or `-b` option.

  > **Note**
  >
  > When importing a small amount of data, using addition mode provides more efficient data storage than background import. Therefore, if a need for importing a small amount of data arises during, for example, after-hours table maintenance, consider using addition mode for the import.

For details about the `-d` and `-b` options, see *Explanation of the specification format and options* in *Specification format for the adbimport command* under *adbimport (Import Data)* in the manual *HADB Command Reference*.

11.1.8 Outputting data from a base table to a file (data export)

You can output data from a base table to a file by using either of the following two methods:

- **Executing the `adbexport` command**

  You can output data from a base table to a file by executing the `adbexport` command.

  - To output all the data that is stored in a base table, execute the `adbexport` command by specifying *table-to-be-processed* for the `-n` option.

    The command outputs the data that is stored in the table specified for the `-n` option.
  - To output only the data that satisfies specific conditions from the target base table, execute the `adbexport` command by specifying the `-q` option.

    The command executes a search based on the SQL statement written in the SQL statement file specified for the `-q` option, and outputs the result of the search.

  The `adbexport` command can output data in a shorter time than the `adbsql` command. Note, however, that the `adbexport` command can be executed by the HADB server, but cannot be executed by an HADB client.

  For details about the `adbexport` command, see *adbexport (Export Data)* in the manual *HADB Command Reference*.
- **Executing the `adbsql` command**

  When you use the `adbsql` command to search a base table for data and redirect the search result, you can output data from the base table to a file.

  Note that the `adbsql` command can be executed by the HADB server and by an HADB client (Linux version).

  For details about the `adbsql` command, see *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

Normally, we recommend that you use the `adbexport` command when outputting data from a base table. Use the `adbsql` command only when an HADB client (Linux version) needs to output data from a base table to a file.

> **Important**
>
> To output data from a chunk in wait status, perform either of the following operations:
>
> - Outputting the data that is stored in a chunk in wait status by specifying the chunk ID
>
>   If you specify a chunk ID in the `adbexport` or `adbsql` command, the command can also output data from a chunk in wait status. For details, see 11.4.5 Exporting data in units of chunks.
> - Changing a chunk in wait status to a chunk in normal status
>
>   If a chunk is in normal status, both of the preceding two methods can output data from the chunk. Therefore, use the `adbchgchunkstatus` command to change the status of the chunk from wait status to normal status. For details, see 11.4.12 Changing the chunk status.

11.1.9 Checking whether a single-chunk table needs to be reorganized

This section describes how to check whether a single-chunk table needs to be reorganized.

> **Note**
>
> For details about how to check whether a multi-chunk table needs to be reorganized, see 11.4.13 Checking whether a multi-chunk table needs to be reorganized.

(1) Reason why table reorganization is necessary

If an SQL statement that inserts, updates, or deletes a row is repeatedly executed for a single-chunk table, the retrieval performance and data storage efficiency of the table are degraded. In such a case, if you reorganize the table, the area that has become unavailable in the data DB area is released, thus improving the retrieval performance and data storage efficiency of the table.

Retrieval performance and data storage efficiency of a single-chunk table might be degraded for the following reasons.

In a case where the single-chunk table is a row store table

When a row is deleted by the `DELETE` statement, the data on the deleted row becomes invalid. When a row is updated by the `UPDATE` statement, the resulting row data is added and the previous row data becomes invalid. Invalid row data is not deleted automatically from the disk.

If you repeat deletion and update of rows, invalid row data increases in the table. If invalid row data increases, it might affect as follows:

- Data storage efficiency drops due to the increase in the amount of invalid row data.
- Retrieval performance slows down because the number of pages to be referenced increases during retrieval processing.

In a case where the single-chunk table is a column store table

- When the `INSERT` statement is used to insert rows or the `UPDATE` statement is used to update rows, the resulting data is stored in row store format. If you repeat insertion and update of rows, the amount of data that is stored in row store format increases. Increase of data stored in row store format will degrade the performance of retrieval processing that is suitable for a column store table. Examples of such processing include retrieval for a specific column on all rows and retrieval for a specific column on the rows within a specific range, such as a year or month.
- When rows are deleted by the `DELETE` statement, the information about the deleted rows is stored on invalid row information pages. Also, when rows are updated by the `UPDATE` statement, the information about the rows that existed before they were updated is stored on invalid row information pages.

  If you repeat deletion and update of rows, the number of invalid row information pages increases and the retrieval performance is degraded because the number of pages to be referenced during retrieval increases.

If the single-chunk table is a row store table, check whether table reorganization is necessary by referring to (2) If the single-chunk table is a row store table.

If the single-chunk table is a column store table, check whether table reorganization is necessary by referring to (3) If the single-chunk table is a column store table.

(2) If the single-chunk table is a row store table

Determine the data storage efficiency, and if it is poor, reorganize the table. Use the following formula to determine the data storage efficiency:

Formula

Data storage efficiency = *A* / (*A* + *B*)

Explanation of variables

A: Number of rows in the table

B: Number of invalid rows

The data storage efficiency is better the closer that the value obtained from the formula is to 1. Conversely, the data storage efficiency is worse the closer that the value obtained from the formula is to 0. When the obtained value is close to 0, reorganize the data in the table as explained in 11.1.10 Reorganizing a single-chunk table.

Determine variables *A* and *B* from the following procedure.

▪ **Procedure (determining variables *A* and *B*)**

1. Make sure that the SQL trace information is set to be output.

   Make sure that the access path statistical information of the SQL trace information is output. You can check whether the access path statistical information is output by checking the specification of the server definition for SQL tracing. For details, see (2) Specifying server definitions in 10.11.5 Preparations for outputting SQL trace information.

2. Execute the following `SELECT` statement:

   ```
   SELECT COUNT(*) FROM "table-name" /*>> WITHOUT INDEX <<*/
   ```

   The execution result of the preceding `SELECT` statement becomes the number of rows in the table (the value of variable *A*).

3. Check the access path statistical information.

   Check the access path statistical information for the `SELECT` statement executed in step 2. Check the value for `Data_deleted_rows_cnt` in the access path statistical information. This value becomes the number of invalid rows (the value of variable *B*).

If the number of invalid rows (the value of variable *B*) is unknown, use the following procedure to determine the data storage efficiency.

▪ **Procedure (when the value of variables *B* is unknown)**

1. Check the number of segments used by the table.

   Use the `adbdbstatus` command to output summary information of the table. Then, check the output value for `Used_segments` (number of segments used by the table). For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

2. Obtain the number of segments in the data DB area based on the number of rows in the table.

   First obtain the number of rows in the table, and then obtain the number of segments in the data DB area. For details about how to obtain the number of rows in the table, see the explanation of variable *A* above.

   For details about how to determine the number of segments in the data DB area, see (a) Determining the variable SGROWTBL (for a single-chunk table) in (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

   The variable *SGROWTBL* is a formula to obtain the summation results for all tables that are stored in the data DB area. In this step, obtain the value only for the target table. The number of rows in the table is used in variables *BP*(*i*) and *VP*(*i*), which are used in the formula for variable *SGROWTBL*.

3. Determine the data storage efficiency based on the obtained results.

   Determine the data storage efficiency based on the results of steps 1 and 2. The following shows the formula:

   Formula

   data storage efficiency = result of step 2 ÷ result of step 1

   If the result of the preceding calculation is close to 0, reorganize the table as explained in 11.1.10 Reorganizing a single-chunk table.

(3) If the single-chunk table is a column store table

Execute the `adbdbstatus` command, check *information about the need for reorganization*, and determine whether to reorganize the single-chunk table. Use the following procedure for determination.

Procedure

1. Execute the `adbdbstatus` command with the `-d reorginfo` option specified to output information about the need for reorganization.
2. In the information about the need for reorganization, check the `Reorganization_necessity` section (whether reorganization is necessary).
   - If `Recommended` is output

     Reorganize the single-chunk table. For details about the reorganization procedure, see 11.1.10 Reorganizing a single-chunk table.
   - If `Not_recommended` is output

     You do not need to reorganize the single-chunk table.

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

11.1.10 Reorganizing a single-chunk table

This section describes how to reorganize a single-chunk table.

> **Important**
>
> Do not execute an update SQL statement or a command that updates a table (such as the `adbimport` or `adbmergechunk` command) for a table that is being reorganized. If you do so, the changes made by the SQL statement or command might be lost after reorganization.

(1) Reorganization procedure

The following shows the procedure for reorganizing a single-chunk table.

Procedure

1. Make sure that no application programs and commands can access the reorganization-target table.

   Perform the following operations to prevent application programs and commands from accessing the reorganization-target table:
   - Execute the `adbchgsrvmode` command with the `--offline` option specified to change the HADB server operation mode to offline mode. Furthermore, if commands and jobs that update the table (such as the `adbimport` and `adbmergechunk` commands that are periodically executed) are executing on the HADB server machine, stop all of them.
   - Stop all application programs and commands that access the HADB server.

   We recommend that you perform both of the preceding operations.
2. Export the data from the table.

Export the data of the table by executing the adbexport command in which the reorganization-target table is specified for the -n option.

For details about the adbexport command, see *adbexport (Export Data)* in the manual *HADB Command Reference*.

3. Import the exported data in the creation mode.

   Execute the adbimport command with the -d option specified to import the data that was exported in step 2. If you specify the -d option, the data is imported in the creation mode. (In this mode, all the existing data in the table is deleted, and then the data exported in step 2 is imported.)

   For details about the adbimport command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

   > **Important**
   >
   > When you execute the adbimport command, make sure that the value you specify for the import option (adb_import_rthd_num) satisfies the formula shown in (2) Estimating the value to be specified for the import option. If you specify a value that does not satisfy the formula, the data storage efficiency might degrade.

4. Make sure that application programs and commands can access the reorganization-target table.

   Perform the following operation to enable application programs and commands to access the reorganization-target table:

   - If you changed the HADB server's operation mode to offline mode in step 1

     Execute the adbchgsrvmode command with the --normal option specified to change the HADB server operation mode to normal mode. If you stopped commands and jobs that were running on the HADB server machine, restart them.

   - If you stopped application programs and commands that access the HADB server in step 1

     Restart the application programs and commands that you stopped.

> **Note**
>
> For details about how to reorganize a multi-chunk table, see the following sections:
>
> - 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization
> - 11.4.15 Reorganizing a multi-chunk table: Reorganization of an entire table
> - 11.4.16 Reorganizing a multi-chunk table: Reorganization using a sample shell script

(2) Estimating the value to be specified for the import option

■ In a case where the single-chunk table is a row store table

When you execute the adbimport command, make sure that the value you specify for the import option (adb_import_rthd_num) satisfies the following formula. If you specify a value that does not satisfy the formula, the data storage efficiency might degrade.

Formula

```
value-specified-for-import-option-adb_import_rthd_num ≤
    ↑ number-of-segments-to-be-reorganized × data-storage-efficiency ↑ + 1
```

number-of-segments-to-be-reorganized

Substitute the number of segments used in the table.

Before you substitute the number of segments used in the table, execute the `adbdbstatus` command to output the *table summary information*. Then, check the output value for `Used_segments` (number of segments used by the table).

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

data-storage-efficiency

Substitute the value obtained in (2) If the single-chunk table is a row store table in 11.1.9 Checking whether a single-chunk table needs to be reorganized.

■ **In a case where the single-chunk table is a column store table**

When you execute the `adbimport` command, make sure that the value you specify for the import option (`adb_import_rthd_num`) satisfies the following formula. If you specify a value that does not satisfy the formula, the data storage efficiency might degrade.

Formula

```
value-specified-for-import-option-adb_import_rthd_num ≤
    ↑ number-of-segments-to-be-reorganized × number-of-rows-to-be-reorganized
       ÷ number-of-rows-stored-in-column-data-segment ↑ + 1
```

number-of-segments-to-be-reorganized

Substitute the number of column-data segments used in the table.

To check the number of column-data segments used in the table, execute the `adbdbstatus` command with the `-d used` option specified to output *usage information for DB areas, tables, and indexes*. Then, check the information output as `Used_segments` (number of used segments) on each row whose `Segment_type` (segment type) is `Column_data` (column-data segment).

number-of-rows-to-be-reorganized

Substitute the number of rows to be reorganized. You can check the number of rows in the table by executing the following `SELECT` statement:

```
SELECT COUNT(*) FROM "table-name"
```

number-of-rows-stored-in-column-data-segment

Substitute the number of rows that were imported into the reorganization-target table by using the `adbimport` command. Alternatively, substitute the number of rows that were converted from row store format to column store format by using the updated-row columnizing facility.

Execute the `adbdbstatus` command to obtain *information about the need for reorganization*. The value of `Column_data_num` is the number of rows stored in the column-data segment.

11.1.11 Changing a row store table to a column store table

To change a row store table to a column store table, you must first delete the row store table, and then redefine the table as a column store table.

Important

Note that some restrictions that are not placed on row store tables are placed on column store tables. Therefore, before you change a row store table to a column store table, check the restrictions on column

store tables. For details about the restrictions on column store tables, see (1) Restrictions on column store tables in 5.2.2 Criteria for selecting row store tables and column store tables.

Note

For details about how to change a column store table to a row store table, see 11.1.12 Changing a column store table to a row store table.

Change a row store table to a column store table by using the following procedure:

Procedure

1. Output all data that is stored in the row store table.
   Use the `adbexport` command to output all data that is stored in the row store table to a file.
   If the row store table is a multi-chunk table, note the following points when executing the `adbexport` command:
   - **If the status and configuration of the chunks in the multi-chunk table do not need to be retained**
     No notes apply. Output all data that is stored in the row store table to a file.
   - **If the status and configuration of the chunks in the multi-chunk table need to be retained**
     You must output all data from the row store table to a file in units of chunks. Unless you output data in units of chunks, the status and configuration of the chunks will not be retained in the column store table that will store the data. Therefore, you must perform an export for each chunk.
     For details about outputting data for each chunk, see 11.4.5 Exporting data in units of chunks.
2. Check the definition information of the row store table.
   Check the specification content of the `CREATE TABLE` statement that was used to define the row store table. You will use the specification content when you define the column store table.
   If you are not sure of the specification content of the `CREATE TABLE` statement that was used to define the row store table, see (28) Finding out base table definition information in B.22 Searching a dictionary table. By retrieving data from a dictionary table, you can check the specification content of the `CREATE TABLE` statement that was used to define the row store table.
3. Check the definition information of the indexes defined for the row store table.
   If indexes have been defined for the row store table, check the specification content of the `CREATE INDEX` statement that was used to define the indexes. You will use the specification content when you define indexes for the column store table.
   If you are not sure of the specification content of the `CREATE INDEX` statement that was used to define the indexes, see (29) Finding out index definition information in B.22 Searching a dictionary table. By retrieving data from a dictionary table, you can check the specification content of the `CREATE INDEX` statement that was used to define the indexes.
4. Delete the row store table.
   Use the `DROP TABLE` statement without specifying *drop-behavior* to delete the row store table.
5. Define a column store table.
   Use the `CREATE TABLE` statement to define a column store table. Note the following points when you create a `CREATE TABLE` statement:
   - Create a `CREATE TABLE` statement with the specification content of the `CREATE TABLE` statement that was used to define the row store table (the specification content you checked in step 2).
   - Specify `COLUMN` for `STORAGE FORMAT` in the `CREATE TABLE` statement.

6. Define indexes for the column store table.

   If the row store table that you deleted had indexes defined, use the `CREATE INDEX` statement to define indexes for the column store table. Note the following points when you create a `CREATE INDEX` statement:

   - Create a `CREATE INDEX` statement with the specification content of the `CREATE INDEX` statement that was used to define the indexes (the specification content you checked in step 3).

     Note that if a B-tree index was defined for the row store table, we recommend that you do not define a B-tree index for the column store table. For details, see (3) B-tree indexes used when retrieving data from column store tables in 5.2.2 Criteria for selecting row store tables and column store tables.
   - You cannot define a text index for column store tables.

7. Store data in the column store table.

   Use the `adbimport` command to store all of the data that was output in step 1 to the column store table defined in step 5.

   If the column store table is a multi-chunk table, note the following points when executing the `adbimport` command:

   - **If the status and configuration of the chunks in the multi-chunk table do not need to be retained**

     Do not specify the `-d` or `-b` option for the `adbimport` command.
   - **If the status and configuration of the chunks in the multi-chunk table need to be retained**

     You must import for each chunk the data that was output in units of chunks in step 1.

     When you perform the first data import, execute the `adbimport` command without specifying the `-d` or `-b` option.

     When you perform the second and following data imports, use the background import method. Execute the `adbimport` command by specifying the `-b` option.

     When you perform the last data import, import the data stored in the chunk that was the current chunk in step 1 by using the background import method. Execute the `adbimport` command by specifying the `-b` option.

8. Re-validate viewed tables.

   The viewed tables whose underlying table is the row store table that was deleted in step 4 are invalidated. Therefore, after defining a column store table, you must re-validate the viewed tables. When you re-validate the viewed tables, see (4) When viewed tables are invalidated due to erroneous deletion of a table in 11.2.8 Releasing a viewed table from invalidation.

The procedure for changing a row store table to a column store table is complete.

> **Note**
>
> - For details about the `adbexport` and `adbimport` commands, see the manual *HADB Command Reference*.
> - For details about the `DROP TABLE`, `CREATE TABLE`, and `CREATE INDEX` statements, see the manual *HADB SQL Reference*.

11.1.12 Changing a column store table to a row store table

To change a column store table to a row store table, you must first delete the column store table, and then redefine the table as a row store table.

Important

Note that the number of items that can be specified during table definition for a row store table is larger than for a column store table. Therefore, there are some points you must note before a change to a row store table. For details, see 5.2.1 Flow of table design.

Note

For details about how to change a row store table to a column store table, see 11.1.11 Changing a row store table to a column store table.

Change a column store table to a row store table by using the following procedure:

Procedure

1. Output all data that is stored in the column store table.

   Use the `adbexport` command to output all data that is stored in the column store table to a file.

   If the column store table is a multi-chunk table, note the following points when executing the `adbexport` command:

   - **If the status and configuration of the chunks in the multi-chunk table do not need to be retained**

     No notes apply. Output all data that is stored in the column store table to a file.
   - **If the status and configuration of the chunks in the multi-chunk table need to be retained**

     You must output all data from the column store table to a file in units of chunks. Unless you output data in units of chunks, the status and configuration of the chunks will not be retained in the row store table that will store the data. Therefore, you must perform an export for each chunk.

     For details about outputting data for each chunk, see 11.4.5 Exporting data in units of chunks.
2. Check the definition information of the column store table.

   Check the specification content of the `CREATE TABLE` statement that was used to define the column store table. You will use the specification content when you define the row store table.

   If you are not sure of the specification content of the `CREATE TABLE` statement that was used to define the column store table, see (28) Finding out base table definition information in B.22 Searching a dictionary table. By retrieving data from a dictionary table, you can check the specification content of the `CREATE TABLE` statement that was used to define the column store table.
3. Check the definition information of the indexes defined for the column store table.

   If indexes have been defined for the column store table, check the specification content of the `CREATE INDEX` statement that was used to define the indexes. You will use the specification content when you define indexes for the row store table.

   If you are not sure of the specification content of the `CREATE INDEX` statement that was used to define the indexes, see (29) Finding out index definition information in B.22 Searching a dictionary table. By retrieving data from a dictionary table, you can check the specification content of the `CREATE INDEX` statement that was used to define the indexes.
4. Delete the column store table.

   Use the `DROP TABLE` statement without specifying *drop-behavior* to delete the column store table.
5. Define a row store table.

   Use the `CREATE TABLE` statement to define a row store table. Note the following points when you create a `CREATE TABLE` statement:

- Create a CREATE TABLE statement with the specification content of the CREATE TABLE statement that was used to define the column store table (the specification content you checked in step 2).
- Specify ROW for STORAGE FORMAT in the CREATE TABLE statement.

6. Define indexes for the row store table.

   If the column store table that you deleted had indexes defined, use the CREATE INDEX statement to define indexes for the row store table. Note the following points when you create a CREATE INDEX statement:

   - Create a CREATE INDEX statement with the specification content of the CREATE INDEX statement that was used to define the indexes (the specification content you checked in step 3).
   - When you create a CREATE INDEX statement, remember that you might need to define B-tree indexes and text indexes if necessary.

     For column store tables, it is not preferable to define B-tree indexes. It is impossible to define text indexes. Because only range indexes are defined in normal cases, keep this in mind when you create a CREATE INDEX statement.

7. Store data in the row store table.

   Use the adbimport command to store all of the data that was output in step 1 to the row store table defined in step 5.

   If the row store table is a multi-chunk table, note the following points when executing the adbimport command:

   - **If the status and configuration of the chunks in the multi-chunk table do not need to be retained**

     Do not specify the -d or -b option for the adbimport command.
   - **If the status and configuration of the chunks in the multi-chunk table need to be retained**

     You must import for each chunk the data that was output in units of chunks in step 1.

     When you perform the first data import, execute the adbimport command without specifying the -d or -b option.

     When you perform the second and following data imports, use the background import method. Execute the adbimport command by specifying the -b option.

     When you perform the last data import, import the data stored in the chunk that was the current chunk in step 1 by using the background import method. Execute the adbimport command by specifying the -b option.

8. Re-validate viewed tables.

   The viewed tables whose underlying table is the column store table that was deleted in step 4 are invalidated. Therefore, after defining a row store table, you must re-validate the viewed tables. When you re-validate the viewed tables, see (4) When viewed tables are invalidated due to erroneous deletion of a table in 11.2.8 Releasing a viewed table from invalidation.

The procedure for changing a column store table to a row store table is complete.

Note

- For details about the adbexport and adbimport commands, see the manual *HADB Command Reference*.
- For details about the DROP TABLE, CREATE TABLE, and CREATE INDEX statements, see the manual *HADB SQL Reference*.

11.1.13 Checking the definition information for a base table (searching a dictionary table)

By searching a dictionary table, you can check the definition information of a base table.

> **Note**
>
> For details about the contents of each dictionary table, see B. Dictionary Tables. For an example of searching a dictionary table, see B.22 Searching a dictionary table.

The following table lists the dictionary tables and the base table definition information that can be checked from the dictionary tables.

Table 11-2: Dictionary tables and the base table definition information that can be checked

No.	Dictionary table	Base table definition information that can be checked
1	SQL_TABLES table	By searching the SQL_TABLES table, you can check the following definition information: • Table name Check the TABLE_SCHEMA column (schema name) and TABLE_NAME column (table identifier). • Specification content of the PCTFREE table option Check the FREE_AREA column (percentage of unused area in the table pages). • Whether the BRANCH ALL table option is specified Check the IS_BRANCH_ALL column (specification of the BRANCH ALL table option). • Table type (single-chunk table or multi-chunk table) Check the specification of the IS_CHUNK column (CHUNK specification in the chunk specification). • Maximum number of chunks created in a multi-chunk table Check the N_CHUNK_RESERVED column (maximum number of chunks that can be created). • Whether the column is an archivable multi-chunk table Check the IS_ARCHIVABLE column (whether a chunk-archive specification exists). • Path name of the archive directory Check the ARCHIVE_DIRECTORY_PATH column (absolute path to the archive directory). • Table type (row store table or column store table) Check the STORAGE_FORMAT column (table-data storage format). For details about the SQL_TABLES table, see B.2 Content of SQL_TABLES.
2	SQL_COLUMNS table	By searching the SQL_COLUMNS table, you can check the following definition information: • Column name Check the COLUMN_NAME column (column name). • Data type and definition length of a column Check the DATA_TYPE_CODE column (data type of a column) and DATA_LENGTH column (definition length of a column). • Specification content of the DEFAULT clause Check the IS_DEFAULT_COLUMN column (whether the DEFAULT clause is specified) and DEFAULT_VALUE column (default value specified for the DEFAULT clause). • Whether BRANCH is specified Check the BRANCH column (value specified for BRANCH in the column definition). • Specification content of a primary key Check the IS_PRIMARY_KEY_COLUMN column (whether the column is a member of columns that comprise the primary key), and the PRIMARY_KEY_COLUMN_SEQUENCE_NUMBER column (column order of the primary key).

No.	Dictionary table	Base table definition information that can be checked
		• Whether the column is an archive range column Check the IS_ARCHIVE_RANGE_COLUMN column (whether the column is an archive range column). • Value specified as the column-data compression type during definition of a column store table Check the COMPRESSION_TYPE column (value specified for COMPRESSION TYPE in the column definition). For details about the SQL_COLUMNS table, see B.3 Content of SQL_COLUMNS.
3	• SQL_DIV_TABLE table • SQL_DBAREAS table	By searching the tables on the left, you can check the following definition information: • DB area ID of the DB area that stores the table Check the DBAREA_ID column of the SQL_DIV_TABLE table (DB area ID of the DB area that stores the table). • DB area name of a DB area Check the DBAREA_NAME column (DB area name) of the SQL_DBAREAS table based on the DB area ID checked in the SQL_DIV_TABLE table. For details about the SQL_DIV_TABLE table, see B.4 Content of SQL_DIV_TABLE. For details about the SQL_DBAREAS table, see B.7 Content of SQL_DBAREAS.
4	• SQL_TABLE_CONSTRAINTS table • SQL_KEY_COLUMN_USAGE table • SQL_REFERENTIAL_CONSTRAINTS table	By searching the tables on the left, you can check the following definition information: • Information related to constraints on a base table (whether primary keys and foreign keys exist) See the SQL_TABLE_CONSTRAINTS table. For details about the SQL_TABLE_CONSTRAINTS table, see B.14 Content of SQL_TABLE_CONSTRAINTS. • Information related to columns comprising the primary key and foreign keys See the SQL_KEY_COLUMN_USAGE table. For details about the SQL_KEY_COLUMN_USAGE table, see B.16 Content of SQL_KEY_COLUMN_USAGE. • Information related to referential constraints See the SQL_REFERENTIAL_CONSTRAINTS table. For details about the SQL_REFERENTIAL_CONSTRAINTS table, see B.17 Content of SQL_REFERENTIAL_CONSTRAINTS.

11.1.14 Changing the data DB area that stores a base table

To change the data DB area that stores a base table to another data DB area, you must first delete the base table, and then redefine the base table.

> **Note**
>
> For details about how to change the data DB area that stores indexes to another data DB area, see 11.3.8 Changing the data DB area that stores indexes.

By using the procedure shown later, change the data DB area that stores a base table to another data DB area.

You might also want to add a new data DB area. For details, see 11.10.1 Adding data DB areas. After seeing the topic and taking action, change the data DB area that stores a base table.

Procedure

1. Output all data that is stored in the base table.

Use the adbexport command to output all data that is stored in the base table to a file.

If the base table is a multi-chunk table, note the following points when executing the adbexport command:

- **If the status and configuration of the chunks in the multi-chunk table do not need to be retained**

  No notes apply. Output all data that is stored in the base table to a file.

- **If the status and configuration of the chunks in the multi-chunk table need to be retained**

  You must output all data from the base table to a file in units of chunks. Unless you output data in units of chunks, the status and configuration of the chunks will not be retained in the base table that will store the data. Therefore, you must perform an export for each chunk.

  For details about outputting data for each chunk, see 11.4.5 Exporting data in units of chunks.

2. Check the definition information of the base table.

   Check the specification content of the CREATE TABLE statement that was used to define the base table. You will use the specification content when you define a base table in another data DB area.

   If you are not sure of the specification content of the CREATE TABLE statement that was used to define the base table, see (28) Finding out base table definition information in B.22 Searching a dictionary table. By searching the dictionary table, you can check the specification content of the CREATE TABLE statement that was used to define the base table.

3. Check the definition information of the indexes defined for the base table.

   If indexes have been defined for the base table, check the specification content of the CREATE INDEX statement that was used to define the indexes. You will use the specification content when you define indexes for a base table defined in another data DB area.

   If you are not sure of the specification content of the CREATE INDEX statement that was used to define the indexes, see (29) Finding out index definition information in B.22 Searching a dictionary table. By retrieving data from a dictionary table, you can check the specification content of the CREATE INDEX statement that was used to define the indexes.

4. Delete the base table.

   Use the DROP TABLE statement without specifying *drop-behavior* to delete the base table.

5. Redefine a base table.

   Use the CREATE TABLE statement to define a base table in another data DB area. Note the following points when you create a CREATE TABLE statement:

   - Create a CREATE TABLE statement with the specification content of the CREATE TABLE statement that was used to define the base table (the specification content you checked in step 2).
   - Specify the DB area name of another data DB area that will store the base table for IN *DB-area-name* in the CREATE TABLE statement.

6. Define indexes for the base table.

   If the base table that you deleted had indexes defined, use the CREATE INDEX statement to define indexes for the redefined base table. Note the following points when you create a CREATE INDEX statement:

   - Create a CREATE INDEX statement with the specification content of the CREATE INDEX statement that was used to define the indexes (the specification content you checked in step 3).

7. Store data in the base table.

   Use the adbimport command to store all of the data that was output in step 1 to the base table defined in step 5.

   If the base table is a multi-chunk table, note the following points when executing the adbimport command:

   - **If the status and configuration of the chunks in the multi-chunk table do not need to be retained**

     Do not specify the -d or -b option for the adbimport command.

   - **If the status and configuration of the chunks in the multi-chunk table need to be retained**

You must import for each chunk the data that was output in units of chunks in step 1.

When you perform the first data import, execute the `adbimport` command without specifying the `-d` or `-b` option.

When you perform the second and following data imports, use the background import method. Execute the `adbimport` command by specifying the `-b` option.

When you perform the last data import, import the data stored in the chunk that was the current chunk in step 1 by using the background import method. Execute the `adbimport` command by specifying the `-b` option.

8. Re-validate viewed tables.

   The viewed tables whose underlying table is the base table that was deleted in step 4 are invalidated. Therefore, after defining a base table, you must re-validate the viewed tables. When you re-validate the viewed tables, see (4) When viewed tables are invalidated due to erroneous deletion of a table in 11.2.8 Releasing a viewed table from invalidation.

The procedure for changing a data DB area that stores a base table to another data DB area is complete.

Note

- For details about the `adbexport` and `adbimport` commands, see the manual *HADB Command Reference*.
- For details about the `DROP TABLE`, `CREATE TABLE`, and `CREATE INDEX` statements, see the manual *HADB SQL Reference*.

11.1.15 Deleting base tables

To delete a base table, execute the `DROP TABLE` definition SQL statement.

Important

- If you delete a base table, viewed tables that depend on it are deleted (or invalidated). Therefore, before you delete a base table, check whether there are viewed tables that depend on it, and make sure that there is no problem if those viewed tables are deleted or invalidated.
- If you delete a base table for which the primary key is defined, foreign keys that use the deleted base table as the referenced table are also deleted. (Foreign keys for other schemas are also deleted.) Therefore, before you delete a base table, make sure that there is no problem if the foreign keys are deleted. However, if you specify `RESTRICT` as the drop behavior, an error occurs during execution of the SQL statement.

The following shows an example of base table deletion.

Example

This example deletes a shop table (`SHOPSLIST`).

Procedure:

1. Check the viewed tables that depend on the base table to be deleted.

   By retrieving data from a dictionary table, you can check the viewed tables that depend on the base table to be deleted. For an example of retrieving data from a dictionary table, see 11.2.11 Checking dependent viewed tables.

2. Check the foreign keys that use the base table to be deleted as the referenced table.

   This step is required for deleting a base table for which the primary key is defined.

   By retrieving data from a dictionary table, you can check the foreign keys that use the base table to be deleted as the referenced table. For an example of retrieving data from a dictionary table, see (21) When checking foreign keys that reference a primary key in B.22 Searching a dictionary table.

3. Check whether the base table to be deleted is in the non-updatable status.

   Before you delete the base table, check whether you need to re-execute the command for that base table (whether the base table is non-updatable). For details about the check method, see (1) Checking whether a base table is non-updatable in 10.9.2 Checking the status and usage of a base table.

> **Important**
>
> **If re-execution of the command is necessary**
>
> Re-execute the command. Then, delete the base table.
>
> **If you deleted the base table when command re-execution was necessary (when the base table was non-updatable)**
>
> Temporary work files created by the interrupted command might remain. To delete the temporary work files, you must release all base tables from the non-updatable status. For details about deleting temporary work files, see (2) When there are unneeded temporary work files on the disk in 15.2.5 Steps to take in the event of a shortage of disk space for storing temporary work files during command execution.

4. Execute the `DROP TABLE` statement.

```
DROP TABLE "SHOPSLIST" CASCADE
```

> **Note**
>
> In the preceding example, `CASCADE` is specified for the drop behavior. Therefore, all viewed tables that depend on the base table to be deleted are deleted. If you omit specifying the drop behavior, all viewed tables that depend on the base table to be deleted are invalidated.

11.2 Viewed table operations

This section explains how to operate viewed tables.

For details about viewed tables, see 2.1.2 Viewed tables.

11.2.1 Defining a viewed table

To define a viewed table, execute the CREATE VIEW definition SQL statement.

The following shows an example of defining a viewed table.

Example

From the CUSTOMER and STOCK base tables, a viewed table named CUSTOMER30s is defined that lists the products purchased by customers in their 30's.

Figure 11-1: Example of CUSTOMER and STOCK base tables

■ CUSTOMER (Base table)

AGE (Age)	SEX (Sex)	GNO (Product No.)	PNUM (Purchases)
30	F	20180	1
20	F	20130	1
30	M	20240	1
40	F	20200	1
50	F	20240	1
50	M	20130	1
60	M	20200	1
20	F	20220	1
30	F	20150	1
50	M	20290	1
60	F	20250	1
50	F	20310	1
40	M	20210	1
20	F	20280	1
30	M	20180	1

■ STOCK (Base table)

GNO (Product No.)	PCODE (Product code)	PNAME (Product name)	PRICE (Unit price)	QUANTITY (Inventory quantity)
20180	C20	Blouse	2000	26
20190	C77	Blouse	2800	105
20130	P10	Shirt	3000	70
20230	K18	Sweater	3500	12
20200	C89	Blouse	3500	30
20140	P23	Shirt	3500	60
20280	L10	Socks	380	200
20150	P32	Shirt	4800	50
20290	L50	Socks	490	260
20240	K20	Sweater	5000	15
20160	P35	Shirt	5580	120
20250	K22	Sweater	7500	36
20300	L80	Socks	750	170
20210	C91	Blouse	3000	35
20260	K24	Sweater	6000	20
20310	L90	Socks	800	230
20220	C95	Blouse	2500	110

■ SQL statement for defining the viewed table (CUSTOMER30s table)

```
CREATE VIEW "CUSTOMER30s"
  ("AGE","SEX","GNO","PNAME","PRICE")
  AS SELECT
   "CUSTOMER"."AGE", "CUSTOMER"."SEX", "CUSTOMER"."GNO",
   "STOCK"."PNAME", "STOCK"."PRICE"
  FROM "CUSTOMER","STOCK"
    WHERE "CUSTOMER"."AGE"=30 AND "CUSTOMER"."GNO"="STOCK"."GNO"
```

Figure 11-2: Example of definition of the viewed table (CUSTOMER30s table)

■ CUSTOMER30s (Viewed table)

AGE (Age)	SEX (Sex)	GNO (Prod. No.)	PNAME (Prod. name)	PRICE (Unit price)
30	F	20180	Blouse	20.00
30	M	20240	Sweater	50.00
30	F	20150	Shirt	48.00
30	M	20180	Blouse	20.00

Whether the defined viewed table becomes an updatable viewed table or a read-only viewed table depends on what is specified in AS *query specification* in the CREATE VIEW statement. For details, see *CREATE VIEW (define a viewed table)* in *Definition SQL* in the manual *HADB SQL Reference*.

11.2.2 Retrieving, updating, adding, and deleting data in viewed tables

To make a retrieval from a viewed table, execute the SELECT data manipulation SQL statement. This is the same procedure used for a retrieval from a base table.

The following shows an example of a retrieval from a viewed table.

Example

The following SQL statement retrieves the rows in which SEX is F (female) from the CUSTOMER30s table defined in 11.2.1 Defining a viewed table.

```
SELECT * FROM "CUSTOMER30s" WHERE "SEX" = 'F'
```

If you have access privileges required for an updatable viewed table, you can update, add, and delete data in the viewed table. To update, add, or delete data in a viewed table, execute the data manipulation SQL statement UPDATE, INSERT, or DELETE, respectively. This is the same procedure as used for updating, adding, or deleting a base table.

11.2.3 Outputting data from a viewed table to a file (data export)

For details about how to output data from a viewed table to a file, see 11.1.8 Outputting data from a base table to a file (data export). The same procedure that is used to output data from a base table to a file applies.

11.2.4 Rebuilding a viewed table

To rebuild a viewed table, execute the ALTER VIEW definition SQL statement. For details about the ALTER VIEW statement, see *ALTER VIEW (re-create a viewed table)* in the manual *HADB SQL Reference*.

A user (the HADB user with the authorization identifier that was used for the current connection to the HADB server) can rebuild only those viewed tables owned by that user. A user cannot rebuild viewed tables owned by other HADB users.

After the cause of invalidation of a viewed table has been removed, if you rebuild the viewed table by using the ALTER VIEW statement, you can release invalidation of the viewed table. For details, see 11.2.8 Releasing a viewed table from invalidation.

Note that the ALTER VIEW statement cannot change an existing view definition.

Important

If you rebuild a viewed table, all viewed tables that depend on it are invalidated. Therefore, before you rebuild a viewed table, check whether there are viewed tables that depend on it, and make sure that there is no problem if those viewed tables are invalidated.

11.2.5 Deleting a viewed table

To delete a viewed table, execute the DROP VIEW definition SQL statement.

A user (the HADB user with the authorization identifier that was used for the current connection to the HADB server) can delete only those viewed tables owned by that user. A user cannot delete viewed tables owned by other HADB users.

Important

If you delete a viewed table, all viewed tables that depend on it are deleted (or invalidated). Therefore, before you delete a viewed table, check whether there are viewed tables that depend on it, and make sure that there is no problem if those viewed tables are deleted or invalidated.

The following shows an example of deleting a viewed table.

Example

The following procedure deletes the CUSTOMER30s table defined in 11.2.1 Defining a viewed table.

Procedure:

1. Check the viewed tables that depend on CUSTOMER30s.

   By retrieving data from a dictionary table, you can check the viewed tables that depend on the viewed table to be deleted. For an example of retrieving data from a dictionary table, see 11.2.11 Checking dependent viewed tables.

2. Execute the DROP VIEW statement to delete CUSTOMER30s.

```
DROP VIEW "CUSTOMER30s" CASCADE
```

Note

In the preceding example, CASCADE is specified for the drop behavior. Therefore, all viewed tables that depend on the viewed table to be deleted (CUSTOMER30s) are deleted. If you omit specifying the drop behavior, all viewed tables that depend on the viewed table to be deleted are invalidated.

11.2.6 Checking whether a viewed table is updatable

To check whether a viewed table can be updated using the INSERT, UPDATE, or DELETE statement, use the SELECT statement to search the SQL_VIEWS dictionary table.

The following shows an example of checking whether a viewed table is updatable.

Example:

This example checks whether the viewed table CUSTOMER30s, defined by ADBUSER01, is updatable.

```
SELECT "IS_UPDATABLE" FROM "MASTER"."SQL_VIEWS"
  WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='CUSTOMER30s'
```

If the search result is Y, the viewed table is updatable. If the search result is N, the viewed table is read-only and cannot be updated.

For details about dictionary table SQL_VIEWS, see B.9 Content of SQL_VIEWS.

11.2.7 Checking whether a viewed table has been invalidated

By retrieving data from a dictionary table, you can check whether a viewed table has been invalidated. The following shows an example of retrieving data from a dictionary table.

Example 1

This example checks whether viewed table CUSTOMER30s defined by HADB user ADBUSER01 has been invalidated.

Execute the following SELECT statement to make a retrieval from a dictionary table.

```
SELECT "IS_INVALID" FROM "MASTER"."SQL_VIEWS"
    WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='CUSTOMER30s'
```

Retrieval result

```
IS_INVALID
----------
Y
```

If the retrieval result is Y, CUSTOMER30s has been invalidated.

If the retrieval result is a null value, CUSTOMER30s has not been invalidated.

Example 2

Of the viewed tables defined by HADB user ADBUSER01, this example displays the list of invalidated viewed tables.

```
SELECT "TABLE_NAME" FROM "MASTER"."SQL_VIEWS"
    WHERE "TABLE_SCHEMA"='ADBUSER01' AND "IS_INVALID"='Y'
```

Retrieval result

```
TABLE_NAME
-----------
CUSTOMER30s
CUSTOMER40s
CUSTOMER50s
```

In the retrieval result, a list of invalidated viewed tables is displayed.

11.2.8 Releasing a viewed table from invalidation

The way to release a viewed table from invalidation differs depending on the reason for the invalidation. The following explains the various ways.

Note

When you become unable to tell the reason for invalidation of the viewed table:

If you execute the `ALTER VIEW` statement on the viewed table, the `ALTER VIEW` statement returns an error. By referring to the error message that is output at this time, you can identify the reason for invalidation of the viewed table.

(1) When viewed tables are invalidated by using an ALTER TABLE statement to change the table type

In the following cases, all viewed tables that depend on the table whose type was changed are invalidated.

- When a regular multi-chunk table is changed to an archivable multi-chunk table by the `ALTER TABLE` statement
- When an archivable multi-chunk table is changed to a regular multi-chunk table by the `ALTER TABLE` statement

The following describes the procedure for releasing viewed tables from invalidation.

Example:

Because the `CUSTOMER` table was changed from a regular multi-chunk table to an archivable multi-chunk table, all viewed tables that depend on the `CUSTOMER` table have been invalidated. In this example, all viewed tables that depend on `CUSTOMER` table are released from invalidation.

Procedure:

1. Check a list of viewed tables that have been invalidated.

   By retrieving data from a dictionary table, you can acquire a list of viewed tables that have been invalidated. For details about retrieving data from a dictionary table, see (33) Finding a list of dependent viewed tables in B.22 Searching a dictionary table.

   In this example, assume that viewed tables `CUSTOMER30s`, `CUSTOMER40s`, and `CUSTOMER50s` have been invalidated.

2. Use the `ALTER VIEW` statement to rebuild the invalidated viewed tables.

   ```
   ALTER VIEW "CUSTOMER30s" RECREATE
   ```

   Also, use the `ALTER VIEW` statement to rebuild viewed tables `CUSTOMER40s` and `CUSTOMER50s`. You can release the viewed tables from invalidation.

   Tip

   If there are multiple viewed tables to be rebuilt, execute the `ALTER VIEW` statement in ascending order of view level.

(2) When viewed tables are invalidated by using an ALTER TABLE statement to change a column name in a table

If a column name in a table is changed by the `ALTER TABLE` statement, all viewed tables that depend on the table whose column name has been changed are invalidated. The following describes the procedure for releasing viewed tables from invalidation.

Note that the procedure for releasing viewed tables from invalidation differs, depending on whether the column name to be changed is specified in the CREATE VIEW statement.

Example:

Because a column name of the CUSTOMER table was changed, all viewed tables that depend on the CUSTOMER table have been invalidated. In this example, all viewed tables that depend on the CUSTOMER table are released from invalidation.

Procedure (when the column name to be changed has not been specified in the CREATE VIEW statement):

1. Check a list of viewed tables that have been invalidated.

   By retrieving data from a dictionary table, you can acquire a list of viewed tables that have been invalidated. For details about retrieving data from a dictionary table, see (33) Finding a list of dependent viewed tables in B.22 Searching a dictionary table.

   In this example, assume that viewed tables CUSTOMER30s, CUSTOMER40s, and CUSTOMER50s have been invalidated.

2. Use the ALTER VIEW statement to rebuild the invalidated viewed tables.

```
ALTER VIEW "CUSTOMER30s" RECREATE
```

   Also, use the ALTER VIEW statement to rebuild viewed tables CUSTOMER40s and CUSTOMER50s. You can release the viewed tables from invalidation.

> **Tip**
>
> If there are multiple viewed tables to be rebuilt, execute the ALTER VIEW statement in ascending order of view level.

Procedure (when the column name to be changed has been specified in the CREATE VIEW statement):

1. Check the list of invalidated viewed tables and their definition information.

   By retrieving data from a dictionary table, you can obtain a list of invalidated viewed tables and their definition information. For details about retrieving data from a dictionary table, see (32) Finding a list of dependent viewed tables, and the definition information of each of the viewed tables in B.22 Searching a dictionary table.

   In this example, assume that viewed tables CUSTOMER30s, CUSTOMER40s, and CUSTOMER50s have been invalidated.

2. Use the DROP VIEW statement to delete the invalidated viewed tables.

```
DROP VIEW "CUSTOMER30s"
DROP VIEW "CUSTOMER40s"
DROP VIEW "CUSTOMER50s"
```

   Delete all the invalidated viewed tables that you checked in step 1.

3. Modify the contents specified in the CREATE VIEW statement.

   Of the specified contents of the CREATE VIEW statement you checked in step 1, modify the column name that was changed to the new name.

   Example: When the column name of the T1 table is changed from C3 to C33:

   <Before modification>

```
CREATE VIEW "VT1" ("VC1","VC2","VC3")
    AS SELECT "C1"."C2","C3" FROM "T1"
```

   <After modification>

```
CREATE VIEW "VT1" ("VC1","VC2","VC3")
    AS SELECT "C1"."C2","C33" FROM "T1"
```

4. Use the CREATE VIEW statement to redefine the invalidated viewed tables.

```
CREATE VIEW "CUSTOMER30s" ("AGE","SEX","GNO","PNAME","PRICE")
  AS SELECT "CUSTOMER"."AGE",
            "CUSTOMER"."SEX",
            "CUSTOMER"."GNO",
            "STOCK"."PNAME",
            "STOCK"."PRICE"
       FROM "CUSTOMER","STOCK"
         WHERE "CUSTOMER"."AGE"=30 AND "CUSTOMER"."GNO"="STOCK"."GNO"
```

In the same way, for CUSTOMER40s and CUSTOMER50s, also use the CREATE VIEW statement to redefine the viewed tables.

Tip

If there are multiple viewed tables to be redefined, execute the CREATE VIEW statement in ascending order of view level.

(3) When viewed tables are invalidated due to erroneous revocation of the SELECT privilege for an underlying table

If the SELECT privilege for an underlying table is erroneously revoked, all viewed tables that depend on the underlying table are invalidated. The following describes the procedure for releasing viewed tables from invalidation.

Example:

Because the SELECT privilege for the underlying table was erroneously revoked, all viewed tables that depend on the underlying table have been invalidated. In this example, the relevant viewed tables and underlying table are identified, and all viewed tables that depend on the underlying table are released from invalidation.

Procedure:

1. Check a list of viewed tables that have been invalidated.

   By retrieving data from a dictionary table, you can acquire a list of viewed tables that have been invalidated. For details about retrieving data from a dictionary table, see (33) Finding a list of dependent viewed tables in B.22 Searching a dictionary table.

   In this example, assume that viewed table CUSTOMER30s has been invalidated.

2. Check the underlying table for the viewed tables that have been invalidated.

   By retrieving data from a dictionary table based on the invalidated viewed tables identified in step 1, you can acquire the underlying table of the viewed tables that have been invalidated. For details about retrieving data from a dictionary table, see (35) Checking which base tables and viewed tables are underlying tables of a viewed table in B.22 Searching a dictionary table.

   This example assumes that the underlying table of the invalidated viewed table CUSTOMER30s is the CUSTOMER table.

3. Execute the GRANT statement to grant the SELECT privilege for the underlying table.

   For underlying table CUSTOMER identified in step 2, execute the GRANT statement to grant the SELECT privilege.

```
GRANT SELECT ON "CUSTOMER" TO "ADBUSER01"
```

The underlined part indicates the authorization identifier of the HADB user who owns viewed table CUSTOMER30s.

4. Use the ALTER VIEW statement to rebuild the invalidated viewed tables.

```
ALTER VIEW "CUSTOMER30s" RECREATE
```

Execute the ALTER VIEW statement to rebuild viewed table CUSTOMER30s. You can release the viewed tables from invalidation.

> **Tip**
>
> If there are multiple viewed tables to be rebuilt, execute the ALTER VIEW statement in ascending order of view level.

(4) When viewed tables are invalidated due to erroneous deletion of a table

If a table is erroneously deleted#, all viewed tables that depend on the deleted table are invalidated. The following describes the procedure for releasing viewed tables from invalidation.

#

This case includes when the DROP TABLE or DROP VIEW statement for which specification of drop behavior is omitted is executed.

Example:

Because base table CUSTOMER is erroneously deleted, all viewed tables that depend on the deleted table are invalidated. In this example, the base table is redefined, and all viewed tables that depend on the base table are released from invalidation.

Procedure:

1. Check a list of viewed tables that have been invalidated.

   By retrieving data from a dictionary table, you can acquire a list of viewed tables that have been invalidated. For details about retrieving data from a dictionary table, see (33) Finding a list of dependent viewed tables in B.22 Searching a dictionary table.

   In this example, assume that viewed table CUSTOMER30s has been invalidated.

2. Redefine the table that was erroneously deleted.

   - If the table that was erroneously deleted is a base table, redefine the base table by using the CREATE TABLE statement.
   - If the table that was erroneously deleted is a viewed table, redefine the viewed table by using the CREATE VIEW statement.

   In this example, base table CUSTOMER is redefined by using the CREATE TABLE statement.

3. Ask another HADB user to execute the GRANT statement to grant to you the SELECT privilege for the table that was redefined by that user.

   If the table redefined in step 2 is owned by another HADB user, ask that user to execute the GRANT statement to grant to you the SELECT privilege for that table.

   Note that this step is not necessary if you redefined the table by yourself.

```
GRANT SELECT ON "CUSTOMER" TO "ADBUSER01"
```

   The underlined part indicates the authorization identifier of the HADB user who owns viewed table CUSTOMER30s.

4. Use the ALTER VIEW statement to rebuild the invalidated viewed tables.

```
ALTER VIEW "CUSTOMER30s" RECREATE
```

Execute the ALTER VIEW statement to rebuild viewed table CUSTOMER30s. You can release the viewed tables from invalidation.

> **Tip**
>
> If there are multiple viewed tables to be rebuilt, execute the ALTER VIEW statement in ascending order of view level.

(5) When viewed tables are invalidated by using the ALTER VIEW statement to rebuild a viewed table that is the underlying table of those viewed tables

If you execute the ALTER VIEW statement to rebuild a viewed table that is the underlying table of viewed tables, all viewed tables that depend on that rebuilt viewed table are invalidated. The following describes the procedure for releasing viewed tables from invalidation.

Note that the procedure for releasing viewed tables from invalidation differs, depending on whether a column name has been changed in a viewed table (underlying table), or whether the column name to be changed has been specified in the CREATE VIEW statement.

Procedure (when a column name has not been changed in a viewed table (underlying table), or when the column name to be changed has not been specified in the CREATE VIEW statement):

If a column name of the viewed table (underlying table) has not been changed, release the viewed table from invalidation by the following procedure. If a column name of the viewed table (underlying table) has been changed, but the column name to be changed has not been specified in the CREATE VIEW statement, also release the viewed table from invalidation by the following procedure.

1. Check a list of viewed tables that have been invalidated.

   By retrieving data from a dictionary table, you can acquire a list of viewed tables that have been invalidated. For details about retrieving data from a dictionary table, see (33) Finding a list of dependent viewed tables in B.22 Searching a dictionary table.

   In this example, assume that viewed tables CUSTOMER30s, CUSTOMER40s, and CUSTOMER50s have been invalidated.

2. Use the ALTER VIEW statement to rebuild the invalidated viewed tables.

   ```
   ALTER VIEW "CUSTOMER30s" RECREATE
   ```

   In the same way, use the ALTER VIEW statement to rebuild viewed tables CUSTOMER40s and CUSTOMER50s. You can release the viewed tables from invalidation.

   > **Tip**
   >
   > If there are multiple viewed tables to be rebuilt, execute the ALTER VIEW statement in ascending order of view level.

Procedure (when the column name to be changed has been specified in the CREATE VIEW statement):

If a column name of the viewed table (underlying table) has been changed and the column name to be changed has been specified in the CREATE VIEW statement, release the viewed tables from invalidation by the following procedure.

1. Check a list of invalidated viewed tables and their definition information.

   By retrieving data from a dictionary table, you can acquire a list of invalidated viewed tables and their definition information. For details about retrieving data from a dictionary table, see (32) Finding a list of dependent viewed tables, and the definition information of each of the viewed tables in B.22 Searching a dictionary table.

   In this example, assume that viewed tables CUSTOMER30s, CUSTOMER40s, and CUSTOMER50s have been invalidated.

2. Use the DROP VIEW statement to delete the invalidated viewed tables.

```
DROP VIEW "CUSTOMER30s"
DROP VIEW "CUSTOMER40s"
DROP VIEW "CUSTOMER50s"
```

   Delete all the invalidated viewed tables that you checked in step 1.

3. Modify the contents specified in the CREATE VIEW statement.

   Of the specified contents of the CREATE VIEW statement you checked in step 1, modify the column name that was changed.

   Example: When a column name of the T1 table is changed from C3 to C33:

   <Before modification>

```
CREATE VIEW "VT1" ("VC1","VC2","VC3")
    AS SELECT "C1"."C2","C3" FROM "T1"
```

   <After modification>

```
CREATE VIEW "VT1" ("VC1","VC2","VC3")
    AS SELECT "C1"."C2","C33" FROM "T1"
```

4. Use the CREATE VIEW statement to redefine the invalidated viewed tables.

```
CREATE VIEW "CUSTOMER30s" ("AGE","SEX","GNO","PNAME","PRICE")
  AS SELECT "CUSTOMER"."AGE",
            "CUSTOMER"."SEX",
            "CUSTOMER"."GNO",
            "STOCK"."PNAME",
            "STOCK"."PRICE"
       FROM "CUSTOMER","STOCK"
         WHERE "CUSTOMER"."AGE"=30 AND "CUSTOMER"."GNO"="STOCK"."GNO"
```

   In the same way, use the CREATE VIEW statement to redefine viewed tables CUSTOMER40s and CUSTOMER50s.

> **Tip**
>
> If multiple viewed tables are to be redefined, execute the CREATE VIEW statement in ascending order of view level.

(6) When rebuilding of viewed tables at version upgrade fails

When version upgrade of the HADB server is performed, viewed tables might be rebuilt. In that case, if the rebuilding of viewed tables fails, the target viewed tables are invalidated. For details, see (5) Re-creation of viewed tables in the event of a version upgrade in 8.6.5 Notes on version upgrading.

The following describes the procedure for releasing the viewed tables (for which rebuilding failed) from invalidation.

Procedure:

1. Check a list of invalidated viewed tables and their definition information.

By retrieving data from a dictionary table, you can acquire a list of invalidated viewed tables and their definition information. For details about retrieving data from a dictionary table, see (32) Finding a list of dependent viewed tables, and the definition information of each of the viewed tables in B.22 Searching a dictionary table.

2. Use the `DROP VIEW` statement to delete the invalidated viewed tables.

   Delete all the invalidated viewed tables that you checked in step 1.

3. Create the `CREATE VIEW` statement.

   Create the `CREATE VIEW` statement for the invalidated viewed tables that you checked in step 1.

4. Based on the output message, modify the contents specified in the `CREATE VIEW` statement.

   Modify the `CREATE VIEW` statement you created in step 3, and eliminate the reason for invalidation of the viewed tables.

   If rebuilding of viewed tables at version upgrade fails, the `KFAA51312-W` message is output, indicating that rebuilding of viewed tables failed. In the message that is output immediately before the `KFAA51312-W` message, the reason for invalidation of the viewed tables and the steps to take are explained. According to the message, modify the `CREATE VIEW` statement.

5. Use the `CREATE VIEW` statement to redefine the invalidated viewed tables.

   Use the `CREATE VIEW` statement that you modified in step 4 to redefine the invalidated viewed tables.

> **Tip**
>
> If multiple viewed tables are to be redefined, execute the `CREATE VIEW` statement in ascending order of view level.

11.2.9 Deleting invalidated viewed tables

If invalidated viewed tables are unnecessary, delete the viewed tables according to the following procedure.

Example:

Of the viewed tables defined by HADB user `ADBUSER01`, this example deletes the invalidated viewed tables.

Procedure:

1. Acquire a list of viewed tables that have been invalidated.

   By retrieving data from a dictionary table, you can acquire a list of viewed tables that have been invalidated. For an example of retrieving data from a dictionary table, see **Example 2** in 11.2.7 Checking whether a viewed table has been invalidated.

2. Use the `DROP VIEW` statement to delete the invalidated viewed tables.

```
DROP VIEW "CUSTOMER30s"
DROP VIEW "CUSTOMER40s"
DROP VIEW "CUSTOMER50s"
```

   Execute the `DROP VIEW` statement for all viewed tables that have been invalidated.

- **When there are many viewed tables to be deleted**

  If you delete the viewed table whose view level is the smallest, all invalidated viewed tables that depend on the deleted viewed table are deleted in a batch.

  Example:

```
DROP VIEW "CUSTOMER30s" CASCADE
```

If you specify CASCADE for drop behavior as in the preceding example, you can delete all viewed tables that depend on CUSTOMER30s (all viewed tables that have been invalidated).

To check the viewed table whose view level is the smallest among the viewed tables that have been invalidated, execute the following SELECT statement. For the underlined part, specify the schema name.

```
SELECT "TABLE_NAME","VIEW_LEVEL" FROM "MASTER"."SQL_VIEWS"
    WHERE "TABLE_SCHEMA"='ADBUSER01' AND IS_INVALID = 'Y'
    ORDER BY "VIEW_LEVEL"
```

Retrieval result

```
TABLE_NAME  VIEW_LEVEL
----------- ----------
CUSTOMER30s 3
CUSTOMER40s 4
    :       :
```

The viewed table (CUSTOMER30s) displayed on the first line of the retrieval result is the viewed table whose view level is the smallest.

11.2.10 Checking the underlying tables of a viewed table

By retrieving data from a dictionary table, you can check the underlying tables of a viewed table. The following shows an example of retrieving data from a dictionary table.

Example:

This example checks the underlying tables of viewed table CUSTOMER30s defined by HADB user ADBUSER01.

Execute the following SELECT statement to make a retrieval from a dictionary table:

```
SELECT "MVU"."TABLE_SCHEMA","MVU"."TABLE_NAME"
  FROM  "MASTER"."SQL_VIEW_TABLE_USAGE" "MVU"
    WHERE "MVU"."VIEW_SCHEMA"='ADBUSER01'
      AND "MVU"."VIEW_NAME" = 'CUSTOMER30s'
      AND "MVU"."IS_DIRECT" ='Y'
```

Retrieval result

```
TABLE_SCHEMA TABLE_NAME
------------ ----------
ADBUSER01    CUSTOMER
ADBUSER01    STOCK
```

This retrieval result shows that the underlying tables of CUSTOMER30s are the CUSTOMER and STOCK tables owned by ADBUSER01.

11.2.11 Checking dependent viewed tables

By retrieving data from a dictionary table, you can check dependent viewed tables. The following shows an example of retrieving data from a dictionary table.

Example:

This example checks the viewed tables that depend on the CUSTOMER table defined by HADB user ADBUSER01.

Execute the following SELECT statement to make a retrieval from a dictionary table.

```
SELECT "MV"."TABLE_SCHEMA","MV"."TABLE_NAME"
    FROM  "MASTER"."SQL_VIEW_TABLE_USAGE" "MVU"
         ,"MASTER"."SQL_VIEWS" "MV"
      WHERE "MVU"."TABLE_SCHEMA"='ADBUSER01'
        AND "MVU"."TABLE_NAME" = 'CUSTOMER'
        AND "MV"."TABLE_SCHEMA" = "MVU"."VIEW_SCHEMA"
        AND "MV"."TABLE_NAME" = "MVU"."VIEW_NAME"
```

Retrieval result

```
TABLE_SCHEMA TABLE_NAME
------------ ----------
ADBUSER01    CUSTOMER30s
ADBUSER01    CUSTOMER40s
ADBUSER02    CUSTOMER90s
```

In the retrieval result, a list of dependent viewed tables is displayed.

11.2.12 Checking the access privileges for a viewed table

By retrieving data from a dictionary table, you can check the access privileges for a viewed table. For an example of retrieving data from a dictionary table, see (23) When checking the access privileges for a table in B.22 Searching a dictionary table.

11.2.13 Operation example of using viewed tables

This subsection describes how to define viewed tables and how to grant access privileges for when the following operation conditions are satisfied.

■ Operation conditions

- A customer table (CUSTOMERS) that stores customer data is defined so that HADB users are allowed to refer to customer data.
- HADB users who refer to customer data will not be able to refer data stored in the customer name column (CUSTOMER_NAME) in the customer table.
- The table name (CUSTOMERS) of the customer table will not be opened to the HADB users who refer to customer data.

■ Tables defined in this operation example

- Customer table (CUSTOMERS)

  This base table stores the actual customer data.
- Viewed table for the customer table (V1_CUSTOMERS)

  V1_CUSTOMERS is a viewed table that depends on the underlying table CUSTOMERS. The customer name column (CUSTOMER_NAME) is deleted from the customer table, and the resulting table is defined as a viewed table.
- Viewed table that depends on the viewed table (underlying table) of the customer table (V2_CUSTOMERS)

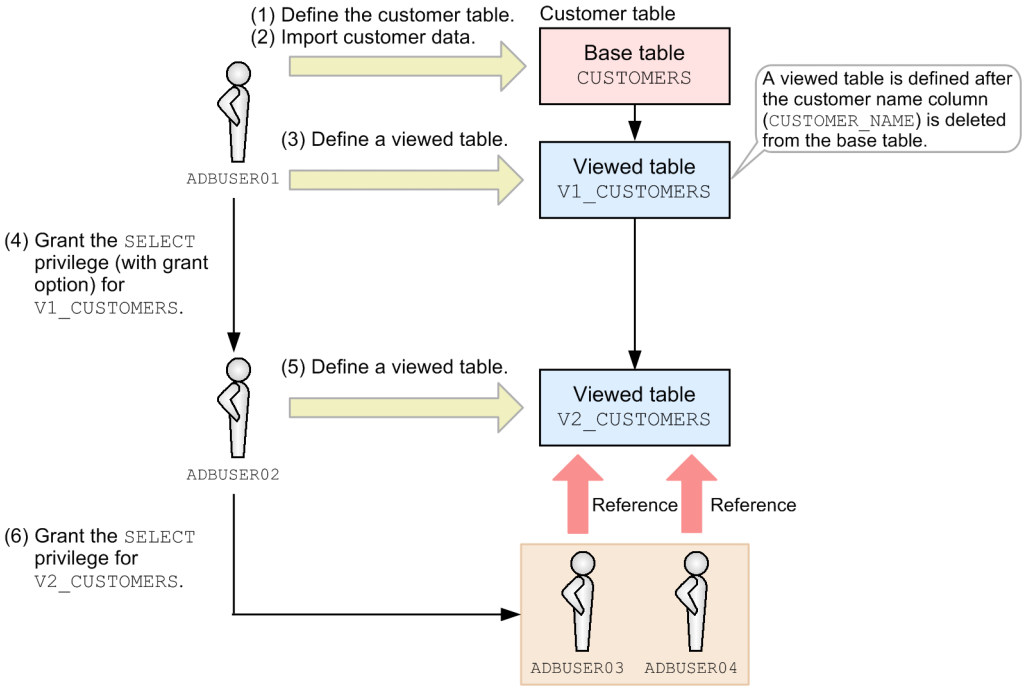
  V2_CUSTOMERS is a viewed table that depends on the underlying table V1_CUSTOMERS.

  HADB users who refer to customer data are allowed to access V2_CUSTOMERS.

■ **HADB users in this operation example**

- HADB user ADBUSER01: Owner of customer table CUSTOMERS and viewed table V1_CUSTOMERS
- HADB user ADBUSER02: Owner of viewed table V2_CUSTOMERS
- HADB user ADBUSER03: HADB user who refers to customer data
- HADB user ADBUSER04: HADB user who refers to customer data

■ **Operation relationship diagram**



Note: The numbers enclosed by parentheses in the diagram correspond to the respective step numbers in the following procedure.

To satisfy the preceding operation conditions, the following procedure defines the customer table and viewed tables, and grants the access privileges for individual tables to HADB users.

Procedure:

1. **ADBUSER01 defines the customer table.**

```
CREATE TABLE "ADBUSER01"."CUSTOMERS"("CUSTOMER_ID" INTEGER,
                                     "CUSTOMER_NAME" CHAR(100),
                                     "REGISTERED_DATE" TIMESTAMP)
             IN "DBAREA01"
```

2. **ADBUSER01 imports customer data into the customer table.**

   Execute the adbimport command to import customer data into the customer table.

3. **ADBUSER01 defines viewed table V1_CUSTOMERS for the customer table.**

```
CREATE VIEW "ADBUSER01"."V1_CUSTOMERS"
    AS SELECT "CUSTOMER_ID","REGISTERED_DATE" FROM "ADBUSER01"."CUSTOMERS"
```

   The CUSTOMER_NAME column is deleted from the customer table, and the resulting table is defined as viewed table V1_CUSTOMERS.

4. **ADBUSER01 grants the SELECT privilege (with grant option) for viewed table V1_CUSTOMERS to ADBUSER02.**

```
GRANT SELECT ON "ADBUSER01"."V1_CUSTOMERS" TO "ADBUSER02" WITH GRANT OPTION
```

The SELECT privilege (with grant option) for viewed table V1_CUSTOMERS is granted to ADBUSER02.

> **Note**
>
> - WITH GRANT OPTION is specified to grant to ADBUSER02 the grant option for the SELECT privilege for viewed table V1_CUSTOMERS.
> - The SELECT privilege for viewed table V1_CUSTOMERS is granted to ADBUSER02 so that, in step 5, ADBUSER02 can define viewed table V2_CUSTOMERS that depends on viewed table V1_CUSTOMERS (as the underlying table).
> - Only the SELECT privilege is granted in this example, but you can also grant other types of access privileges as necessary.

5. **ADBUSER02 defines viewed table V2_CUSTOMERS.**

```
CREATE VIEW "ADBUSER02"."V2_CUSTOMERS"
    AS SELECT * FROM "ADBUSER01"."V1_CUSTOMERS"
```

Viewed table V2_CUSTOMERS that depends on viewed table V1_CUSTOMERS (as the underlying table) is defined.

> **Note**
>
> - In the CREATE VIEW statement, * is specified for SELECT in the query expression. However, viewed table V2_CUSTOMERS is defined with the CUSTOMER_NAME column deleted because that table depends on the underlying table V1_CUSTOMERS, which also has its CUSTOMER_NAME column deleted.
> - By specifying viewed table V1_CUSTOMERS as the underlying table for viewed table V2_CUSTOMERS, the name of the customer table (CUSTOMERS) that stores actual customer data can be hidden from users.

6. **ADBUSER02 grants the SELECT privilege for viewed table V2_CUSTOMERS to ADBUSER03 and ADBUSER04.**

```
GRANT SELECT ON "ADBUSER02"."V2_CUSTOMERS" TO "ADBUSER03","ADBUSER04"
```

ADBUSER03 and ADBUSER04 can now refer to customer data (in viewed table V2_CUSTOMERS) because the SELECT privilege for viewed table V2_CUSTOMERS has been granted.

> **Note**
>
> ADBUSER03 and ADBUSER04 are HADB users who only refer to customer data. Therefore, the grant option for the SELECT privilege for viewed table V2_CUSTOMERS is not granted. Thus, in the preceding GRANT statement, WITH GRANT OPTION is not specified.

If HADB user ADBUSER05 who refers to customer data is added while the preceding operation continues, ADBUSER02 needs to grant the SELECT privilege for viewed table V2_CUSTOMERS to ADBUSER05. Then, ADBUSER05 is allowed to refer to customer data.

Note

- In this example, only ADBUSER01 can update, add, or delete customer data in the customer table.
- ADBUSER02 performs management of the HADB users who refer to customer data (that is, management of access privileges for viewed table V2_CUSTOMERS).
- Users from ADBUSER02 to ADBUSER05 can only refer to customer data. The CUSTOMER_NAME column cannot be referred.

11.3 Index operations

This section explains index operations.

11.3.1 Defining indexes

To define an index (B-tree index, text index, or range index) for a base table, execute the `CREATE INDEX` statement. For details about the `CREATE INDEX` statement, see *CREATE INDEX (define an index)* in *Definition SQL Statements* in the manual *HADB SQL Reference*.

You can define indexes only for base tables that you (the HADB user with the authorization identifier currently connected to the HADB server) own. A user cannot define indexes for a base table that is owned by another HADB user.

Text indexes cannot be defined for column store tables.

See the following sections to define indexes for appropriate columns:

- 5.3 Designing a B-tree index
- 5.4 Designing a text index
- 5.5 Designing a range index

Important

When you define an index for a table for which segments for storing rows are allocated, the index is placed in unfinished status (and no index data is created). Indexes in unfinished status cannot be used to retrieve, add, update, or delete data.

For details about how to release an index from unfinished status, see the following subsections:

- 15.9.1 Steps to take when unfinished status is applied to a B-tree index
- 15.10.1 Steps to take when unfinished status is applied to a text index
- 15.11.1 Steps to take when unfinished status is applied to a range index

For details about the status of allocating segments for storing rows, see 5.3.1 Notes on defining B-tree indexes (unfinished status of B-tree indexes).

11.3.2 Rebuilding B-tree indexes

If you have performed row insertion, row updating, and row deletion operations frequently on a table for which B-tree indexes have been defined, consider rebuilding the B-tree indexes.

You can expect the following benefits from rebuilding B-tree indexes:

- Prevention of decreases in the performance of retrieval processing when the B-tree indexes are used.
- Reduction of the amount of space used by data DB areas (the data for invalid B-tree indexes resulting from row updating and deletion operations can be deleted).
- Suppression of index page splits (unused areas in index pages for B-tree indexes can be reallocated).

To rebuild B-tree indexes, execute the `adbidxrebuild` command. For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

Note

- To rebuild a B-tree index that is in unfinished status, see 15.9.1 Steps to take when unfinished status is applied to a B-tree index.
- If you rebuilt a unique index that violated the uniqueness constraint, see 15.9.2 Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output).

The following subsections provide guidelines for determining whether B-tree indexes need to be rebuilt.

(1) Prevention of decreases in the performance of retrieval processing when B-tree indexes are used

If row insertion, updating, or deletion operations are performed frequently on a table for which B-tree indexes have been defined, the performance of retrieval processing using those B-tree indexes decreases.

If this is the case, check one of the output items listed below. If the value of the output item is large, rebuild the B-tree indexes. By rebuilding B-tree indexes, you can prevent deterioration in the performance of retrieval processing when B-tree indexes are used.

- Value of `Bidx_validation_check_cnt` that is output as the SQL statement statistical information (number of times row data validity was checked during retrievals using the B-tree indexes)

  The SQL statement statistical information is output to both the statistics log files and the SQL trace files.

  To check the SQL statement statistical information that has been output to the statistics log files, execute the `adbstat` command. For details about the `adbstat` command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

  To check the SQL statement statistical information that has been output to the SQL trace files, check the SQL trace information in the SQL trace files. For details about SQL trace information, see (9) SQL statement statistical information in 10.11.2 Information that is output as SQL trace information.

- Value of `Data_bidx_validation_check_cnt` that is output as the data access information in the access path statistical information (number of times row validity was checked during retrievals using the B-tree indexes)

  The access path statistical information is output to the SQL trace files. Check the SQL trace information in the SQL trace files. For details about access path statistical information, see 10.11.3 Examples of output of and output items for access path statistical information.

If you can check both the SQL statement statistical information and the access path statistical information, we recommend that you check the access path statistical information. Checking the access path statistical information enables you to identify the tables whose B-tree indexes need to be rebuilt when you are retrieving data from multiple tables.

(2) Reduction of the amount of space used by data DB areas

If you have performed row updating or deletion operations frequently on a table for which B-tree indexes have been defined, use the following procedure to determine whether the B-tree indexes need to be rebuilt.

Procedure

1. Use the `adbdbstatus` command to check the amount of space used by the B-tree indexes.

Use the `adbdbstatus` command to output index summary information. Then, check the output value for `Used_segments` (number of segments used by indexes). For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

2. Check the sizes of the B-tree indexes that had been estimated based on the valid data in the base table.

   For details about how to determine the sizes of B-tree indexes, see the following subsections in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the table.

   - Single-chunk table
     (c) Determining the variable SGIDX (for a single-chunk table)
   - Multi-chunk table
     (h) Determining the variable SGIDX (for a multi-chunk table)

   Note that the variable *SGIDX* determines the sum of all B-tree indexes stored in the data DB area. In this step, obtain the size of only the target B-tree indexes.

Compare the results of steps 1 and 2. If their difference is large, rebuild the B-tree indexes.

(3) Suppression of index page splits in B-tree indexes

If row insertion or updating operations are performed frequently on a table for which B-tree indexes have been defined, index page splits might be occurring.

To determine whether index page splits have occurred, check `Bidx_page_split_cnt` (number of times index page splits occurred in B-tree indexes) in the SQL statement statistical information. If index page splits have occurred, rebuild the B-tree indexes. Rebuilding B-tree indexes enables you to suppress index page splits because unused areas in the index pages can be reallocated.

The SQL statement statistical information is output to the statistics log files and the SQL trace files.

To check the SQL statement statistical information that has been output to the statistics log files, execute the `adbstat` command. For details about the `adbstat` command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

To check the SQL statement statistical information that has been output to the SQL trace files, check the SQL trace information in the SQL trace files. For details about SQL trace information, see (9) SQL statement statistical information in 10.11.2 Information that is output as SQL trace information.

11.3.3 Rebuilding text indexes

If you have performed row insertion, row updating, and row deletion operations frequently on a table for which text indexes have been defined, consider rebuilding the text indexes.

You can expect the following benefits from rebuilding text indexes:

- Prevention of decreases in the performance of retrieval processing when the text indexes are used.
- Reduction of the amount of space used by data DB areas (the data for invalid text indexes resulting from row updating and deletion operations can be deleted).
- Suppression of index page splits (unused areas in index pages for text indexes can be reallocated).

To rebuild text indexes, execute the `adbidxrebuild` command. For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

> **Note**
>
> To rebuild a text index that is in unfinished status, see 15.10.1 Steps to take when unfinished status is applied to a text index.

The following subsections provide guidelines for determining whether text indexes need to be rebuilt.

(1) Prevention of decreases in the performance of retrieval processing using when text indexes are used

If row insertion, updating, or deletion operations are performed frequently on a table for which text indexes have been defined, the performance of retrieval processing using those text indexes decreases.

If you have updated (by performing row insertion and updating operations) 0.01% or more of the data stored in a table for which text indexes have been defined, consider rebuilding the text indexes.

Similarly, if you have deleted (by performing row deletion operations) 30% or more of the data stored in a table for which text indexes have been defined, consider rebuilding the text indexes.

Rebuilding text indexes enables you to prevent deterioration in the performance of retrieval processing using those text indexes.

(2) Reduction of the amount of space used by data DB areas

If you have performed row updating or deletion operations frequently on a table for which text indexes have been defined, use the following procedure to determine whether the text indexes need to be rebuilt.

Procedure

1. Use the `adbdbstatus` command to check the amount of space used by the text indexes.

   Use the `adbdbstatus` command to output index summary information. Then, check the output value for `Used_segments` (number of segments used by indexes). For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

2. Check the sizes of the text indexes that had been estimated based on the valid data in the base table.

   For details about how to determine the sizes of text indexes, see the following subsections in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the table.

   - Single-chunk table

     (e) Determining the variable SGTIX (for a single-chunk table)
   - Multi-chunk table

     (j) Determining the variable SGTIX (for a multi-chunk table)

   Note that the formula for the variable *SGTIX* obtains the sum of the values for all text indexes stored in the data DB area. In this step, obtain the value only for the target text indexes.

Compare the results of steps 1 and 2. If their difference is large, rebuild the text indexes.

(3) Suppression of index page splits in text indexes

If row insertion or update operations are performed frequently on a table for which text indexes have been defined, index page splits might be occurring.

To determine whether index page splits have occurred, check `Tidx_page_split_cnt` (number of times index page split occurred in text indexes) in the SQL statement statistical information. If index page splits have occurred, rebuild the text indexes. Rebuilding text indexes enables you to suppress index page splits because unused areas in the text pages can be reallocated.

The SQL statement statistical information is output to the statistics log files and the SQL trace files.

To check the SQL statement statistical information that has been output to the statistics log files, execute the `adbstat` command. For details about the `adbstat` command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

To check the SQL statement statistical information that has been output to the SQL trace files, check the SQL trace information in the SQL trace files. For details about SQL trace information, see (9) SQL statement statistical information in 10.11.2 Information that is output as SQL trace information.

11.3.4 Checking a text index (checking the index option that was specified)

To check the index option that was specified when a text index was defined, see (34) Checking index options specified for text indexes in B.22 Searching a dictionary table.

Whether the text index is used for correction search or word-context search is determined by the specified index option.

11.3.5 Rebuilding range indexes

If you have performed row insertion, row updating, and row deletion operations frequently on a table for which range indexes have been defined, consider rebuilding the range indexes.

You can expect the following benefits from rebuilding range indexes:

- Prevention of decreases in the performance of retrieval processing when the range indexes are used.
- Reduction of the amount of space used by data DB areas (the data for invalid range indexes resulting from row updating and deletion operations can be deleted).

To rebuild range indexes, execute the `adbidxrebuild` command. For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

> **Note**
>
> To rebuild a range index that is in unfinished status, see 15.11.1 Steps to take when unfinished status is applied to a range index.

The following subsections provide guidelines for determining whether range indexes need to be rebuilt.

(1) Prevention of decreases in the performance of retrieval processing when range indexes are used

If row insertion, update, or deletion operations are performed frequently on a table for which range indexes have been defined, the performance of retrieval processing using those range indexes decreases.

To check the status of retrieval processing when range indexes are used, see 13.1.2 Re-evaluating the defined range indexes. Check either the SQL statement statistical information or the access path statistical information to determine the usage status of the range indexes. If the range indexes are providing no benefit (the calculation result is 0 or close to 0), rebuild the range indexes.

(2) Reduction of the amount of space used by data DB areas

If you have performed row updating or deletion operations frequently on a table for which range indexes have been defined, use the following procedure to determine whether the range indexes need to be rebuild.

Procedure

1. Use the `adbdbstatus` command to check the amount of space used by the range indexes.

   Use the `adbdbstatus` command to output index summary information. Then, check the output value for `Used_segments` (number of segments used by indexes). For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.
2. Check the sizes of the range indexes that had been estimated based on the valid data in the base table.

   For details about how to determine the sizes of range indexes, see the following subsections in (2) Explanation of variables under 5.8.1 Determining the total number of pages in the data DB area. Note that the subsection to see differs depending on the type of the table.

   - Single-chunk table

     (d) Determining the variable SGRIX (for a single-chunk table)
   - Multi-chunk table

     (i) Determining the variable SGRIX (for a multi-chunk table)

   Note that the formula for the variable *SGRIX* obtains the sum of the values for all range indexes stored in the data DB area. In this step, obtain the value only for the target range indexes.

Compare the results of steps 1 and 2. If their difference is large, rebuild the range indexes.

11.3.6 Checking a range index (whether it can skip chunks)

This subsection explains how to check whether a defined range index can skip chunks.

> **Note**
>
> Range indexes defined in versions earlier than 02-02 cannot skip chunks.

By searching the `IS_CHUNK_SKIP` column of the `SQL_INDEXES` dictionary table, you can determine whether a range index can skip chunks. The following shows a `SELECT` statement specification example:

■ **SELECT statement specification example**

```
SELECT "TABLE_SCHEMA","TABLE_NAME","INDEX_NAME","IS_CHUNK_SKIP"
  FROM "MASTER"."SQL_INDEXES" WHERE "INDEX_TYPE" = 'R'
```

If the IS_CHUNK_SKIP column in the search result is Y, the range index can skip chunks. If the IS_CHUNK_SKIP column is the null value, the range index cannot skip chunks.

To change a range index that cannot skip chunks into one that can skip chunks, see 11.3.7 Redefining a range index (setting up chunk skipping).

For details about dictionary table SQL_INDEXES, see B.5 Content of SQL_INDEXES.

> **Note**
>
> If you know the schema name and index identifier of the range index in question, you can also perform a search based on the explanation in (19) When checking whether the index is a range index that can skip chunks under B.22 Searching a dictionary table.

11.3.7 Redefining a range index (setting up chunk skipping)

To change the defined range index that cannot skip chunks into one that can skip chunks, redefine the range index, and then execute the adbidxrebuild command. The following shows the procedure.

1. Delete the range index.
   Use the DROP INDEX statement to delete the range index that cannot skip chunks. See 11.3.9 Deleting an index.
2. Redefine a range index.
   Use the CREATE INDEX statement to redefine a range index for the table for which the range index deleted in step 1 had been defined.
   For details about the CREATE INDEX statement, see *CREATE INDEX (define an index)* in *Definition SQL* in the manual *HADB SQL Reference*.
3. Rebuild a range index.
   Use the adbidxrebuild command to rebuild a range index for the table for which the range index was redefined in step 2.
   For details about the adbidxrebuild command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

After the range index is redefined, executing the adbidxrebuild command creates a range index that can skip chunks.

11.3.8 Changing the data DB area that stores indexes

To change the data DB area that stores indexes to another data DB area, you must first delete the indexes, and then redefine the indexes. After that, you must rebuild the indexes.

Note

For details about how to change the data DB area that stores a base table to another data DB area, see 11.1.14 Changing the data DB area that stores a base table.

By using the procedure shown later, change the data DB area that stores indexes to another data DB area.

You might also want to add a new data DB area. For details, see 11.10.1 Adding data DB areas. After seeing the subsection and taking action, change the data DB area that stores indexes.

Procedure

1. Check the definition information of the indexes.
   Check the specification content of the `CREATE INDEX` statement that was used to define the indexes for the base table. You will use the specification content when you redefine indexes for the base table.
   If you are not sure of the specification content of the `CREATE INDEX` statement that was used to define the indexes, see (29) Finding out index definition information in B.22 Searching a dictionary table. By retrieving data from a dictionary table, you can check the specification content of the `CREATE INDEX` statement that was used to define the indexes.
2. Delete the indexes.
   Use the `DROP INDEX` statement to delete the indexes.
3. Redefine indexes.
   Use the `CREATE INDEX` statement to redefine indexes for the base table. Note the following points when you create a `CREATE INDEX` statement:
   - Create a `CREATE INDEX` statement with the specification content of the `CREATE INDEX` statement that was used to define the indexes (the specification content you checked in step 1).
   - Specify the DB area name of another data DB area that will store the indexes for `IN` *DB-area-name* in the `CREATE TABLE` statement.
4. Rebuild indexes.
   Use the `adbidxrebuild` command to rebuild the indexes you defined in step 3.

The procedure for changing a data DB area that stores indexes to another data DB area is complete.

Note

- For details about the `adbidxrebuild` command, see the manual *HADB Command Reference*.
- For details about the `DROP INDEX` and `CREATE INDEX` statements, see the manual *HADB SQL Reference*.

11.3.9 Deleting an index

To delete an index (B-tree index, text index, or range index) defined for a table, execute the `DROP INDEX` statement.

For details about the `DROP INDEX` statement, see *DROP INDEX (delete an index)* in *Definition SQL Statements* in the manual *HADB SQL Reference*.

A user (the HADB user with the authorization identifier that was used for the current connection to the HADB server) can delete only those indexes owned by that user. A user cannot delete indexes owned by other HADB users.

Also, you cannot delete the following indexes by using the `DROP INDEX` statement. If you want to delete such an index, use the `DROP TABLE` statement to delete the entire base table.

- B-tree indexes automatically defined for the columns that comprise the primary key
- Range indexes automatically defined for the archive range columns in an archivable multi-chunk table

Important

Before you delete indexes, check whether you need to re-execute the command for the base table for which the target indexes (B-tree indexes or text indexes) are defined (whether the base table is non-updatable). For details about the check method, see (1) Checking whether a base table is non-updatable in 10.9.2 Checking the status and usage of a base table.

If re-execution of the command is necessary

Re-execute the command. Then, delete the indexes.

If you deleted indexes when command re-execution was necessary (when the base table was non-updatable)

Temporary work files created by the interrupted command might remain. To delete the temporary work files, you must release all base tables from the non-updatable status. For details about deleting temporary work files, see (2) When there are unneeded temporary work files on the disk in 15.2.5 Steps to take in the event of a shortage of disk space for storing temporary work files during command execution.

11.4 Performing operations on multi-chunk tables

This section explains operations specific to multi-chunk tables. A multi-chunk table is a general name for the following tables:

- Regular multi-chunk table
- Archivable multi-chunk table

For details about base table operations, see 11.1 Base table operations.

11.4.1 Defining a multi-chunk table

To define a multi-chunk table, execute the `CREATE TABLE` statement in which chunk specification is specified.

Note that the contents of chunk specification differs between regular multi-chunk tables and archivable multi-chunk tables.

When defining a regular multi-chunk table

Of *chunk-specification*, specify `CHUNK=`*maximum-number-of-chunks* only.

When defining an archivable multi-chunk table

Specify the following:

- `CHUNK=`*maximum-number-of-chunks*
- *chunk-archive-specification*

For an example definition of multi-chunk table, see *Example* in *Specification format and rules for the CREATE TABLE statement* in the manual *HADB SQL Reference*.

Before you define an archivable multi-chunk table, you need to create an archive directory. Note the following when you operate on archive directories.

▪ Notes on archive directories

- Do not delete archive directories that are being used. If you delete an archive directory being used, database compatibility is lost, and operations of the HADB server cannot be guaranteed.
- Do not update or delete any files stored under an archive directory. If you update or delete such files, operations of the HADB server cannot be guaranteed.
- Do not create directories or store files under an archive directory. If you do so, relevant directories and files might be deleted when an operation on an archivable multi-chunk table is performed.

11.4.2 Storing data in a multi-chunk table (background import)

This subsection explains how to store data in a multi-chunk table by using background import.

Note

In addition to using background import, you can also use a method that stores data in a multi-chunk table in addition mode. For details about data import in addition mode, see 11.1.7 Storing data in a base table (data import).

To perform background import, execute the `adbimport` command with the `-b` option specified. The following shows an example of performing background import by specifying the `-b` option for the `adbimport` command.

For the specification format of the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

■ Background import execution example

The HADB user (`ADBUSER01`) executes background import on the shop table (`SHOPSLIST`).

```
adbimport -u ADBUSER01 -p '#HelloHADB_01' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file.txt
          -b -m '2014/01/01-2014/01/31'
          SHOPSLIST
          /home/adbmanager/imp_file/imp_data_path.txt
```

Note

■ Adding a comment to a chunk

When you import data into a multi-chunk table, you can add a comment to the chunk that will store the data. You use the `-m` option to add a comment.

When the operation targets a specific chunk, adding a comment to that chunk will make it easier to identify the target chunk.

Important

When you perform background import, for the `adb_import_rthd_num` import option, specify a value that satisfies the following formula. Specifying a value that satisfies the following formula for the `adb_import_rthd_num` import option will improve the data storage efficiency.

Formula

$$value\text{-}specified\text{-}for\text{-}import\text{-}option\text{-}adb\_import\_rthd\_num \leq \uparrow \mathrm{MIN}\left(C \times db\_org\ ,\ db\_org \times index\_num \times 0.25 \right) \uparrow + 1$$

Explanation of the variables

C

If the background import target table is a row store table, substitute 2. If the target table is a column store table, substitute 0.5.

db_org

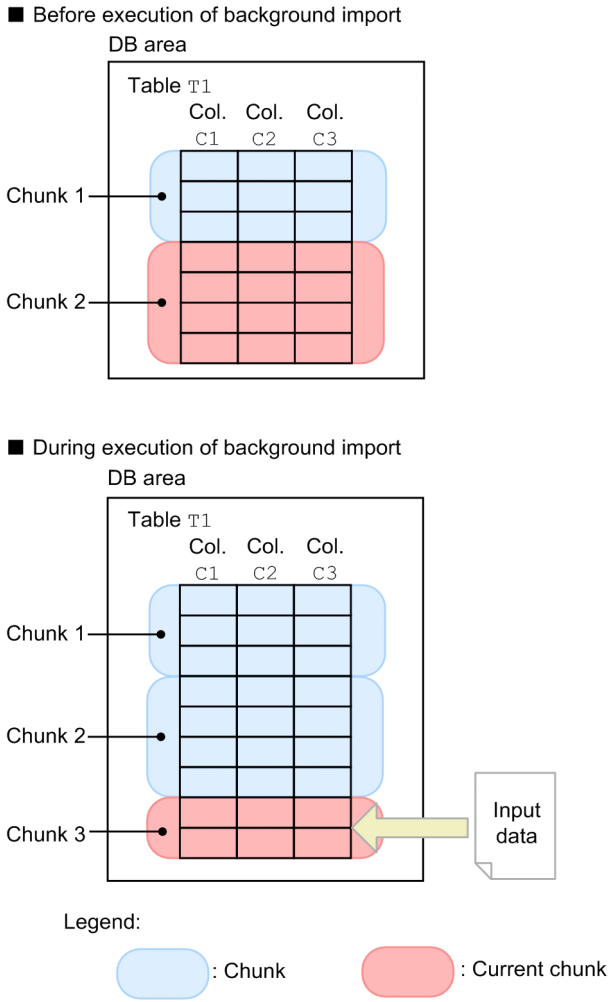
File size of the input data file to be used for background import (gigabytes)

index_num

Number of B-tree indexes and text indexes that are defined for the background import target table

When background import executes, HADB creates a new chunk and stores the import data in that chunk. The following figure shows where data is stored by background import.

Figure 11-3: Data storage destination by background import



Explanation

When the `adbimport` command is executed with the background-import facility, data is stored in newly created chunk 3. During this processing, the current chunk is changed from chunk 2 to chunk 3.

While data is being stored in chunk 3, you can retrieve the data stored in chunk 1 and chunk 2.

Important

Each time background import is executed, a new chunk is created. Consequently, repeating background import increases the number of chunks. When the number of chunks increases, performance might decline during retrieval processing using B-tree indexes and text indexes. Therefore, consider reducing the number of chunks by merging them. To merge chunks, see 11.4.9 Merging chunks (to reduce the number of chunks).

11.4.3 Notes on retrieving and updating data in an archivable multi-chunk table

This subsection provides notes on retrieving and updating data in an archivable multi-chunk table.

(1) Retrieving rows with the SELECT statement

When you perform retrieval processing on an archivable multi-chunk table, the HADB server first narrows down the chunks to be accessed, based on the range in the archive range column for each chunk and the search conditions in the `SELECT` statement. When you retrieve data in an archived chunk, you cannot use an index for retrieval.

Therefore, to narrow down the chunks to be accessed during retrieval from an archivable multi-chunk table, in the `SELECT` statement, specify a search condition that uses the archive range column. For details, see *Considerations when searching an archivable multi-chunk table* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

If the number of retrievals of data in an archived chunk temporarily grows, consider that you use the `adbunarchivechunk` command to release the archived state of the chunk.

(2) Inserting rows with the INSERT statement

No special considerations need be taken when executing the `INSERT` statement to insert a row into an archivable multi-chunk table. The inserted row data is stored in the current chunk.

(3) Updating rows with the UPDATE statement

You can use the `UPDATE` statement to update data in an unarchived chunk, but there are special rules for specifying the search conditions in the `UPDATE` statement. For details, see *Rules* in *Specification format and rules for the UPDATE statement* in the manual *HADB SQL Reference*.

Note that you cannot use the `UPDATE` statement to update data in an archived chunk. If you want to use the `UPDATE` statement to update data in an archived chunk, execute the `adbunarchivechunk` command to release the archived state of the chunk beforehand.

(4) Deleting rows with the DELETE statement

You can use the `DELETE` statement to delete data in an unarchived chunk, but there are special rules for specifying the search conditions in the `DELETE` statement. For details, see *Rules* in *Specification format and rules for the DELETE statement* in the manual *HADB SQL Reference*.

Note that you cannot use the `DELETE` statement to delete data in an archived chunk. If you want to use the `DELETE` statement to delete data in an archived chunk, execute the `adbunarchivechunk` command to release the archived state of the chunk beforehand.

11.4.4 Temporarily excluding data to be imported to a multi-chunk table from retrieval (creating a chunk in wait status)

If you want to keep data stored using background import from being retrieved, and then make it retrievable at a specific time, specify the `-b` and `--status wait` options for the `adbimport` command.

A chunk in wait status is created and data is stored in it. During this process, the current chunk is not changed. Furthermore, the chunk creation date and time in the system table STATUS_CHUNKS and the date and time when the current chunk was swapped are not set.

Important

- A chunk in wait status cannot be manipulated using a data manipulation SQL statement. You cannot execute the SELECT or UPDATE statement on data that is in a chunk in wait status. However, you can use the PURGE CHUNK or TRUNCATE TABLE statement to delete a chunk in wait status. If you want to manipulate a chunk in wait status by using a data manipulation SQL statement, change the status of the chunk to normal status.
- If you want to change a chunk in wait status into a chunk in normal status (a chunk that can be manipulated by a data manipulation SQL statement), execute the adbchgchunkstatus command. For details, see 11.4.12 Changing the chunk status.
- At any time during operation, a chunk created in wait status by using the adbimport command can be made subject to (or not subject to) retrieval, by using the adbchgchunkstatus command with that chunk. For details, see 11.4.24 Operation taking background import and chunks into consideration (Example 2: Using chunks in wait status).

Note

For details about the system table STATUS_CHUNKS, see C.5 Content of STATUS_CHUNKS.

An execution example is described below in which background import is performed by specifying the -b and --status wait options for the adbimport command.

For details about the specification format of the adbimport command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

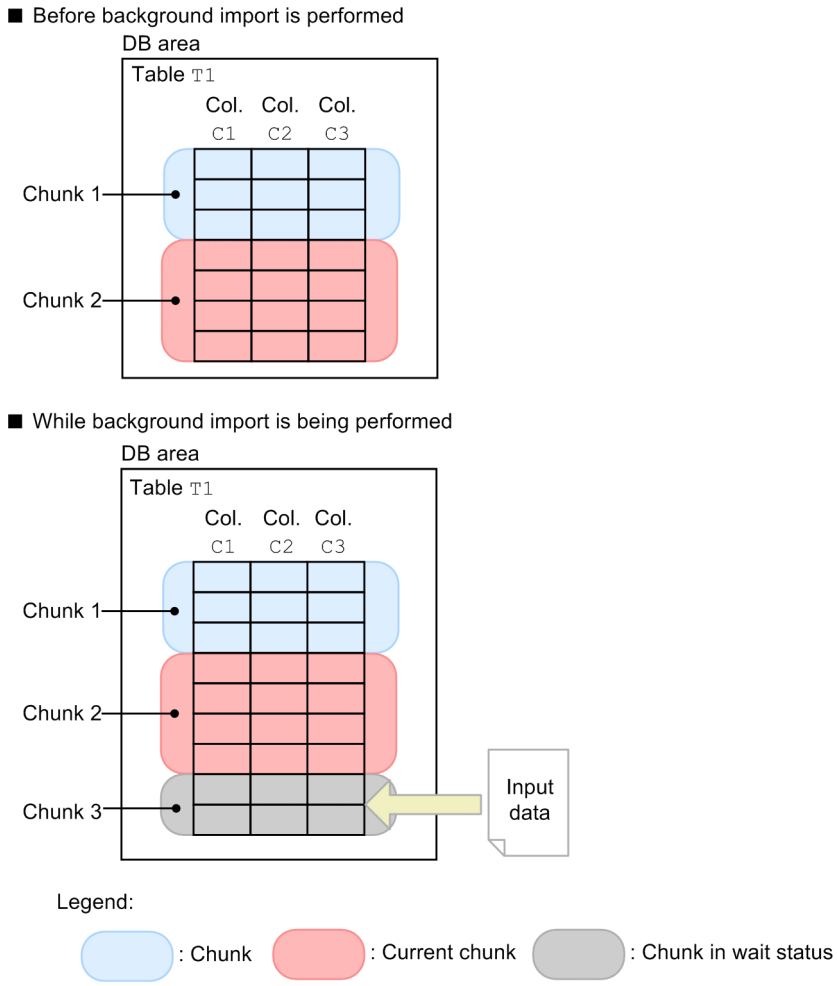
■ Execution example using background import to create a chunk in wait status

An HADB user (ADBUSER01) uses background import to store non-retrievable data in a shop table (SHOPSLIST).

```
adbimport -u ADBUSER01 -p '#HelloHADB_01' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file.txt
          -b --status wait
          -m '2014/06/01-2014/06/30'
          SHOPSLIST
          /home/adbmanager/imp_file/imp_data_path.txt
```

When background import is performed to create a chunk in wait status, HADB creates a new chunk in wait status and stores data in that chunk. The following figure shows the use of background import to create a chunk in wait status.

Figure 11-4: Use of background import to create a chunk in wait status



Explanation

When a chunk in wait status is created by executing the `adbimport` command with the background-import facility applied, data is stored in the newly created chunk in wait status (chunk 3). During this process, chunk 2 remains as the current chunk.

While data is being stored in chunk 3, you can retrieve the data stored in chunk 1 and chunk 2. Note that since chunk 3 is a chunk in wait status, it cannot be manipulated using a data manipulation SQL statement.

Important

Every time background import is performed with the `-b` and `--status wait` options specified, a new chunk in wait status is created. Consequently, if you repeat background importing, the number of chunks in wait status increases as a result. When the number of chunks in wait status increases, execution of the `adbidxrebuild` command might take longer. This is because the index rebuilding process targets both chunks in normal status and chunks in wait status. If there are a large number of chunks in wait status, consider reducing their number by merging chunks. To merge chunks in wait status, see 11.4.9 Merging chunks (to reduce the number of chunks).

11.4.5 Exporting data in units of chunks

You can output data from a multi-chunk table to a file in units of chunks by using either of the following two methods. Both methods require you to specify the chunk ID of each chunk to be output.

- **Execute the `adbexport` command by specifying a chunk ID for the `-c` option**

  To output data from a multi-chunk table to a file in units of chunks, execute the `adbexport` command by specifying a chunk ID for the `-c` option. Note that the `adbexport` command can be executed by the HADB server, but not by an HADB client.

  For details about the `adbexport` command, see *adbexport (Export Data)* in the manual *HADB Command Reference*.

- **Execute the `#GETDATA` subcommand of the `adbsql` command**

  To output data from a multi-chunk table to a file in units of chunks, execute the `#GETDATA` subcommand of the `adbsql` command. When you execute `#GETDATA` by specifying the chunk ID of the output target chunk, you can output the execution result to a file by using redirection. Note that the `adbsql` command can be executed by the HADB server and by an HADB client (Linux version).

  For details about the `adbsql` command, see *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

> **Note**
>
> We recommend that, of the two preceding methods, you use the method that uses the `adbexport` command. The `adbexport` command can output data in a shorter time than the `adbsql` command. Use the `adbsql` command only when an HADB client (Linux version) needs to output data from a multi-chunk table to a file.

The following shows the types of chunks from which you can output data by using the preceding two methods.

Types of chunk data that can be output

- Chunk in normal status
- Chunk in wait status

With the preceding two methods, you can output data from not only chunks in normal status, but also chunks in wait status. Note that you cannot output data from chunks in wait status in the case shown in 11.1.8 Outputting data from a base table to a file (data export).

The following describes the procedures for outputting data in units of chunks by using the `adbexport` command and by using the `adbsql` command.

> **Important**
>
> If a chunk is being merged with other chunks by using the `adbmergechunk` command, we recommend that you do not output data from that chunk. Output data from the chunk to a file after the `adbmergechunk` command has completed the chunk merge processing. If you output data from a chunk that is being merged with other chunks by using the `adbmergechunk` command, the data to be output to the file differs depending on the timing. For details, see (4) Chunks that are retrieved during execution of the adbmergechunk command in 11.4.9 Merging chunks (to reduce the number of chunks).

■ Procedure for outputting data by executing the `adbexport` command with the `-c` option

1. Determine the chunk ID of the chunk to be output.

   Check the ID of the chunk you intend to specify in the -c option of the adbexport command. The chunk ID information is stored in the STATUS_CHUNKS system table.

   For details about how to determine the chunk ID, see C.9 Searching system tables.

2. Execute the adbexport command.

   Based on the determined chunk ID, execute the adbexport command with the -c option specified to output the data that is stored in the chunk you want to output.

   To output data to a CSV file compressed in GZIP format, also specify the --compress GZIP option.

   For details about the adbexport command, see *adbexport (Export Data)* in the manual *HADB Command Reference*.

■ Procedure for outputting data by executing the #GETDATA subcommand of the adbsql command

1. Determine the chunk ID of the chunk to be output.

   Check the chunk ID to be specified for CHUNKID in the #GETDATA subcommand of the adbsql command. The chunk ID information is stored in the STATUS_CHUNKS system table.

   For details about how to determine the chunk ID, see C.9 Searching system tables.

2. Executing the adbsql command

   Based on the determined chunk ID, execute the #GETDATA subcommand of the adbsql command to output the data that is stored in the chunk you want to output.

   For details about the #GETDATA subcommand of the adbsql command, see *adbsql subcommands* in *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

11.4.6 Deleting data in units of chunks

To delete data in units of chunks, execute the PURGE CHUNK statement. The following describes how to use the PURGE CHUNK statement to delete data:

Deletion procedure

1. Determine the chunk ID of the chunk to be deleted.

   To execute the PURGE CHUNK statement to delete the data in a chunk, determine the chunk ID of the chunk to be deleted. Chunk ID information is stored in the STATUS_CHUNKS system table.

   For details about how to determine the chunk ID, see C.9 Searching system tables.

2. Execute the PURGE CHUNK statement.

   Execute the PURGE CHUNK statement for the determined chunk ID to delete the desired chunk.

   Note that you cannot delete the current chunk.

   For details about the PURGE CHUNK statement, see *PURGE CHUNK (delete all rows in a chunk)* in the manual *HADB Command Reference*.

You can specify a condition (INpredicate, quantified predicate) for the WHERE clause of the PURGE CHUNK statement to delete in a batch the chunks that satisfy the condition. For an example of specifying the PURGE CHUNK statement, see *Example* in *Specification format and rules for the PURGE CHUNK statement* of *PURGE CHUNK (delete all rows in a chunk)* in the manual *HADB Command Reference*.

Note

- If you delete a chunk by using the PURGE CHUNK statement, you can reuse the storage area used by the chunk.
- If you delete an archived chunk, the relevant archive files are also deleted.

11.4.7 Deleting data stored in a multi-chunk table in a batch

To delete data stored in a multi-chunk table in a batch, execute the TRUNCATE TABLE statement.

For details about how to delete all rows in a base table by using the TRUNCATE TABLE statement, see 11.1.5 Deleting all rows from a base table.

Note

- If you execute the TRUNCATE TABLE statement, all chunks in a multi-chunk table are deleted. An empty chunk is then created.
- All chunks for indexes that are defined in a multi-chunk table are also deleted.
- If there is an archived chunk, the corresponding archive files are also deleted.

11.4.8 Checking the chunk status and the number of chunks created

This subsection describes how to check the status of chunks created in a multi-chunk table and the number of chunks created.

You can check information of chunks (the chunk status and the number of chunks) in the following ways:

- Executing the adbdbstatus command
- Retrieving data in dictionary table SQL_TABLES
- Retrieving data in system table STATUS_CHUNKS
- Retrieving data in dictionary table SQL_COLUMNS
- Retrieving data in dictionary table SQL_INDEXES

The following subsections describe the methods of checking chunk information.

(1) Executing the adbdbstatus command

You can use the adbdbstatus command to check the following chunk information:

Chunk information that can be checked in the DB area summary information

Execute the adbdbstatus command with the -c dbarea option specified. In the execution result of the command, you can check the following information regarding a multi-chunk table:

- Creatable_chunks (Number of chunks that can be created in a DB area)
- Created_chunks (Number of chunks that have been created in a DB area)

Chunk information that can be checked in the table summary information

Execute the `adbdbstatus` command with the `-c table` option specified. In the execution result of the command, you can check the following information regarding a multi-chunk table:

- `Creatable_chunks` (Number of chunks that can be created in a base table)
- `Created_chunks` (Number of chunks that have been created in a base table)
- `Current_chunk_ID` (Current chunk ID of a base table)
- `Storage_format` (Table-data storage format)

For an archivable multi-chunk table, you can also check the following information from the execution result of the command:

- `Archive_chunks` (Number of archived chunks)

Chunk information that can be checked in the archived chunk summary information

Execute the `adbdbstatus` command with the `-c archivechunk` option specified. In the execution result of the command, you can check the following information regarding an archivable multi-chunk table:

- `Chunk_ID` (Chunk ID)
- `Chunk_status` (Chunk status)
- `Archive_status` (Whether the chunk is archived)
- `Range_min` (Minimum value for the archive range column)
- `Range_max` (Maximum value for the archive range column)
- `Rows` (Number of rows stored in an archived chunk)
- `Archive_file_num` (Total number of archive files corresponding to an archived chunk)
- `Archive_file_size` (Total size of all archive files corresponding to an archived chunk)
- `Unarchive_table_size` (Size of segments that were used to store a table in the chunk before being archived)
- `Unarchive_index_size` (Size of segments that were used to store an index in the chunk before being archived)
- `Compression_ratio` (Compression ratio of an archived chunk)
- `Chunk_comment` (Comment set for a chunk)

Chunk information that can be checked in the information about the usage of DB areas, tables, and indexes

Execute the `adbdbstatus` command with the `-d used` option specified. In the execution result of the command, you can check the following information regarding a multi-chunk table:

- `Chunk_ID` (Chunk ID)
- `Chunk_create_time` (Creation date and time of a chunk)
- `Chunk_swap_time` (Date and time the current chunk was switched)
- `Chunk_status` (Chunk status)

If you use `Chunk_ID` as a key, you can also check the following information:

- `Used_segments` (Number of segments being used for each chunk)
- `Used_pages` (Number of pages being used for each chunk)
- `Storage_format` (Chunk-data storage format)

Chunk information that can be checked in the archived chunk usage information

Execute the `adbdbstatus` command with the `-d used` and `-c archivechunk` options specified. In the execution result of the command, you can check the following information regarding an archivable multi-chunk table:

- `Chunk_status` (Chunk status)
- `Range_min` (Minimum value for the archive range column)
- `Range_max` (Maximum value for the archive range column)
- `Archive_file_name` (Name of an archive file)
- `Archive_file_size` (Size of an archive file)

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

(2) Searching the SQL_TABLES dictionary table

When you use the `SELECT` statement to search the `SQL_TABLES` table, you can check the following chunk information:

- `IS_CHUNK` (Whether the table is a multi-chunk table)
- `N_CHUNK_RESERVED` (Maximum number of chunks)

For an archivable multi-chunk table, you can also check the following information:

- `IS_ARCHIVABLE` (Whether the table is an archivable multi-chunk table)
- `ARCHIVE_DIRECTORY_PATH` (Absolute path to the archive directory)

For details about the `SQL_TABLES` table, see B.2 Content of SQL_TABLES.

For details about how to retrieve data from the `SQL_TABLES` table, see B.22 Searching a dictionary table.

(3) Searching the STATUS_CHUNKS system table

When you use the `SELECT` statement to search the `STATUS_CHUNKS` table, you can check the following chunk information:

- `CHUNK_ID` (Chunk ID)
- `CREATE_TIME` (Creation date and time of a chunk)
- `SWAP_TIME` (Date and time the current chunk was switched)
- `CHUNK_COMMENT` (Comment set for a chunk)
- `CHUNK_STATUS` (Chunk status)
- `STORAGE_FORMAT` (Chunk-data storage format)

For an archivable multi-chunk table, you can also check the following information:

- `ARCHIVE_STATUS` (Whether the chunk is archived)

For details about the `STATUS_CHUNKS` table, see C.5 Content of STATUS_CHUNKS.

For details about how to retrieve data from the `STATUS_CHUNKS` table, see C.9 Searching system tables.

(4) Searching the SQL_COLUMNS dictionary table

For an archivable multi-chunk table, if you use the SELECT statement to search the SQL_COLUMNS table, you can check the following chunk information:

- IS_ARCHIVE_RANGE_COLUMN (Whether the column is the archive range column)

For details about the SQL_COLUMNS table, see B.3 Content of SQL_COLUMNS.

For details about how to retrieve data from the SQL_COLUMNS table, see B.22 Searching a dictionary table.

(5) Searching the SQL_INDEXES dictionary table

For an archivable multi-chunk table, if you use the SELECT statement to search the SQL_INDEXES table, you can check the following chunk information:

- IS_ARCHIVE_RANGE (Whether the index is the range index that was automatically defined for the archive range column)

For details about the SQL_INDEXES table, see B.5 Content of SQL_INDEXES.

For details about how to retrieve data from the SQL_INDEXES table, see B.22 Searching a dictionary table.

11.4.9 Merging chunks (to reduce the number of chunks)

If the number of chunks increases due to repeated background import, performance might decline during retrieval processing using B-tree indexes and text indexes. When indexes are split into chunks, the total amount of index data increases and the number of I/O operations increases proportionately to the number of chunks.

Therefore, as the number of chunks increases, consider using the adbmergechunk command to reduce the number of chunks by merging them.

(1) Notes on chunk statuses

You can merge the following chunks. (You can merge multiple chunks of the same status.)

For a regular multi-chunk table:

- You can merge multiple chunks that are in normal status into a single chunk that is in normal status.
- You can merge multiple chunks that are in wait status into a single chunk that is in wait status.

For an archivable multi-chunk table:

- You can merge multiple chunks that are in normal status and not in archived state into a single chunk that is in normal status and not in archived state.
- You can merge multiple archived chunks that are in normal status into a single archived chunk that is in normal status.
- You can merge multiple chunks that are in wait status and not in archived state into a single chunk that is in wait status and not in archived state.
- You can merge multiple archived chunks that are in wait status into a single archived chunk that is in wait status.

Important

- You cannot merge a chunk that is in normal status with a chunk that is in wait status.
- You cannot merge an unarchived chunk with an archived chunk.

(2) Notes on merging chunks in a column store table

If all of the following conditions are met, reorganize the merge-source chunks before merging them:

- The target table for the `adbmergechunk` command is a column store table.
- Update SQL statements have been executed for the data of the merge-source chunks.

If you merge chunks and then reorganize the merge-target chunk, the following problems will arise:

- A larger size of file or data DB area is required for reorganization.
- A longer time is required for reorganization.

You can check whether update SQL statements have been executed for the merge-source chunks from *information about the need for reorganization* that can be obtained by running the `adbdbstatus` command. In the obtained information, if the value of either of the following items is 1 or larger, update SQL statements have been executed for the target chunks:

- `Base_row_pages` (number of basic row pages included in the row-data segments)
- `Invalid_row_information_pages` (number of invalid row information pages included in the row-data segments)

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

(3) Using the adbmergechunk command to merge chunks

The example below uses the `adbmergechunk` command to merge chunks.

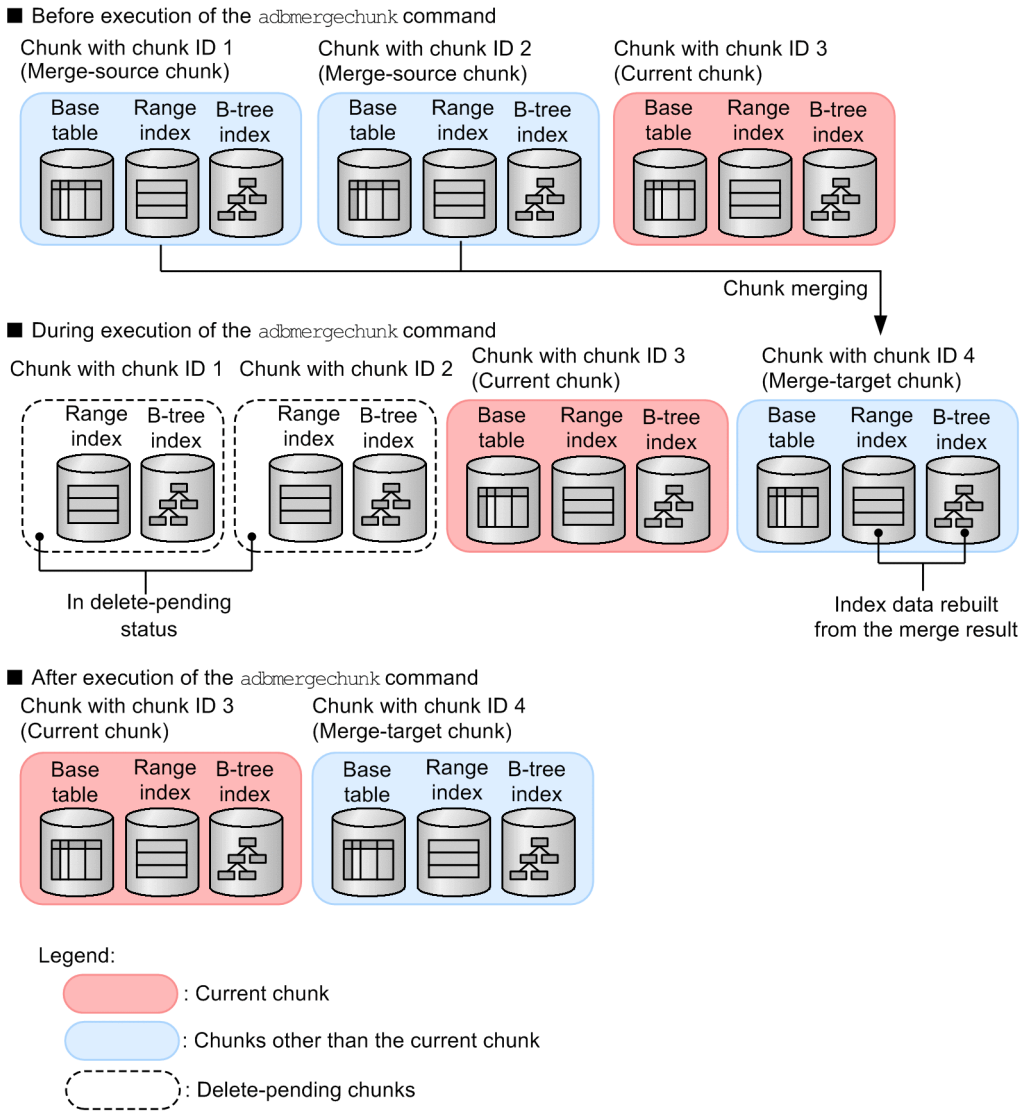
For details about the `adbmergechunk` command, see *adbmergechunk (Merge Chunks)* in the manual *HADB Command Reference*.

■ Chunk merging execution example

The HADB user (`ADBUSER01`) merges two chunks (with chunk IDs `1` and `2`) in the shop table (`SHOPSLIST`).

```
adbmergechunk -u ADBUSER01 -p '#HelloHADB_01' -g 2
              -w /home/adbmanager/tmp
              -z /home/adbmanager/merge_file/merge_opt_file.txt
              -m 'January 2014'
              -c 1,2
              SHOPSLIST
```

Figure 11-5: Merge chunk processing example



Explanation

Executing the `adbmergechunk` command merges two merge-source chunks (with chunk IDs of `1` and `2`) into a single chunk (with chunk ID `4`). When merging is completed, the merge-source chunks are deleted.

Note

■ Delete-pending chunks

If the `adbmergechunk` command is interrupted after the merge-target chunk is ready but before the merge-source chunks are completely deleted, the merge-source chunks are placed in delete-pending status. At this time, the merge-source chunks will remain undeleted. For example, merge-source chunks might remain undeleted if the HADB server has insufficient memory or when the `adbcancel` command is executed.

If the `adbmergechunk` command is executed with `NOWAIT` specified in the `--purge-chunk` option, the merge-source chunks will be in delete-pending status from the time when the merge-target chunk is ready until deletion of the merge-source chunks begins. This remains true even when there is still processing that is referencing the merge-chunk target tables. This means that the merge-source chunks will remain undeleted.

You need to manually delete merge-source chunks that are in delete-pending status.

To manually delete merge-source chunks, determine their chunk IDs based on the explanation in (12) Checking for delete-pending chunks under C.9 Searching system tables, and use the PURGE CHUNK statement to delete those chunks.

You can specify a condition (IN predicate, quantified predicate) for the WHERE clause of the PURGE CHUNK statement to delete in a batch the deletion-pending chunks. For an example of specifying the PURGE CHUNK statement, see *Example* in *Specification format and rules for the PURGE CHUNK statement* of *PURGE CHUNK (delete all rows in a chunk)* in the manual *HADB Command Reference*.

■ Temporary increase in the amount of index data that is associated with chunk merging

While the adbmergechunk command is being executed, index data of both the merge-source chunks and merge-target chunks might exist temporarily. Consequently, the amount of index data might increase temporarily.

During this process, if the adbmergechunk command is interrupted such that there are chunks that remain in delete-pending status, the indexes that were temporarily created also remain. Therefore, you need to manually delete the chunks that remain in delete-pending status.

■ Relationship between merge-source chunk deletion and search/update processing

When the merge-target chunk is completed, the adbmergechunk command checks whether or not there is a search executed on any of the merge-chunk target tables (SQL statements, adbexport command, adbdbstatus command with the --shared-lock option specified, and others). If a search is being performed, the merge-source chunks are deleted after waiting for the completion of all detected searches. For this reason, depending on the search, it might take time for the deletion of merge-source chunks to start. Any searches executed after the merge-target chunk is completed do not affect the deletion of the merge-source chunks.

If, after the completion of the merge-target chunk, a merge-chunk target table needs to be updated without waiting for the completion of merge-source chunk deletion, cancel the adbmergechunk command by using the adbcancel command. After canceling, you can update a table that is to be merge-chunked.

However, the status of the merge-source chunks is delete-pending status; the merge-source chunks are still not deleted. You will have to manually delete merge-source chunks with this status.

■ System chunk

A data DB area contains system chunks that are separate from normal chunks. System chunks make it possible to execute the adbmergechunk command even when no chunks can be created in a base table or data DB area.

When the adbmergechunk command is executed but no chunks can be created in a base table or data DB area, a system chunk is used as the merge-target chunk. Then, when the merge-source chunks are deleted, one of the deleted chunks becomes a system chunk.

(4) Chunks that are retrieved during execution of the adbmergechunk command

The following table describes the chunks that can be retrieved by chunk-ID-based retrieval (the adbexport command with the -c option or the #GETDATA subcommand of the adbsql command) performed during execution of the adbmergechunk command.

Table 11-3: Chunks that are retrieved during execution of the adbmergechunk command

No.	adbmergechunk command processing status	Chunks to be retrieved	Can it be retrieved?
1	After command start but before completion of the merge-target chunk After the KFAA90000-I message (start of the adbmergechunk command processing) is output but before the KFAA51242-I message (creation of the merge-target chunk) is output	Merge-source chunks	Y
		Merge-target chunk	N
		Chunks other than those that are to be merged	Y
2	After completion of the merge-target chunk but before deletion of the merge-source chunks After the KFAA51242-I message (creation of the merge-target chunk) but before the KFAA80243-I message (deletion of the merge-source chunk) is output	Merge-source chunks (in delete-pending status)	N
		Merge-target chunk	Y
		Chunks other than those that are to be merged	Y
3	From complete deletion of the merge-source chunks to command termination After the KFAA80243-I message (deletion of the merge-source chunk) but before the KFAA90001-I message (end of the adbmergechunk command processing) is output	Merge-source chunks	N
		Merge-target chunk	Y
		Chunks other than those that are to be merged	Y

Legend:

Y: Can be retrieved.

N: Because these chunks cannot be retrieved, the retrieval result is 0.

(5) Notes on merging chunks in wait status

When multiple chunks in wait status (merge-source chunks) are merged, the merge-target chunk becomes a chunk in wait status. During this process, the current chunk is not changed.

Furthermore, when chunks in wait status are merged, depending on the combination of the statuses of the chunks to be merged, the information that is set for the chunk creation date and time in the system table STATUS_CHUNKS differs from the information that is set for the date and time when the current chunk was swapped. Details are provided below.

Note

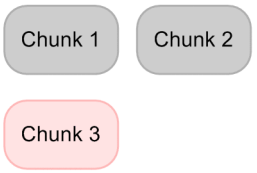
For details about the system table STATUS_CHUNKS, see C.5 Content of STATUS_CHUNKS.

(a) When only chunks in wait status that were never in normal status in the past are merged

When only chunks in wait status that were never in normal status in the past are merged, the chunk creation date and time and the date and time when the current chunk was swapped are not set in the merge-target chunk.

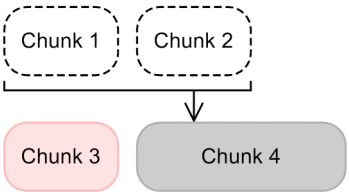
Figure 11-6: Example of merging only chunks in wait status that were never in normal status in the past

■ Before chunks in wait status are merged



Chunk ID	Creation date and time	Swapping date and time
Chunk 1		
Chunk 2		
Chunk 3	2014/09/03 01:00:00	

■ After chunks in wait status are merged



Chunk ID	Creation date and time	Swapping date and time
Chunk 3	2014/09/03 01:00:00	
Chunk 4		

Legend:

: Chunk in wait status that was never in normal status in the past

: Chunk in normal status (current chunk)

: Merge-source chunk (deleted)

Creation date and time: Date and time when a chunk was created

Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

When chunks in wait status (chunks 1 and 2) that were never in normal status in the past are merged, a chunk in wait status (chunk 4) is created as the result of the merge.

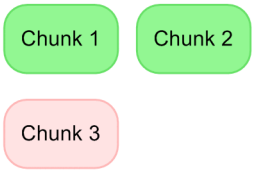
During this process, the current chunk is not changed. Further, the chunk creation date and time and the date and time when the current chunk was swapped are not set in chunk 4.

(b) When only chunks in wait status that were in normal status in the past are merged

When only chunks in wait status that were in normal status in the past are merged, the chunk creation date and time and the date and time when the current chunk was swapped are set in the merge-target chunk.

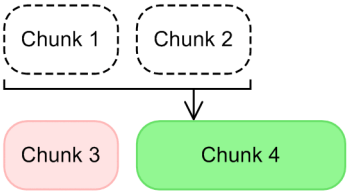
Figure 11-7: Example of merging only chunks in wait status that were in normal status in the past

■ Before chunks in wait status are merged



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	

■ After chunks in wait status are merged



Chunk ID	Creation date and time	Swapping date and time
Chunk 3	2014/09/03 01:00:00	
Chunk 4	2014/09/01 01:00:00	2014/09/03 01:00:00

Legend:

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Merge-source chunk (deleted)

Creation date and time: Date and time when a chunk was created

Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

When chunks in wait status (chunks 1 and 2) that were in normal status in the past are merged, a chunk in wait status (chunk 4) is created as the result of the merge.

During this process, the current chunk is not changed. Further, the chunk creation date and time and the date and time when the current chunk was swapped are set in chunk 4.

- Information set for the chunk creation date and time

  The oldest date and time information associated with the merge-source chunk is set.

- Information set for the date and time when the current chunk was swapped

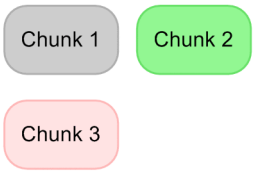
  The most recent date and time information associated with the merge-source chunk is set.

(c) When a chunk in wait status that was never in normal status in the past is merged with a chunk in wait status that was in normal status in the past

When a chunk in wait status that was never in normal status in the past is merged with a chunk in wait status that was in normal status in the past, the chunk creation date and time and the date and time when the current chunk was swapped are set in the merge-target chunk.

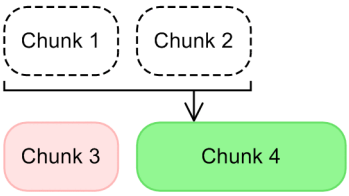
Figure 11-8: Example of merging a chunk in wait status that was never in normal status in the past with a chunk in wait status that was in normal status in the past

■ Before chunks in wait status are merged



Chunk ID	Creation date and time	Swapping date and time
Chunk 1		
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	

■ After chunks in wait status are merged



Chunk ID	Creation date and time	Swapping date and time
Chunk 3	2014/09/03 01:00:00	
Chunk 4	2014/09/02 01:00:00	2014/09/03 01:00:00

Legend:

: Chunk in wait status that was never in normal status in the past

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Merge-source chunk (deleted)

Creation date and time: Date and time when a chunk was created

Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

When a chunk in wait status that was never in normal status in the past (chunk 1) is merged with a chunk in wait status that was in normal status in the past (chunk 2), a chunk in wait status (chunk 4) is created as the merge-target chunk.

During this process, the current chunk is not changed. Furthermore, the chunk creation date and time and the date and time when the current chunk was swapped are set in chunk 4.

- Information set for the chunk creation date and time

  The oldest date and time information associated with the merge-source chunk is set.

- Information set for the date and time when the current chunk was swapped

  The most recent date and time information associated with the merge-source chunk is set.

Important

When a chunk in wait status that was never in normal status in the past is merged with a chunk in wait status that was in normal status in the past, changing the merge-target chunk (chunk in wait status) into a chunk in normal status will not make it the current chunk.

Therefore, we do not recommend merging a chunk in wait status that was never in normal status in the past with a chunk in wait status that was in normal status in the past.

(6) Points to consider when specifying the --purge-chunk option of the adbmergechunk command

This subsection explains the points to consider when executing the adbmergechunk command with the --purge-chunk option specified.

For the --purge-chunk option, you can specify WAIT or NOWAIT. Under normal circumstances, specify WAIT (or omit the --purge-chunk option altogether).

If you execute the adbmergechunk command with WAIT specified for the --purge-chunk option, merge chunk processing takes place in the following sequence. This behavior is the same as if the --purge-chunk option were omitted.

■ Merge chunk processing when WAIT is specified for --purge-chunk option

1. Multiple chunks (merge-source chunks) are merged into a single chunk (the merge-target chunk).
2. After the merge is completed, if there are no more SQL statements or commands whose processing references the table processed by the merge chunk command, the merge-source chunks are deleted.

   If there are still SQL statements or commands whose processing references the table processed by the merge chunk command, the system will wait to delete the merge-source chunks. The merge-source chunks will be deleted as soon as there is no more processing that references the table processed by the merge chunk command.
3. When deletion of the merge-source chunks has completed, the adbmergechunk command terminates.

If you specify WAIT for the --purge-chunk option, as described earlier, the system will sometimes wait for other processes before deleting the merge-source chunks. The adbmergechunk command does not terminate during this waiting period. Therefore, processing that updates the table processed by the merge chunk command cannot be executed. For example, you will be unable to perform background import using the adbimport command.

In some circumstances, having the adbmergechunk command wait to delete merge-source chunks when processing remains that references the table processed by the merge chunk command might be problematic. In these circumstances, consider specifying NOWAIT in the --purge-chunk option. If you execute the adbmergechunk command with NOWAIT specified for the --purge-chunk option, merge chunk processing takes place in the following sequence.

■ Merge chunk processing when NOWAIT is specified for --purge-chunk option

1. Multiple chunks (merge-source chunks) are merged into a single chunk (the merge-target chunk).

   After the merge is completed, if there are no more SQL statements or commands whose processing references the table subject to merge chunk processing, the process advances to step 2.

   If there are still SQL statements or commands whose processing references the table subject to merge chunk processing, the process advances to step 3.
2. If there are no more SQL statements or commands whose processing references the table processed by the merge chunk command, the merge-source chunks are deleted.

   When deletion of the merge-source chunks has completed, the adbmergechunk command terminates.
3. If there are SQL statements or commands whose processing references the table processed by the merge chunk command, the adbmergechunk command terminates without deleting the merge-source chunks.

   The merge-source chunks that are not deleted remain as deletion-pending chunks.

If you specify NOWAIT for the --purge-chunk option, as explained earlier, the adbmergechunk command will sometimes terminate without deleting the merge-source chunks. Because the adbmergechunk command terminates, processing that updates the table processed by the merge chunk command can now be executed. For example, you will be able to perform tasks such as background import using the adbimport command. In this manner, specify NOWAIT

for the --purge-chunk option lets you avoid situations in which the adbmergechunk command waits to delete merge-source chunks while processing remains that references the table being processed.

However, because the merge-source chunks that are not deleted remain as deletion-pending chunks, you then need to delete them by using the PURGE CHUNK statement. You will be unable to execute the adbmergechunk command until the merge-source chunks are deleted. The number of chunks you can create is also reduced by a number equivalent to the number of remaining merge-source chunks. After the adbmergechunk command terminates, use the PURGE CHUNK statement at an appropriate time to delete the merge-source chunks. To delete merge-source chunks, first determine their chunk IDs based on the explanation in (12) Checking for delete-pending chunks under C.9 Searching system tables. Then, use the PURGE CHUNK statement to delete the desired chunks based on the chunk IDs you determined. You can also obtain the chunk ID of the merge-source chunk from the KFAA96785-E message.

Important

Do not specify NOWAIT in the --purge-chunk option if there will be no opportunity for a PURGE CHUNK statement to delete the remaining merge-source chunks before the next execution of the adbmergechunk command. You cannot execute a PURGE CHUNK statement if processing is pending that references or updates a table processed by the merge chunk command for which merge-source chunks remain.

11.4.10 Changing the maximum number of chunks

To change the value for the maximum number of chunks that was specified when a multi-chunk table was defined (value specified for CHUNK), execute the ALTER TABLE statement. The following shows the procedure for changing the value.

Procedure:

1. Check the value for the maximum number of chunks that was specified when a multi-chunk table was defined.

   Check the value specified as the maximum number of chunks when defining the multi-chunk table based on the explanation in (13) When identifying the maximum number of chunks created in all multi-chunk tables under B.22 Searching a dictionary table.

2. Execute the ALTER TABLE statement to change the value for the maximum number of chunks.

   The following shows a specification example of the ALTER TABLE statement.

   Specification example

   This example changes the value for the maximum number of chunks for the shop table (SHOPSLIST) to 500.

```
ALTER TABLE "SHOPSLIST"
      CHANGE OPTION CHUNK=500
```

Important

- Determine the value to be specified for CHANGE OPTION CHUNK of the ALTER TABLE statement based on the explanation in (4) Determining the maximum number of chunks under 5.2.4 Points to consider in defining a multi-chunk table.
- Check whether the value specified for CHANGE OPTION CHUNK of the ALTER TABLE statement does not exceed the *maximum number of chunks that can be managed in one data DB area*, described in 5.6.2 Points to consider in storing a multi-chunk table in the data DB area.

11.4.11 Changing the comment for a chunk

To change the comment for a chunk (including newly setting and deleting a chunk), execute the `adbchgchunkcomment` command. The following shows the procedure for changing the comment.

Procedure:

1. Determine the chunk ID of the chunk whose comment you want to change.

   Determine the chunk ID of the chunk whose comment you want to change by following the instructions in (11) Searching for chunks that included a specified comment under C.9 Searching system tables.

2. Change the comment using the `adbchgchunkcomment` command.

   Execute the `adbchgchunkcomment` command to change the comment set for a chunk, based on the chunk ID determined in step 1. The following shows a specification example of the `adbchgchunkcomment` command.

   Specification example

   An HADB user (`ADBUSER01`) changes the comment for a chunk (whose chunk ID is `5`) in a shop table (`SHOPSLIST`) to `'2014/09/01-2014/09/30'`.

```
adbchgchunkcomment -u ADBUSER01 -p '#HelloHADB_01'
                   -m '2014/09/01-2014/09/30'
                   -c 5 SHOPSLIST
```

This changes the comment that was set for the chunk.

> **Note**
>
> For details about the `adbchgchunkcomment` command, see *adbchgchunkcomment (Set, Change, and Delete Comments for Chunks)* in the manual *HADB Command Reference*.

11.4.12 Changing the chunk status

To change the chunk status, execute the `adbchgchunkstatus` command. For details about the `adbchgchunkstatus` command, see *adbchgchunkstatus (Change Chunk Status)* in the manual *HADB Command Reference*. For details about chunk statuses, see (4) Changing the chunk status in 2.14.2 Managing data in data-import units (chunks).

By executing the `adbchgchunkstatus` command, you can make the following types of chunk status changes:

- Changing a chunk in wait status to a chunk in normal status
  Specify the `-n` option for the `adbchgchunkstatus` command.
- Changing a chunk in normal status to a chunk in wait status
  Specify the `-w` option for the `adbchgchunkstatus` command.

> **Important**
>
> - You can specify the `-n` and `-w` options in the `adbchgchunkstatus` command at the same time.
> - To place the chunk status in archived state, execute the `adbarchivechunk` command. To release a chunk from archived state, execute the `adbunarchivechunk` command.

For details, see 11.4.17 Archiving chunks (when using an archivable multi-chunk table) and 11.4.18 Unarchiving chunks (when using an archivable multi-chunk table).

Note

- For details about how to create a chunk in wait status by background import by using the `adbimport` command, see 11.4.4 Temporarily excluding data to be imported to a multi-chunk table from retrieval (creating a chunk in wait status).
- At any time during operation, a chunk created in wait status by using the `adbimport` command can be made subject to (or not subject to) retrieval, by using the `adbchgchunkstatus` command with that chunk. For details, see (4) Operational example (replacing the data in a chunk).

(1) Using the adbchgchunkstatus command to change the status of a chunk

The following is the procedure for using the `adbchgchunkstatus` command to change the status of a chunk.

■ Procedure for changing the status of a chunk

1. Determine the chunk ID of the chunk whose status you want to change.

   Determine the chunk ID of the chunk whose status you want to change by following the instructions in (13) Checking for chunks in normal status or (14) Checking for chunks in wait status under C.9 Searching system tables.

2. Use the `adbchgchunkstatus` command to change the status of the chunk.

   Based on the chunk ID that you determined in step 1, use the `adbchgchunkstatus` command to change the status of the chunk. Specification examples of the `adbchgchunkstatus` command follow.

   Specification example 1 (Changing a chunk in wait status to a chunk in normal status)

   An HADB user (`ADBUSER01`) changes a chunk in wait status (whose chunk ID is `1`) in a shop table (`SHOPSLIST`) to a chunk in normal status.

```
adbchgchunkstatus -u ADBUSER01 -p '#HelloHADB_01'
                  -n 1 SHOPSLIST
```

   Important

   If there are a large number of chunks in wait status, reduce the number of chunks by merging them before changing their status. If the number of chunks in normal status becomes large, retrieval performance might decline. To merge chunks, see 11.4.9 Merging chunks (to reduce the number of chunks).

   Specification example 2 (Changing a chunk in normal status to a chunk in wait status)

   An HADB user (`ADBUSER01`) changes a chunk in normal status (whose chunk ID is `2`) in a shop table (`SHOPSLIST`) to a chunk in wait status.

```
adbchgchunkstatus -u ADBUSER01  -p '#HelloHADB_01'
                  -w 2 SHOPSLIST
```

This changes the chunk status.

Note that changing the status of a chunk might result in swapping of the current chunk. For details, see (3) Swapping of the current chunk as a result of a chunk status change. As a result of changing the status of a chunk, if swapping of the current chunk occurs, you can make the chunk that was previously the current chunk the current chunk again. For details about the procedure, see (4) Operational example (replacing the data in a chunk).

(2) Relationship of chunk statuses with SQL statements and commands that can be executed

Depending on the status of a chunk, the SQL statements and commands that can be executed on the data contained in the chunk vary. The following two tables show the relationship of chunk statuses with SQL statements and commands that can be executed.

- For chunks that are not in archived state:

  See Table 11-4: Relationship of chunk statuses with SQL statements and commands that can be executed (for chunks that are not in archived state).
- For archived chunks:

  See Table 11-5: Relationship of chunk statuses with SQL statements and commands that can be executed (for archived chunks).

Table 11-4: Relationship of chunk statuses with SQL statements and commands that can be executed (for chunks that are not in archived state)

No.	SQL statement or command to be executed	Chunk status		
		Normal status	Wait status	Delete-pending status
1	`SELECT` statement	Y	N	N
2	`INSERT` statement	Y	N	N
3	`UPDATE` statement	Y	N	N
4	`DELETE` statement	Y	N	N
5	`PURGE CHUNK` statement	Y	Y	Y
6	`TRUNCATE TABLE` statement	Y	Y	Y
7	`adbstart` command (for pre-reading of range indexes)	Y	Y	N
8	`adbsql` command	Y	R[#1]	N
9	`adbmergechunk` command	R[#2]	R[#3]	N
10	`adbexport` command	Y	R[#4]	N
11	`adbidxrebuild` command	Y	Y[#5]	N
12	`adbdbstatus` command (for referencing chunk information)	Y	Y	Y
13	`adbchgchunkcomment` command	Y	Y	Y
14	`adbchgchunkstatus` command	Y	Y	N
15	`adbarchivechunk` command	Y	R[#6]	N
16	`adbunarchivechunk` command	N	N	N

Legend:

Y: Can be executed.

R: Can be executed, but with some restrictions.

N: Cannot be executed.

#1

Only the subcommands `#GETDATA` and `#GETCOUNT` of the `adbsql` command can be executed.

#2

You can merge multiple chunks that are in normal status and not in archived state, with a chunk that is in normal status and not in archived state. You cannot merge other types of chunks.

#3

You can merge multiple chunks that are in wait status and not in archived state with a chunk that is in wait status and not in archived state. You cannot merge other types of chunks.

#4

Only retrieval using the `adbexport` command with the `-c` option specified can be executed.

#5

Indexes contained in a chunk in wait status are also rebuilt.

#6

Chunks in wait status that were never in normal status cannot be placed into archived state.

Table 11-5: Relationship of chunk statuses with SQL statements and commands that can be executed (for archived chunks)

No.	SQL statement or command to be executed	Chunk status		
		Normal status	Wait status	Delete-pending status
1	`SELECT` statement	Y	N	N
2	`INSERT` statement	N	N	N
3	`UPDATE` statement	N	N	N
4	`DELETE` statement	N	N	N
5	`PURGE CHUNK` statement	Y	Y	Y
6	`TRUNCATE TABLE` statement	Y	Y	Y
7	`adbstart` command (for pre-reading of range indexes)	N[#1]	N[#1]	N[#1]
8	`adbsql` command	Y	R[#2]	N
9	`adbmergechunk` command	R[#3]	R[#4]	N
10	`adbexport` command	Y	R[#5]	N
11	`adbidxrebuild` command	N[#6]	N[#6]	N[#6]
12	`adbdbstatus` command (for referencing chunk information)	Y	Y	Y
13	`adbchgchunkcomment` command	Y	Y	Y
14	`adbchgchunkstatus` command	Y	Y	N

No.	SQL statement or command to be executed	Chunk status		
		Normal status	Wait status	Delete-pending status
15	adbarchivechunk command	N	N	N
16	adbunarchivechunk command	Y	Y	N

Legend:

Y: Can be executed.

R: Can be executed, but with some restrictions.

N: Cannot be executed.

#1

No index exists in data in an archived chunk. Therefore, you cannot use the adbstart command to perform pre-reading of range indexes.

#2

Only the subcommands #GETDATA and #GETCOUNT of the adbsql command can be executed.

#3

You can merge multiple archived chunks that are in normal status with an archived chunk that is in normal status. You cannot merge other types of chunks.

#4

You can merge multiple archived chunks that are in wait status with an archived chunk that is in wait status. You cannot merge other types of chunks.

#5

Only retrieval using the adbexport command with the -c option specified can be executed.

#6

No index exists in data in an archived chunk. Therefore, you cannot use the adbidxrebuild command to perform index rebuild.

(3) Swapping of the current chunk as a result of a chunk status change

When the status of a chunk is changed by the adbchgchunkstatus command, the current chunk might be swapped, depending on the type of chunk whose status is being changed and the content of the status-changing process. Furthermore, the information that is set for the chunk creation date and time in the system table STATUS_CHUNKS differs from the information that is set for the date and time when the current chunk was swapped.

Note

For details about the system table STATUS_CHUNKS, see C.5 Content of STATUS_CHUNKS.

The following table shows the details.

Table 11-6: Execution conditions for changing the chunk status

Type of chunk whose status is being changed and the content of the status-changing process		Change to a chunk in normal status			
		None	Chunk in wait status that was in normal status in the past#1	Chunk in wait status that was never in normal status in the past#2	Chunk in wait status that was in normal status in the past#1, and chunk in wait status that was never in normal status in the past#2
Change to a chunk in wait status	None	--	(a)	(b)	(c)
	Chunk in normal status that is not the current chunk	(d)	(e)	(f)	(g)
	Chunk in normal status that is the current chunk	(h)	(i)	(j)	(k)
	Current chunk and a chunk in normal status that is not the current chunk	(l)	(m)	(n)	(o)

Legend:

--: Not applicable.

#1

This is a chunk in wait status for which the chunk creation date and time and the date and time when the current chunk was swapped are set.

#2

This is a chunk in wait status for which the chunk creation date and time and the date and time when the current chunk was swapped are not set.

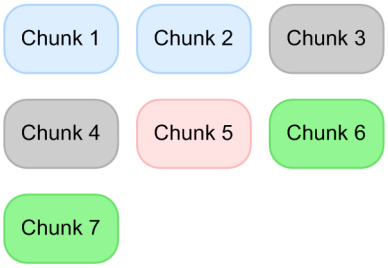
■ Detailed explanation of (*a*) through (*o*)

For details about (*a*) to (*o*) in the preceding table, see (a) adbchgchunkstatus command execution example 1 to (o) adbchgchunkstatus command execution example 15.

In each execution example, the `adbchgchunkstatus` command is executed on the chunks shown in the following figure.

Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	
Chunk 6		
Chunk 7		

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

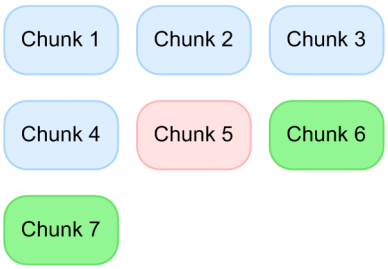
Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

(a) adbchgchunkstatus command execution example 1

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*a*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-10: Execution of the adbchgchunkstatus command under condition (a)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	
Chunk 6		
Chunk 7		

Legend:

: Chunk in normal status

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in wait status that were in normal status in the past (chunks 3 and 4) are changed to chunks in normal status.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*a*)

Current chunk

Not swapped.

Chunk creation date and time

Not set.

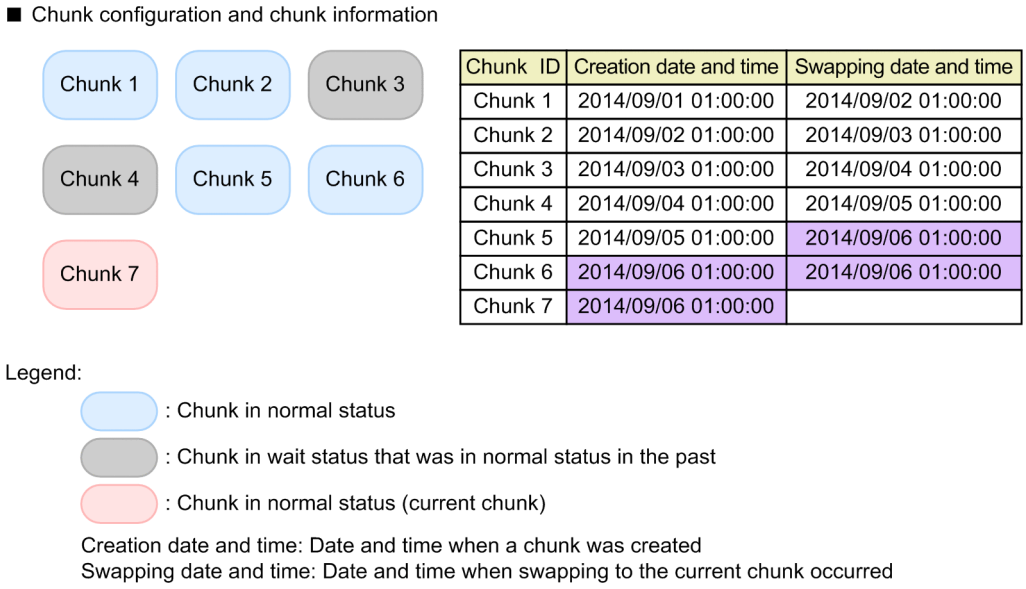
Date and time when the current chunk was swapped

Not set.

(b) adbchgchunkstatus command execution example 2

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*b*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-11: Execution of the adbchgchunkstatus command under condition (b)



Explanation

Chunks in wait status that were never in normal status (chunks 6 and 7) are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Furthermore, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*b*)

Current chunk

Swapped to one of the chunks in wait status that was never in normal status, and which has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- The chunk in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

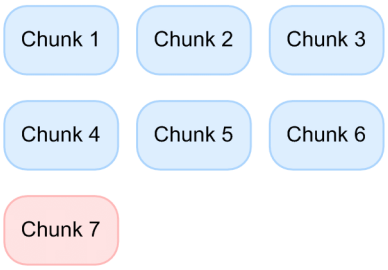
- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and which did not become the current chunk
- The chunk that was the current chunk before the swap

(c) adbchgchunkstatus command execution example 3

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*c*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-12: Execution of the adbchgchunkstatus command under condition (c)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6	2014/09/06 01:00:00	2014/09/06 01:00:00
Chunk 7	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in normal status (current chunk)

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in wait status that were in normal status in the past (chunks 3 and 4) and chunks in wait status that were never in normal status (chunks 6 and 7) are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Further, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*c*)

Current chunk

This is swapped to one of the chunks in wait status that was never in normal status, and that has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that was never in normal status in the past

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

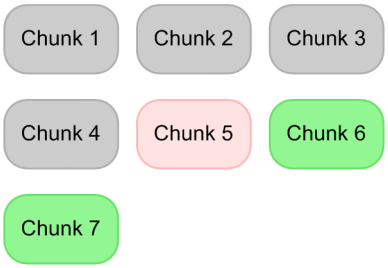
- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, that did not become the current chunk, and that was never in normal status in the past
- The chunk that was the current chunk before the swap

(d) adbchgchunkstatus command execution example 4

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*d*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-13: Execution of the adbchgchunkstatus command under condition (d)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	
Chunk 6		
Chunk 7		

Legend:

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) are changed to chunks in wait status.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*d*)

Current chunk

Not swapped.

Chunk creation date and time

Not set.

Date and time when the current chunk was swapped

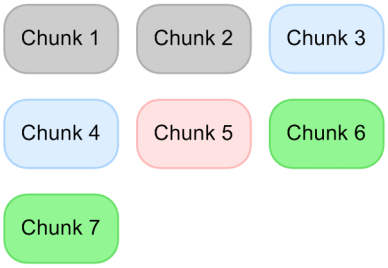
Not set.

(e) adbchgchunkstatus command execution example 5

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*e*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-14: Execution of the adbchgchunkstatus command under condition (e)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	
Chunk 6		
Chunk 7		

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) are changed to chunks in wait status. Additionally, chunks in wait status that were in normal status in the past (chunks 3 and 4) are changed to chunks in normal status.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*e*)

Current chunk

Not swapped.

Chunk creation date and time

Not set.

Date and time when the current chunk was swapped

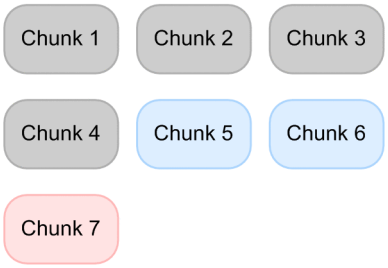
Not set.

(f) adbchgchunkstatus command execution example 6

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*f*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-15: Execution of the adbchgchunkstatus command under condition (f)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6	2014/09/06 01:00:00	2014/09/06 01:00:00
Chunk 7	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) are changed to chunks in wait status. Additionally, chunks in wait status that were never in normal status (chunks 6 and 7) are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Further, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*f*)

Current chunk

Swapped to one of the chunks in wait status that was never in normal status, and that has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that was never in normal status in the past

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

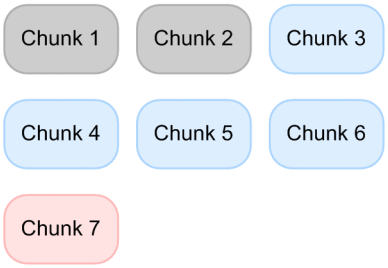
- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that did not become the current chunk and was never in normal status in the past
- The chunk that was the current chunk before the swap

(g) adbchgchunkstatus command execution example 7

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*g*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-16: Execution of the adbchgchunkstatus command under condition (g)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6	2014/09/06 01:00:00	2014/09/06 01:00:00
Chunk 7	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) are changed to chunks in wait status. Additionally, chunks in wait status that were in normal status in the past (chunks 3 and 4), as well as chunks in wait status that were never in normal status (chunks 6 and 7), are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Further, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*g*)

Current chunk

This is swapped to one of the chunks in wait status that was never in normal status, and that has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that was never in normal status in the past

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

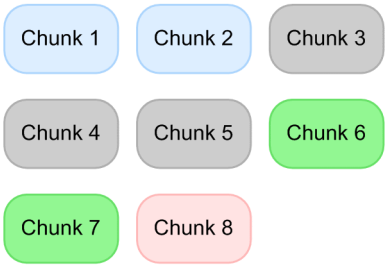
- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that did not become the current chunk and was never in normal status in the past
- The chunk that was the current chunk before the swap

(h) adbchgchunkstatus command execution example 8

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*h*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-17: Execution of the adbchgchunkstatus command under condition (h)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6		
Chunk 7		
Chunk 8	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

The current chunk (chunk 5) is changed to a chunk in wait status.

During this process, a new chunk is created and the current chunk is swapped (to chunk 8). Additionally, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*h*)

Current chunk

A new chunk is created, and that chunk becomes the current chunk.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- The newly-created chunk

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

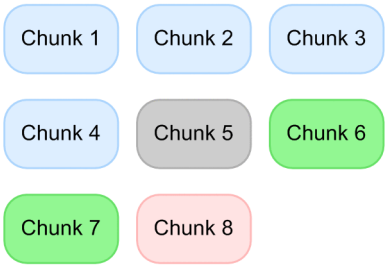
- The chunk that was the current chunk before the swap

(i) adbchgchunkstatus command execution example 9

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*i*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-18: Execution of the adbchgchunkstatus command under condition (i)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6		
Chunk 7		
Chunk 8	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

The current chunk (chunk 5) is changed to a chunk in wait status. Additionally, chunks in wait status that were in normal status in the past (chunks 3 and 4) are changed to chunks in normal status.

During this process, a new chunk is created, and the current chunk is swapped (to chunk 8). Additionally, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*i*)

Current chunk

A new chunk is created, and that chunk becomes the current chunk.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- The newly-created chunk

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

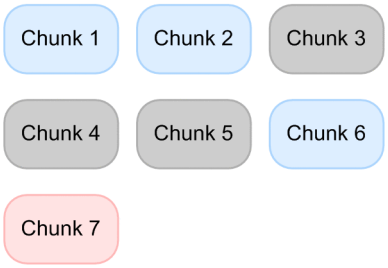
- The chunk that was the current chunk before the swap

(j) adbchgchunkstatus command execution example 10

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*j*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-19: Execution of the adbchgchunkstatus command under condition (j)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6	2014/09/06 01:00:00	2014/09/06 01:00:00
Chunk 7	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

The current chunk (chunk 5) is changed to a chunk in wait status. Additionally, chunks in wait status that were never in normal status (chunks 6 and 7) are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Additionally, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*j*)

Current chunk

This is swapped to one of the chunks in wait status that was never in normal status, and that has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that was never in normal status in the past

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

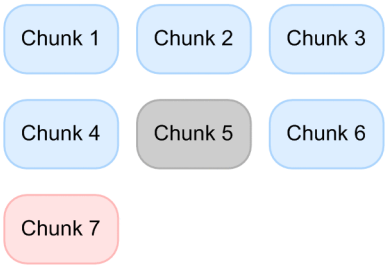
- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that did not become the current chunk and was never in normal status in the past
- The chunk that was the current chunk before the swap

(k) adbchgchunkstatus command execution example 11

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*k*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-20: Execution of the adbchgchunkstatus command under condition (k)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6	2014/09/06 01:00:00	2014/09/06 01:00:00
Chunk 7	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

The current chunk (chunk 5) is changed to a chunk in wait status. Additionally, chunks in wait status that were in normal status in the past (chunks 3 and 4), as well as chunks in wait status that were never in normal status (chunks 6 and 7), are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Additionally, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*k*)

Current chunk

Swapped to one of the chunks in wait status that was never in normal status, and that has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that was never in normal status in the past

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

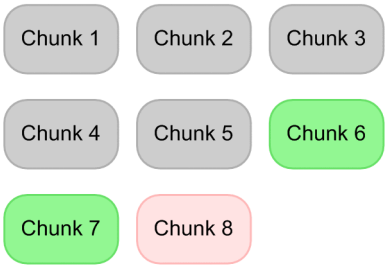
- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that did not become the current chunk and was never in normal status in the past
- The chunk that was the current chunk before the swap

(l) adbchgchunkstatus command execution example 12

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*l*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-21: Execution of the adbchgchunkstatus command under condition (l)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6		
Chunk 7		
Chunk 8	2014/09/06 01:00:00	

Legend:

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) and the current chunk (chunk 5) are changed to chunks in wait status.

During this process, a new chunk is created, and the current chunk is swapped (to chunk 8). Additionally, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*l*)

Current chunk

A new chunk is created and that chunk becomes the current chunk.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- The newly created chunk

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

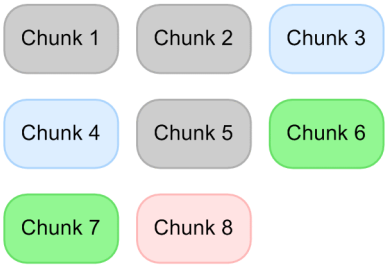
- The chunk that was the current chunk before the swap

(m) adbchgchunkstatus command execution example 13

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*m*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-22: Execution of the adbchgchunkstatus command under condition (m)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6		
Chunk 7		
Chunk 8	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

: Chunk in wait status that was never in normal status in the past

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) and the current chunk (chunk 5) are changed to chunks in wait status. Additionally, chunks in wait status that were in normal status in the past (chunks 3 and 4) are changed to chunks in normal status.

During this process, a new chunk is created and the current chunk is swapped (to chunk 8). Additionally, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*m*)

Current chunk

A new chunk is created, and that chunk becomes the current chunk.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- The newly created chunk

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

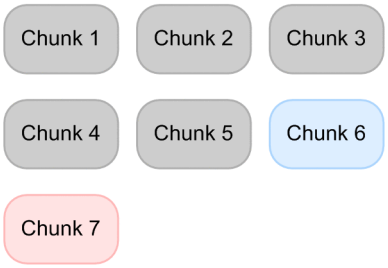
- The chunk that was the current chunk before the swap

(n) adbchgchunkstatus command execution example 14

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*n*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-23: Execution of the adbchgchunkstatus command under condition (n)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6	2014/09/06 01:00:00	2014/09/06 01:00:00
Chunk 7	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) and the current chunk (chunk 5) are changed to chunks in wait status. Additionally, chunks in wait status that were never in normal status (chunks 6 and 7) are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Further, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*n*)

Current chunk

Swapped to one of the chunks in wait status that was never in normal status, and that has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that was never in normal status in the past

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

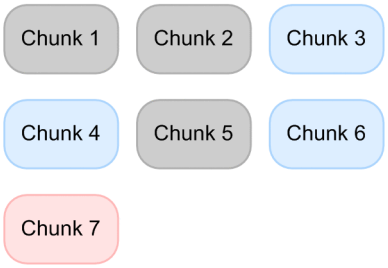
- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, that did not become the current chunk, and that was never in normal status in the past
- The chunk that was the current chunk before the swap

(o) adbchgchunkstatus command execution example 15

The following shows an example of executing the `adbchgchunkstatus` command on the chunks shown in Figure 11-9: Statuses of chunks before execution of the adbchgchunkstatus command, under condition (*o*) in Table 11-6: Execution conditions for changing the chunk status.

Figure 11-24: Execution of the adbchgchunkstatus command under condition (o)

■ Chunk configuration and chunk information



Chunk ID	Creation date and time	Swapping date and time
Chunk 1	2014/09/01 01:00:00	2014/09/02 01:00:00
Chunk 2	2014/09/02 01:00:00	2014/09/03 01:00:00
Chunk 3	2014/09/03 01:00:00	2014/09/04 01:00:00
Chunk 4	2014/09/04 01:00:00	2014/09/05 01:00:00
Chunk 5	2014/09/05 01:00:00	2014/09/06 01:00:00
Chunk 6	2014/09/06 01:00:00	2014/09/06 01:00:00
Chunk 7	2014/09/06 01:00:00	

Legend:

: Chunk in normal status

: Chunk in wait status that was in normal status in the past

: Chunk in normal status (current chunk)

Creation date and time: Date and time when a chunk was created
Swapping date and time: Date and time when swapping to the current chunk occurred

Explanation

Chunks in normal status (chunks 1 and 2) and the current chunk (chunk 5) are changed to chunks in wait status. Additionally, chunks in wait status that were in normal status in the past (chunks 3 and 4), and chunks in wait status that were never in normal status (chunks 6 and 7), are changed to chunks in normal status.

During this process, the current chunk is swapped (to chunk 7). Further, the chunk creation date and time and the date and time when the current chunk was swapped are set.

■ Processing results when the `adbchgchunkstatus` command is executed under condition (*o*)

Current chunk

Swapped to one of the chunks in wait status that was never in normal status, and that has the largest chunk ID specified for the `-n` option of the `adbchgchunkstatus` command.

Chunk creation date and time

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunk:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, and that was never in normal status in the past

Date and time when the current chunk was swapped

The date and time when the `adbchgchunkstatus` command was executed is set in the following chunks:

- One of the chunks in wait status that was specified for the `-n` option of the `adbchgchunkstatus` command, that did not become the current chunk, and that was never in normal status in the past
- The chunk that was the current chunk before the swap

(4) Operational example (replacing the data in a chunk)

This section provides an operational example for replacing the data in a chunk. To change the data in a chunk, you use the `adbchgchunkstatus` command to change the chunk status.

Operational example

The shop information of Area C has changed as a result of addition and consolidation of shops. The shop information of Area C is stored in a chunk (chunk ID: 3) in the shop list (`SHOPSLIST`). The information about the new shops

in Area C is stored in a CSV-format file. Therefore, it is necessary to replace all data in the chunk (chunk ID: 3) that stores the old shop information of Area C.

The old shop information of Area C can be viewed while the data is being replaced.

Overview of replacing the data in a chunk

The following is an overview of replacing the data in a chunk:

1. Background-import the new shop information of Area C.
2. Change the chunk status so that the old shop information of Area C cannot be viewed (while at the same time the new shop information of Area C can be viewed).
3. Delete the old shop information of Area C.

Information to check before starting data replacement

When you perform *Procedure for replacing the data in a chunk*, the current chunk changes. If you do not want the current chunk to change, you must check the chunk ID and comment of the current chunk. Check `CHUNK_ID` (chunk ID) and `CHUNK_COMMENT` (comment set for a chunk) as explained in (16) Checking the information about the current chunk in C.9 Searching system tables.

If the current chunk contains the data to be replaced, you do not need to check the chunk ID and comment of the current chunk.

Procedure for replacing the data in a chunk

1. Background-import the new shop information of Area C.

```
adbimport -u ADBUSER01 -p '#HelloHADB_01' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file02.txt
          -b --status wait
          SHOPSLIST
          /home/adbmanager/imp_file/imp_data_path02.txt
```

When you execute the `adbimport` command, specify the `--status wait` option. If you specify this option, the chunk from which the data is to be imported is placed in the wait status (in this status, the new shop information cannot be viewed).

> **Note**
>
> When background import is being executed, the old shop information can be viewed.

2. Check the chunk ID of the chunk that stores the imported new shop information.

The `KFAA51242-I` message is output during the background import in step 1. The chunk ID is displayed in this message.

```
KFAA51242-I A chunk was created. (chunk id = 5 )
```

In the preceding example, the imported new shop information is stored in the chunk whose chunk ID is 5.

3. Change the chunk status so that the old shop information of Area C cannot be viewed (while at the same time the new shop information of Area C can be viewed).

```
adbchgchunkstatus -u ADBUSER01 -p '#HelloHADB_01'
                  -n 5 -w 3 SHOPSLIST
```

Use the `adbchgchunkstatus` command to change the chunk status so that the old shop information cannot be viewed and the new shop information can be viewed. At this time, specify `5` for the `-n` option and `3` for the `-w` option. The chunk that stores the new shop information (chunk ID: 5) is placed in the normal status, causing the data in that chunk to be visible. The chunk that stores the old shop information (chunk ID: 3) is placed in the wait status, causing the data in that chunk to not be viewable.

4. Delete the old shop information of Area C.

```
PURGE CHUNK "SHOPSLIST" WHERE CHUNKID=3
```

Use the `PURGE CHUNK` statement to delete the data in the chunk that stores the old shop information (chunk ID: 3).

Note that if there is a transaction that references or updates a DB area that will be locked by the `PURGE CHUNK` statement, the `PURGE CHUNK` statement results in an error. To check whether there is a transaction that references or updates a DB area, execute the `adbls -d lock` command, and check whether there are connections that have locked DB areas. For details about the `adbls -d lock` command, see *adbls -d lock (Display the Status of Locked Resources)* in the manual *HADB Command Reference*.

In this example, when the preceding procedure is completed, the chunk whose chunk ID is 5 becomes the current chunk. To make the chunk that was the current chunk previously (before data replacement) the current chunk again, perform the procedure described in *Procedure for changing the current chunk*.

Procedure for changing the current chunk

1. Use the `adbimport` command to create an empty chunk (a chunk containing no data).

   Execute the `adbimport` command with the `-b` option to perform background import, specifying an empty file as the input data file.

2. Merge chunks.

   Use the `adbmergechunk` command to merge the empty chunk that you created in step 1 and the current chunk that you checked in *Information to check before starting data replacement*. At this time, for the `-m` option in the `adbmergechunk` command, specify the comment that you checked in *Information to check before starting data replacement*.

   When merging of the chunks finishes, the chunk that was the current chunk previously (before data replacement) becomes the current chunk again.

11.4.13 Checking whether a multi-chunk table needs to be reorganized

This section describes how to check whether a multi-chunk table needs to be reorganized.

Note

- For details about how to check whether a single-chunk table needs to be reorganized, see 11.1.9 Checking whether a single-chunk table needs to be reorganized.
- You do not have to reorganize data for archived chunks.

(1) Reason why table reorganization is necessary

If the following operations are repeatedly performed for a multi-chunk table, the retrieval performance and data storage efficiency of the table are degraded:

- Deleting rows with the `DELETE` statement
- Updating rows with the `UPDATE` statement
- Inserting rows with the `INSERT` statement
- Background import of a small amount of data by using the `adbimport` command

In such a case, if you reorganize the multi-chunk table, the area that has become unavailable in the data DB area is released, thus improving the retrieval performance and data storage efficiency of the table.

Retrieval performance and data storage efficiency of a multi-chunk table are degraded for the following reasons:

In a case where the multi-chunk table is a row store table

The reasons in the case of a single-chunk table also apply. For details about the reasons, see **In a case where the single-chunk table is a row store table** in (1) Reason why table reorganization is necessary in 11.1.9 Checking whether a single-chunk table needs to be reorganized.

In a case where the multi-chunk table is a column store table

The reasons in the case of a single-chunk table also apply. For details about the reasons, see **In a case where the single-chunk table is a column store table** in (1) Reason why table reorganization is necessary in 11.1.9 Checking whether a single-chunk table needs to be reorganized.

Reasons common to a row store table and column store table

If the `adbimport` command is repeatedly executed to perform background import of a small amount of data for a multi-chunk table, part of free space might become unavailable. Such space is free but unavailable (useless) because it is never reused. Such space degrades the data storage efficiency, resulting in lower retrieval performance of the table.

(2) How to check whether reorganization is necessary

This section describes how to check whether a multi-chunk table needs to be reorganized.

If the multi-chunk table is a row store table and SQL statements are used to perform update or deletion of rows repeatedly for the table, check whether the table needs to be reorganized as explained in (a) In a case where the multi-chunk table is a row store table.

If the multi-chunk table is a column store table and SQL statements are used to perform addition, update, or deletion of rows repeatedly for the table, check whether the table needs to be reorganized as explained in (b) In a case where the multi-chunk table is a column store table.

If the `adbimport` command is used to repeatedly perform background import of a small amount of data for the table, check whether the table needs to be reorganized as explained in (c) If a small amount of data is repeatedly added by background import.

(a) In a case where the multi-chunk table is a row store table

Calculate the data storage efficiency. If data storage is inefficient, reorganize the multi-chunk table. Check the data storage efficiency on a chunk basis. The following shows the procedure.

Procedure

1. Obtain the number of segments being used by the current chunk.

   Use the `adbdbstatus` command to output the usage information. Then, check the value for `Used_segments` (number of segments being used) by using `Chunk_ID` (chunk ID) as a key. For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

2. Obtain the number of segments in the data DB area based on the number of data items in the current chunk.

   First check the number of data items in the current chunk, and then obtain the number of segments in the data DB area.

   You can use the subcommand `#GETCOUNT` of the `adbsql` command to check the number of data items in the current chunk. Execute the subcommand `#GETCOUNT` of the `adbsql` command by using the chunk ID you checked

in step 1. For details about the `adbsql` command, see *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

For details about how to determine the number of segments in the data DB area, see (f) Determining the variable SGROWTBL (for a multi-chunk table) in (2) Explanation of variables in 5.8.1 Determining the total number of pages in the data DB area.

Note that the formula to determine the value of the *SGROWTBL* variable obtains the summation results for all chunks in all multi-chunk tables that are stored in the data DB area. In this step, obtain the value only for the target chunk for the target table.

3. Determine the data storage efficiency based on the obtained results.

   Determine the data storage efficiency in the chunks based on the results of steps 1 and 2. The following shows the formula:

   Formula

   data storage efficiency = result of step 2 ÷ result of step 1

   If the result of the preceding calculation is close to 0, data storage efficiency is poor. In such a case, reorganize the chunk. There are several reorganization operations. Determine the operation to be used by referring to (3) Selecting the reorganization method.

(b) In a case where the multi-chunk table is a column store table

Execute the `adbdbstatus` command, check *information about the need for reorganization*, and determine whether to reorganize the multi-chunk table. Use the following procedure for determination.

Procedure

1. Execute the `adbdbstatus` command with the `-d reorginfo` option specified to output information about the need for reorganization.
2. In the information about the need for reorganization, check the `Reorganization_necessity` section (whether reorganization is necessary).
   - If `Recommended` is output

     Reorganize the chunks for which `Recommended` was output. There are several reorganization operations. Determine the operation to be used by referring to (3) Selecting the reorganization method.
   - If `Not_recommended` is output

     You do not need to reorganize the chunks for which `Not_recommended` was output.

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

(c) If a small amount of data is repeatedly added by background import

Calculate the data storage efficiency. If data storage efficiency is poor, reorganize the multi-chunk table. The following shows the formula for calculating the data storage efficiency.

Formula

$$\text{Storage efficiency if a small amount of data is repeatedly added by background import} = \frac{A \times B}{C \times 4{,}096}$$

The closer the value obtained from the preceding formula is to 1, the better the storage efficiency of chunk data. Conversely, the closer the value obtained from the formula is to 0, the worse the storage efficiency of chunk data. If the result is close to 0, reorganize the multi-chunk table.

Explanation of variables

A

The value to be substituted differs depending on the reorganization method.

- Chunk-based reorganization

  Substitute the number of pages used in the chunk.

  To check the number of used pages in the chunk, execute the `adbdbstatus` command with the `-d used` option specified. The *information about the usage of DB areas, tables, and indexes* can be output. From the *information about the usage of DB areas, tables, and indexes*, check the content of `Used_pages` based on `Chunk_ID` of the chunk for which you want to check the storage efficiency.

- Reorganization of an entire table

  Substitute the number of pages used in the table.

  To check the number of pages used in the table, execute the `adbdbstatus` command to output the *table summary information*. In the *table summary information*, check the value of `Used_pages`.

B

Page size of the data DB area that stores a multi-chunk table (kilobytes)

To check the page size of a data DB area, execute the `adbdbstatus` command with the `-c dbarea` option specified. The *DB area summary information* can be output. From the *DB area summary information*, check the content of `Page_size` for the data DB area whose page size you want to check.

C

The value to be substituted differs depending on the reorganization method.

- Chunk-based reorganization

  Substitute the number of segments used in the chunk.

  To check the number of used segments in a chunk, execute the `adbdbstatus` command with the `-d used` option specified. The *information about the usage of DB areas, tables, and indexes* can be output. From the *information about the usage of DB areas, tables, and indexes*, check the content of `Used_segments` based on `Chunk_ID` of the chunk for which you want to check the storage efficiency.

  Important

  The data storage efficiency does not improve even by reorganizing a chunk in which only one segment is used. If the number of segments used in the reorganization-target chunk is 1 or close to 1, use either of the following methods to reorganize the data:

  - Merge multiple chunks, and then reorganize the resulting chunk as explained in 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization.
  - Reorganize the entire multi-chunk table as explained in (2) In a case where the table data can be stored to one chunk after reorganization in 11.4.15 Reorganizing a multi-chunk table: Reorganization of an entire table.

- Reorganization of an entire table

  Substitute the number of segments used in the table.

To check the number of segments used in the table, execute the `adbdbstatus` command to output the *table summary information*. In the *table summary information*, check the value of `Used_segments`.

(3) Selecting the reorganization method

There are several methods for reorganizing a multi-chunk table, as shown in the following table. You can select a method only if the conditions indicated are met.

> **Important**
>
> Note that if you reorganize an entire multi-chunk table (especially when the table has many data items), reorganization might take time. Therefore, we recommend that you periodically perform chunk-based reorganization rather than reorganizing the entire multi-chunk table (that is, the method in item 1 in the following table is recommended).

Table 11-7: Reorganization methods for a multi-chunk table

No.	Basis of reorganization	Availability of retrieval in the table and chunk configuration	Condition required for reorganization	Reorganization method
1	Chunk-based	**(1)** Retrieval in the table during reorganization is possible.[#1] **(2)** The configuration, statuses, and comments of the chunks before reorganization can be maintained.	1. Free space in the data DB area Both the reorganization-target data and reorganization-result data must be temporarily stored. Therefore, the data DB area must have as large a free space as the reorganization-target chunk. For example, to reorganize a chunk that stores 1 TB of data, free space of 1 TB is required in the data DB area.[#2] 2. Space required at the destination of the output data file There must be free disk space sufficient for storing the output data file to which the chunk data will be exported. 3. Space required at the destination of the temporary work file A temporary work file is temporarily created when data is imported. Therefore, there must be free disk space sufficient for storing the temporary work file.	Perform reorganization by using the procedure described in (1) Reorganizing a table while continuing retrieval in 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization.
2		**(1)** Retrieval in the table during reorganization is not possible. **(2)** The configuration, statuses, and comments of the chunks before reorganization can be maintained.	Conditions 2 and 3 in item 1 must be met. If only the current chunk is to be reorganized, the data DB area must have as large a free space as the data stored in the current chunk.	Perform reorganization by using the procedure described in (2) Reorganizing a table after stopping retrieval in 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization.
3	Entire table	**(1)** Retrieval in the table during reorganization is possible.[#1] **(2)** The configuration, statuses, and	1. Free space in the data DB area Both the reorganization-target data and reorganization-result data must be temporarily stored. Therefore, the data DB area must have as large a free space as the largest chunk in the reorganization-target table.	Perform reorganization by using the procedure described in (1) In a case where you want to maintain the current chunk configuration after reorganization in 11.4.15

No.	Basis of reorganization	Availability of retrieval in the table and chunk configuration	Condition required for reorganization	Reorganization method
		comments of the chunks before reorganization can be maintained.	For example, if the table has three chunks whose sizes are 1 TB, 2 TB, and 3 TB respectively, the data DB area must have free space of 3 TB. 2. Space required at the destination of the output data file Because the data is exported in units of chunks, there must be free disk space sufficient for storing the output data file to which the chunk data will be exported. 3. Space required at the destination of the temporary work file The data is imported in units of chunks and a temporary work file is temporarily created when the data is imported. Therefore, there must be free disk space sufficient for storing the temporary work file.	Reorganizing a multi-chunk table: Reorganization of an entire table.
4		**(1)** Retrieval in the table during reorganization is not possible. **(2)** Chunks can be merged into one.	1. Space required at the destination of the output data file All the data in the table is exported at one time. There must be free disk space sufficient for storing the output data file to which all table data will be exported. 2. Space required at the destination of the temporary work file All the data in the table is imported at one time. A temporary work file is temporarily created when all table data is imported. Therefore, there must be free disk space sufficient for storing the temporary work file.	Perform reorganization by using the procedure described in (2) In a case where the table data can be stored to one chunk after reorganization in 11.4.15 Reorganizing a multi-chunk table: Reorganization of an entire table.

#1

The `PURGE CHUNK` statement is executed in a step of the reorganization procedure. Retrieval cannot be performed in the table while the `PURGE CHUNK` statement is being executed.

#2

If there are two chunks to be reorganized and their sizes are 1 TB and 2 TB respectively, the data DB area must have free space of 2 TB.

11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization

This section describes how to perform chunk-based reorganization for a multi-chunk table. You can use either of the following methods to perform chunk-based reorganization. Normally, use method 1 to perform chunk-based reorganization.

1. Reorganizing a table while continuing retrieval

   With this method, you can retrieve (but cannot update) the data in the table that is being reorganized.

   Note that this method requires both the reorganization-target data and reorganization-result data to be stored temporarily. Therefore, the data DB area that stores the reorganization-target table must have free space for storing both types of data. If the free space is insufficient, you cannot perform this method. Therefore, before you perform this method, use the `adbdbstatus` command to compare the size of the reorganization-target chunk with the size

of free space in the data DB area that stores the table. For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

For details about reorganization using this method, see (1) Reorganizing a table while continuing retrieval.

2. Reorganizing a table after stopping retrieval

   If you cannot use method 1 because the data DB area that stores the reorganization-target table does not have sufficient free space, use this method to perform chunk-based reorganization. With this method, you cannot retrieve or update the data in the table that is being reorganized.

   For details about reorganization using this method, see (2) Reorganizing a table after stopping retrieval.

   If only the current chunk is to be reorganized, the data DB area must have as large a free space as the data stored in the current chunk. If the free space is insufficient in the data DB area, secure sufficient free space as explained in 11.10.6 Securing free space in a data DB area.

> **Note**
>
> HADB provides a sample shell script that performs chunk-based reorganization. You can use the sample shell script for only a multi-chunk table that is a column store table. For details, see 11.4.16 Reorganizing a multi-chunk table: Reorganization using a sample shell script.

(1) Reorganizing a table while continuing retrieval

The procedure for chunk-based reorganization is as follows.

Procedure

1. Make sure that no application programs and commands can update the reorganization-target table.

   Perform the following operations to prevent application programs and commands from updating the reorganization-target table:

   - If commands and jobs that update the reorganization-target table (such as the `adbimport` and `adbmergechunk` commands that are periodically executed) are executing on the HADB server machine, stop all of them.
   - Stop all application programs and commands that access the HADB server to update the reorganization-target table.

   We recommend that you perform both of the preceding operations.

2. Obtain the information about the reorganization-target chunk.

   Perform data retrieval in the system table `STATUS_CHUNKS` to obtain the following information about the reorganization-target chunk. For details about how to retrieve data from the `STATUS_CHUNKS` table, see (3) Checking the information about all chunks in a table based on a table name in C.9 Searching system tables.

   - `CHUNK_COMMENT` (Comment set for a chunk)
   - `CHUNK_STATUS` (Chunk status)

3. Obtain the information about the current chunk.

   Perform data retrieval in the system table `STATUS_CHUNKS` to obtain the following information about the current chunk. For details about how to retrieve data from the `STATUS_CHUNKS` table, see (16) Checking the information about the current chunk in C.9 Searching system tables.

   - `CHUNK_ID` (Chunk ID)

     Check whether the value of this item is the same as the chunk ID of the reorganization-target chunk (whether the reorganization-target chunk is the current chunk).

- CHUNK_COMMENT (Comment set for a chunk)

4. Export the data in a multi-chunk table in units of chunks.

   Use the adbexport command with the -c option specified to export the data in the multi-chunk table in units of chunks. For details about how to export data in units of chunks, see 11.4.5 Exporting data in units of chunks.

5. Use the background-import facility to import the exported data as chunks in wait status.

   Perform background import of the data exported in step 4 by executing the adbimport command with the following options specified:

   - -b option
   - --status wait option

     If the --status wait option is specified, the command creates a chunk in wait status and stores data in the chunk.
   - -m option

     Specify the chunk comment that you checked in step 2.

   For details about background import that creates a chunk in wait status, see 11.4.4 Temporarily excluding data to be imported to a multi-chunk table from retrieval (creating a chunk in wait status).

   > **Important**
   >
   > When you execute the adbimport command, make sure that the value you specify for the import option (adb_import_rthd_num) satisfies the formula shown in (3) Estimating the value to be specified for the import option. If you specify a value that does not satisfy the formula, the data storage efficiency might degrade.

6. Change the chunk's status.

   If the reorganization-target chunk that you checked in step 2 is in wait status, you can skip this step. Proceed to step 7.

   If the reorganization-target chunk that you checked in step 2 is in normal status, execute the adbchgchunkstatus command with both the -w and -n options specified. Change the status of two types of chunks at the same time.

   - For the -w option, specify the chunk that manages the data exported in step 4 (the chunk subject to reorganization).

     Change the status of the chunk from normal status to wait status.
   - For the -n option, specify the chunk that was imported by background import processing in step 5 and is in wait status.

     Change the status of the chunk from wait status to normal status.

   For details about how to change the chunk status, see 11.4.12 Changing the chunk status.

7. Delete the chunk subject to reorganization.

   Use the PURGE CHUNK statement to delete the chunk that you exported in step 4. Note that you cannot execute the PURGE CHUNK statement simultaneously with data retrieval from the multi-chunk table. Therefore, execute the PURGE CHUNK statement for the multi-chunk table when no operations are being performed for the table.

   For details about how to delete chunks by using the PURGE CHUNK statement, see 11.4.6 Deleting data in units of chunks.

   Notes on the current chunk

   - If the current chunk is reorganized, you can skip steps 8 and 9. Proceed to step 10.
   - When a chunk other than the current chunk is reorganized, the current chunk changes. If you want to change the current chunk back to the one before reorganization, perform steps 8 and 9.

- If it is not necessary to change the current chunk to the one that was the current chunk before reorganization, skip steps 8 and 9. Proceed to step 10.

8. Use the `adbimport` command to create an empty chunk (a chunk containing no data).

   Execute the `adbimport` command with the `-b` option to perform background import, specifying an empty file as the input data file.

9. Merge chunks.

   Use the `adbmergechunk` command to merge the empty chunk that you created in step 8 and the chunk that was the current chunk before reorganization. At this time, for the `-m` option in the `adbmergechunk` command, specify the comment (that you checked in step 3) on the chunk that was the current chunk before reorganization.

   When merging of the chunks finishes, the chunk that was the current chunk before reorganization becomes the current chunk again.

10. Make sure that the application programs and commands can update the reorganization-target table.

    Perform the following operation to enable application programs and commands to update the reorganization-target table:

    - If you stopped commands and jobs running on the HADB server machine in step 1

      Restart the commands and jobs that were stopped.

    - If you stopped application programs and commands that access the HADB server in step 1

      Restart the application programs and commands that you stopped.

Important

- If you perform chunk-based reorganization in accordance with the procedure described here, the chunk ID of the reorganization-target chunk changes after reorganization. However, the configuration, status, and comment of the chunk do not change.
- Do not execute an update SQL statement or a command that updates a table (such as the `adbimport` or `adbmergechunk` command) for a table that is being reorganized. If you do so, the changes made by the SQL statement or command might be lost after reorganization.

(2) Reorganizing a table after stopping retrieval

The procedure for chunk-based reorganization is as follows.

Procedure

1. Make sure that no application programs and commands can access the reorganization-target table.

   Perform the following operations to prevent application programs and commands from accessing the reorganization-target table:

   - Execute the `adbchgsrvmode` command with the `--offline` option specified to change the HADB server operation mode to offline mode. Furthermore, if commands and jobs that update the target table (such as the `adbimport` and `adbmergechunk` commands that are periodically executed) are executing on the HADB server machine, stop all of them.
   - Stop all application programs and commands that access the HADB server.

   We recommend that you perform both of the preceding operations.

2. Obtain the information about the reorganization-target chunk.

Perform data retrieval in the system table STATUS_CHUNKS to obtain the following information about the reorganization-target chunk. For details about how to retrieve data from the STATUS_CHUNKS table, see (3) Checking the information about all chunks in a table based on a table name in C.9 Searching system tables.

- CHUNK_COMMENT (Comment set for a chunk)
- CHUNK_STATUS (Chunk status)

When reorganizing multiple chunks, obtain the information about all the chunks to be reorganized.

3. Obtain the information about the current chunk.

   Perform data retrieval in the system table STATUS_CHUNKS to obtain the following information about the current chunk. For details about how to retrieve data from the STATUS_CHUNKS table, see (16) Checking the information about the current chunk in C.9 Searching system tables.

   - CHUNK_ID (Chunk ID)

     Check whether the value of this item is the same as the chunk ID of the reorganization-target chunk (whether the reorganization-target chunks include the current chunk).
   - CHUNK_COMMENT (Comment set for a chunk)

   Important

   About the subsequent procedure

   - If the reorganization-target chunks do not include the current chunk:
     Perform steps 4 to 6 for each chunk. Then, proceed to step 7.
   - If the reorganization-target chunks include the current chunk and other chunks:
     • First, perform steps 4 to 6 for each of the chunks that are not the current chunk.
     • Then, perform steps 4 to 6 for the current chunk.
     • After you complete the preceding operations, proceed to step 8.
   - If the current chunk is the only reorganization-target chunk:
     Perform step 4, step 6, and step 5, in this order. Then, proceed to step 8.

4. Export the data in a multi-chunk table in units of chunks.

   Use the adbexport command with the -c option specified to export the data in the multi-chunk table in units of chunks. For details about how to export data in units of chunks, see 11.4.5 Exporting data in units of chunks.

5. Delete the chunk subject to reorganization.

   Use the PURGE CHUNK statement to delete the reorganization-target chunk from which data was exported in step 4.

   For details about how to delete chunks by using the PURGE CHUNK statement, see 11.4.6 Deleting data in units of chunks.

6. Import the exported data into the multi-chunk table by background import.

   - If the reorganization-target chunk that you checked in step 2 is in normal status:
     Perform background import of the data exported in step 4 by executing the adbimport command with the following options specified:
     - -b option
     - -m option

       Specify the chunk comment that you checked in step 2.

     For details about background import, see 11.4.2 Storing data in a multi-chunk table (background import).

- If the reorganization-target chunk that you checked in step 2 is in wait status:

  Perform background import of the data exported in step 4 by executing the `adbimport` command with the following options specified:

  - `-b` option
  - `--status wait` option

    If the `--status wait` option is specified, the command creates a chunk in wait status and stores data in the chunk.
  - `-m` option

    Specify the chunk comment that you checked in step 2.

  For details about background import that creates a chunk in wait status, see 11.4.4 Temporarily excluding data to be imported to a multi-chunk table from retrieval (creating a chunk in wait status).

> **Important**
>
> When you execute the `adbimport` command, make sure that the value you specify for the import option (`adb_import_rthd_num`) satisfies the formula shown in (3) Estimating the value to be specified for the import option. Specifying a value that does not satisfy the formula might adversely affect data storage efficiency.

7. Change the current chunk.

   The status of the current chunk changes when reorganization is performed. If you want to change the current chunk back to the one before reorganization, perform the following operations. If it is not necessary to change the current chunk back to the one before reorganization, you can skip the following operations. Proceed to step 8.

   - Use the `adbimport` command to create an empty chunk (a chunk containing no data).

     Execute the `adbimport` command with the `-b` option to perform background import, specifying an empty file as the input data file.
   - Merge chunks.

     Use the `adbmergechunk` command to merge the empty chunk that you created by the preceding operations and the chunk that was the current chunk before reorganization. At this time, for the `-m` option in the `adbmergechunk` command, specify the comment (that you checked in step 3) on the chunk that was the current chunk before reorganization.

   When merging of the chunks finishes, the chunk that was the current chunk before reorganization becomes the current chunk again.

8. Make sure that application programs and commands can access the reorganization-target table.

   Perform the following operation to enable application programs and commands to access the reorganization-target table:

   - If you changed the HADB server's operation mode to offline mode in step 1

     Execute the `adbchgsrvmode` command with the `--normal` option specified to change the HADB server operation mode to normal mode. If you stopped commands and jobs that were running on the HADB server machine, restart them.
   - If you stopped application programs and commands that access the HADB server in step 1

     Restart the application programs and commands that you stopped.

Important

- If you perform chunk-based reorganization in accordance with the procedure described here, the chunk ID of the reorganization-target chunk changes after reorganization. However, the configuration, status, and comment of the chunk do not change.
- Do not execute an update SQL statement or a command that updates a table (such as the `adbimport` or `adbmergechunk` command) for a table that is being reorganized. If you do so, the changes made by the SQL statement or command might be lost after reorganization.

(3) Estimating the value to be specified for the import option

The formula for estimating the value to be specified for the import option (`adb_import_rthd_num`) differs depending on the reason why reorganization was performed. Use the appropriate formula according to the reason.

■ If reorganization was performed because update and deletion of rows were performed repeatedly (when the multi-chunk table was a row store table)

When you execute the `adbimport` command, make sure that the value you specify for the import option (`adb_import_rthd_num`) satisfies the following formula. If you specify a value that does not satisfy the formula, the data storage efficiency might degrade.

Formula

```
value-specified-for-import-option-adb_import_rthd_num ≤
    ↑ number-of-segments-to-be-reorganized × data-storage-efficiency ↑ + 1
```

number-of-segments-to-be-reorganized

Substitute the number of segments used in the reorganization-target chunk.

Execute the `adbdbstatus` command with the `-d used` option specified to output the *information about the usage of DB areas, tables, and indexes*. Then, check the information output as `Used_segments` (number of used segments for each chunk) by using `Chunk_ID` (chunk ID) as a key.

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

If you reorganize the entire table, obtain the total number of segments used in all chunks in the table.

data-storage-efficiency

Substitute the data storage efficiency of the reorganization-target chunk. Substitute the value obtained in (a) In a case where the multi-chunk table is a row store table in (2) How to check whether reorganization is necessary in 11.4.13 Checking whether a multi-chunk table needs to be reorganized.

If you reorganize the entire table, calculate the data storage efficiency of all chunks in the table.

■ If reorganization was performed because addition, update, and deletion of rows were performed repeatedly (when the multi-chunk table was a column store table)

When you execute the `adbimport` command, make sure that the value you specify for the import option (`adb_import_rthd_num`) satisfies the following formula. If you specify a value that does not satisfy the formula, the data storage efficiency might degrade.

Formula

```
value-specified-for-import-option-adb_import_rthd_num ≤
    ↑ number-of-segments-to-be-reorganized × number-of-rows-to-be-reorganized
       ÷ number-of-rows-stored-in-column-data-segment ↑ + 1
```

number-of-segments-to-be-reorganized

Substitute the number of column-data segments used in the reorganization-target chunk. Execute the `adbdbstatus` command with the `-d used` option specified to output the *information about the usage of DB areas, tables, and indexes*. Then, check the information output as `Used_segments` (number of used segments) for the rows whose `Segment_type` (segment type) is `Column_data`, by using `Chunk_ID` (chunk ID) as a key.

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

If you reorganize the entire table, obtain the total number of column-data segments used in all chunks in the table.

number-of-rows-to-be-reorganized

Substitute the number of data items in the reorganization-target chunk.

If you reorganize the entire table, substitute the number of data items in the table.

You can obtain the number of data items in a chunk or table by executing the `#GETCOUNT` subcommand of the `adbsql` command.

number-of-rows-stored-in-column-data-segment

Substitute the number of rows that were imported into the reorganization-target chunk by using the `adbimport` command. Alternatively, substitute the number of rows converted from row store format to column store format by using the updated-row columnizing facility.

Execute the `adbdbstatus` command to obtain *information about the need for reorganization*. The value of `Column_data_num` for the reorganization-target chunk is the number of rows stored in the column-data segment.

To reorganize an entire table, sum the `Column_data_num` values for all chunks in the table.

■ **If reorganization was performed because a small amount of data was repeatedly added by background import**

When you execute the `adbimport` command, make sure that the value you specify for the import option (`adb_import_rthd_num`) satisfies the following formula. If you specify a value that does not satisfy the formula, the data storage efficiency might degrade.

Formula

```
value-specified-for-import-option-adb_import_rthd_num ≤
    ↑ number-of-segments-to-be-reorganized × data-storage-efficiency ↑ + 1
```

number-of-segments-to-be-reorganized

- To reorganize a row store table:

  Substitute the number of segments used in the reorganization-target chunk.

  Execute the `adbdbstatus` command with the `-d used` option specified to output the *information about the usage of DB areas, tables, and indexes*. Then, check the information output as `Used_segments` (number of used segments for each chunk) by using `Chunk_ID` (chunk ID) as a key.

  For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

  If you reorganize the entire table, obtain the total number of used segments of all chunks in the table.

- To reorganize a column store table:

  Substitute the number of column-data segments used in the reorganization-target chunk. Execute the `adbdbstatus` command with the `-d used` option specified to output the *information about the usage of DB areas, tables, and indexes*. Then, check the information output as `Used_segments` (number of used segments) for the rows whose `Segment_type` (segment type) is `Column_data`, by using `Chunk_ID` (chunk ID) as a key.

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

If you reorganize the entire table, obtain the total number of used column-data segments in all chunks in the table.

data-storage-efficiency

Substitute the data storage efficiency of the reorganization-target chunk. Substitute the value obtained in (c) If a small amount of data is repeatedly added by background import in (2) How to check whether reorganization is necessary in 11.4.13 Checking whether a multi-chunk table needs to be reorganized.

If you reorganize the entire table, calculate the data storage efficiency of all chunks in the table.

11.4.15 Reorganizing a multi-chunk table: Reorganization of an entire table

This section describes how to reorganize an entire multi-chunk table.

Important

Note that if you reorganize an entire multi-chunk table (especially when the table has many data items), reorganization might take time. Therefore, we recommend that you periodically reorganize only the chunks that need to be reorganized rather than reorganizing the entire multi-chunk table.

You can use either of the following methods to perform reorganization of an entire multi-chunk table. Normally, use method 1 to perform reorganization of an entire multi-chunk table.

1. Method for maintaining the chunk configuration after reorganization

   With this method, you can retrieve (but cannot update) the data in the table that is being reorganized. The configuration, statuses, and comments of the chunks before reorganization can be maintained.

   Note that this method requires both the reorganization-target data and reorganization-result data to be stored temporarily. Therefore, the data DB area that stores the reorganization-target table must have free space for storing both types of data. If the free space is insufficient, you cannot perform this method. Therefore, before you perform this method, use the `adbdbstatus` command to compare the size of the reorganization-target chunk with the size of free space in the data DB area that stores the table. For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

   For details about reorganization using this method, see (1) In a case where you want to maintain the current chunk configuration after reorganization.

2. Method for storing the table data to a single chunk after reorganization

   If you cannot use method 1 because the data DB area that stores the reorganization-target table does not have sufficient free space, use this method to perform reorganization of the table. With this method, you cannot retrieve or update the data in the table that is being reorganized. Because all data in the table is stored to a single chunk after reorganization, the configuration, statuses, and comments of the chunks cannot be maintained.

   For details about reorganization using this method, see (2) In a case where the table data can be stored to one chunk after reorganization.

   If only the current chunk is to be reorganized, the data DB area must have as large a free space as the data stored in the current chunk.

(1) In a case where you want to maintain the current chunk configuration after reorganization

If you reorganize a table by using the method described in this section, the configuration, statuses, and comments of the chunks will be maintained in the resulting table. The chunk IDs before reorganization cannot be maintained.

The table reorganization procedure is as follows.

Procedure

1. Make sure that no application programs or commands can update the reorganization-target table.

   Perform the following operations to prevent application programs and commands from updating the reorganization-target table:

   - If commands and jobs that update the reorganization-target table (such as the `adbimport` and `adbmergechunk` commands that are periodically executed) are executing on the HADB server machine, stop all of them.
   - Stop all application programs and commands that access the HADB server to update the reorganization-target table.

   We recommend that you perform both of the preceding operations.

2. Obtain the information about all chunks in the reorganization-target table.

   Perform data retrieval in the system table `STATUS_CHUNKS` by using SQL statements to obtain the information about all chunks in the reorganization-target table. Obtain the following types of information:

   - `CHUNK_ID` (Chunk ID)
   - `CHUNK_COMMENT` (Comment set for a chunk)
   - `CHUNK_STATUS` (Chunk status)
   - `CREATE_TIME` (Creation date and time of a chunk)
   - `SWAP_TIME` (Date and time the current chunk was swapped)

   For examples of SQL statements that retrieve the preceding types of information, see (3) Checking the information about all chunks in a table based on a table name in C.9 Searching system tables.

   > **Important**
   >
   > Check the current chunk. The current chunk is the chunk for which `CREATE_TIME` (creation date and time of a chunk) is set and `SWAP_TIME` (date and time the current chunk was switched) is a null value.

3. Perform reorganization for each chunk.

   Repeat steps a to d for each chunk in the reorganization-target table, so that the table is reorganized on a chunk basis.

   > **Important**
   >
   > Reorganize the current chunk last.

   a. Export data for each chunk.

   Execute the `adbexport` command with the `-c` option specified to export data for each chunk. For details about how to export data in units of chunks, see 11.4.5 Exporting data in units of chunks.

   b. Use the background-import facility to import the exported data as chunks in wait status.

To perform background import of the data that was exported in step a, execute the `adbimport` command with the following options specified:

- `-b` option
- `--status wait` option

  If you execute the preceding command with the `--status wait` option specified, a chunk in wait status is created and data is stored in it.
- `-m` option

  Specify the comment that is set for the chunk that you checked in step 2.

For details about background import that creates a chunk in wait status, see 11.4.4 Temporarily excluding data to be imported to a multi-chunk table from retrieval (creating a chunk in wait status).

> **Important**
>
> When you execute the `adbimport` command, make sure that the value you specify for the import option (`adb_import_rthd_num`) satisfies the formula shown in (3) Estimating the value to be specified for the import option in 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization. Specifying a value that does not satisfy the formula might adversely affect data storage efficiency.

c. Change the chunk's status.

If the reorganization-target chunk that you checked in step 2 is in wait status, you can skip this step. Proceed to step d.

If the reorganization-target chunk that you checked in step 2 is in normal status, execute the `adbchgchunkstatus` command with the `-w` and `-n` options specified. Two chunk statuses will be changed simultaneously.

- For the `-w` option, specify the chunk that stores the data exported in step a (reorganization-target chunk). The status of this chunk will change from normal status to wait status.
- For the `-n` option, specify the chunk that was imported by background import in step b and is placed in wait status. The status of this chunk will change from wait status to normal status.

For details about how to change the chunk status, see 11.4.12 Changing the chunk status.

d. Delete the chunk subject to reorganization.

Use the `PURGE CHUNK` statement to delete the data of the chunk that you exported in step a. Note that you cannot execute the `PURGE CHUNK` statement simultaneously with data retrieval from the multi-chunk table. Therefore, execute the `PURGE CHUNK` statement for the multi-chunk table when no operations are being performed for the table.

For details about how to delete chunks by using the `PURGE CHUNK` statement, see 11.4.6 Deleting data in units of chunks.

4. Make sure that the application programs and commands can update the reorganization-target table.

After the operation in step 3 is completed for all chunks, use the following operation to allow the application programs and commands to update the reorganization-target table:

- If you stopped commands and jobs running on the HADB server machine in step 1

  Restart the commands and jobs that were stopped.
- If you stopped application programs and commands that access the HADB server in step 1

  Restart the application programs and commands that you stopped.

Important

Do not execute an update SQL statement or a command that updates a table (such as the `adbimport` or `adbmergechunk` command) for a table that is being reorganized. If you do so, the changes made by the SQL statement or command might be lost after reorganization.

(2) In a case where the table data can be stored to one chunk after reorganization

If you reorganize a table by using the method described in this subsection, the resulting table will have only one chunk. Therefore, the current configuration, statuses, comments and chunk IDs of the chunks cannot be maintained.

The table reorganization procedure is as follows.

Procedure

1. Make sure that no application programs and commands can access the reorganization-target table.

   Perform the following operations to prevent application programs and commands from accessing the reorganization-target table:

   - Execute the `adbchgsrvmode` command with the `--offline` option specified to change the HADB server operation mode to offline mode. Furthermore, if commands and jobs that update the target table (such as the `adbimport` and `adbmergechunk` commands that are periodically executed) are executing on the HADB server machine, stop all of them.
   - Stop all application programs and commands that access the HADB server.

   We recommend that you perform both of the preceding operations.

2. Export the data from the table.

   Export the data of the table by executing the `adbexport` command in which the reorganization-target table is specified for the `-n` option.

   For details about the `adbexport` command, see *adbexport (Export Data)* in the manual *HADB Command Reference*.

3. Import the exported data in the creation mode.

   Execute the `adbimport` command with the `-d` option specified to import the data that was exported in step 2. If the `-d` option is specified, the data is imported in creation mode. All existing table data is deleted, and then the data exported in step 2 is imported.

   For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

   Important

   When you execute the `adbimport` command, make sure that the value you specify for the import option (`adb_import_rthd_num`) satisfies the formula shown in (3) Estimating the value to be specified for the import option in 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization. Specifying a value that does not satisfy the formula might adversely affect data storage efficiency.

4. Make sure that application programs and commands can access the reorganization-target table.

   Perform the following operation to enable application programs and commands to access the reorganization-target table:

- If you changed the HADB server's operation mode to offline mode in step 1

  Execute the `adbchgsrvmode` command with the `--normal` option specified to change the HADB server operation mode to normal mode. If you stopped commands and jobs that were running on the HADB server machine, restart them.
- If you stopped application programs and commands that access the HADB server in step 1

  Restart the application programs and commands that you stopped.

11.4.16 Reorganizing a multi-chunk table: Reorganization using a sample shell script

HADB provides a sample shell script (`$ADBDIR/sample/reorg_column_sample.sh`) that performs chunk-based reorganization for the chunks that need to be reorganized in a multi-chunk table. This section describes how to reorganize a table by using this sample shell script and the processing executed by the sample shell script.

Important

- You can use this sample shell script for only a multi-chunk table that is a column store table. You cannot use this sample shell script for a multi-chunk table that is a row store table.
- Note that this sample shell script is provided for only demonstration and its operation is not guaranteed. Before you use the sample shell script, always examine its content, and, if necessary, modify the script. Failure to examine the content of the sample shell script might cause an error that damages the database.
- A table that is being reorganized by using the sample shell script cannot be accessed.

(1) Procedure for table reorganization using a sample shell script

The following shows the procedure for table reorganization using a sample shell script.

Procedure

1. Check the size of free space in the data DB area.

   To execute the sample shell script, the data DB area that stores the reorganization-target table must have as large a free space as the data stored in the current chunk.

   If the free space is insufficient in the data DB area, secure sufficient free space as explained in 11.10.6 Securing free space in a data DB area.
2. Prepare a work directory.

   Note that this directory must be empty. This directory will store files that the HADB server will create during table reorganization.
3. Execute the sample shell script.

   The following shows the command syntax of the sample shell script.

   ■ Command syntax of the sample shell script

```
reorg_column_sample.sh user password schema table workdir
```

Table 11-8: Explanation of arguments

No.	Argument	Explanation
1	*user*	Specify the authorization identifier of the HADB user.

No.	Argument	Explanation
2	*password*	Specify the password of the HADB user.
3	*schema*	Specify the schema name of the column store table to be reorganized.
4	*table*	Specify the table name of the column store table to be reorganized.
5	*workdir*	Specify the work directory. Make sure that the directory you specify is empty. While the sample shell script is being executed, the following files are generated in the work directory. When execution of the sample shell script ends, all of these files are deleted. • File containing the execution results of the `adbdbstatus` command • Output data path file • Output data file for the chunk being reorganized • Input data file for creating an empty chunk • Input data path file for creating an empty chunk

Example

The following shows an example of the sample shell script for reorganizing the data in a column store table on a chunk basis. The specified table name and other items are as follows:

- Authorization identifier and schema name: `ADBUSER01`
- Password: `ADBPASSWORD`
- Processing-target column store table: `T1`
- Work directory: `/home/adbmanager/work_dir`

■ Example of specifying commands in the sample shell script

```
reorg_column_sample.sh ADBUSER01 ADBPASSWORD ADBUSER01 T1 /home/adbmanager/work_di
r
```

Important

- If an error occurs in execution of a command that is called in the sample shell script, always check the details of the error. Troubleshoot the error, if necessary, and then use the sample shell script.
- In the sample shell script, you cannot specify command arguments (such as the user name and table name) that include spaces.

(2) Processing that is executed by the sample shell script

When the sample shell script is executed, it stops all access to the target table and reorganizes all chunks that need to be reorganized.

The following shows an overview of the processing that is executed by the sample shell script.

■ Overview of processing

1. The `adbchgsrvmode --offline` command is executed, and the HADB server operation mode changes to offline mode.
   For details about offline mode, see 10.2.3 HADB server operation modes.
2. The comment, chunk ID, and status of the current chunk are obtained.
3. The following `adbdbstatus` command is executed, and the chunk IDs of the chunks that need to be reorganized are obtained.

```
adbdbstatus -d reorginfo -n processing-target-column-store-table -t
```

4. The following steps (a to d) are repeated for each of the chunks other than the current chunk, among the chunks determined in step 3.
   a. The comments and statuses of the reorganization-target chunks are obtained.
   b. The data of the reorganization-target chunks is exported.
   c. The data of the reorganization-target chunks is deleted.
   d. The data exported in step b is imported. At this time, import is performed according to the chunk comment and chunk status obtained in step a.
5. If the current chunk obtained in step 2 is subject to reorganization, steps a to d of step 4 are executed for the current chunk. If the current chunk obtained in step 2 is not subject to reorganization, an empty chunk is created, which is then merged with the current chunk.
6. The `adbchgsrvmode --normal` command is executed, and the HADB server operation mode changes to normal mode.

11.4.17 Archiving chunks (when using an archivable multi-chunk table)

To archive chunks in an archivable multi-chunk table (to place chunks in archived state), execute the `adbarchivechunk` command. The following shows the procedure.

Procedure:

1. Identify the chunks that you want to place in archived state.

   Identify the chunks that you want to place in archived state by using the value in the archive range column. If you execute the `adbarchivechunk` command with the `-r` and `-t` options specified, you can identify the IDs of chunks that are included in the value range in the archive range column.

   Specification example

   This example identifies the chunks whose date information in the archive range column includes a date between January 1, 2015 and January 31, 2015.

```
adbarchivechunk -u ADBUSER01 -p '#HelloHADB_01'
                -r '2015/01/01-2015/01/31'
                -t
                SALESLIST
```

   If you execute the `adbarchivechunk` command with the `-r` and `-t` options specified, the `KFAA80245-I` message is output. In the `KFAA80245-I` message, you can identify the IDs of the chunks that are included in the date range specified for the `-r` option.

2. Place the chunks in archived state.

   Execute the `adbarchivechunk` command to place the chunks in archived state. When executing the preceding command, specify the chunk IDs determined in step 1 for the `-c` option.

   Specification example

   This example places the chunk whose chunk ID is 5 in archived state.

```
adbarchivechunk -u ADBUSER01 -p '#HelloHADB_01'
                -z /home/adbmanager/archive_file/archive_opt_file.txt
                -c 5
                SALESLIST
```

> **Note**
>
> When you execute the adbarchivechunk command, you can specify the -r option, instead of the -c option. In this case, specify the value range in the archive range column for this option to specify the chunks to be placed in archived state.

To check the compression ratio of an archived chunk and the size of archive files, execute the adbdbstatus command with the -c archivechunk and -n options specified to output the archived chunk summary information. For the -n option, specify the name of the archivable multi-chunk table in which archived chunks are contained. Then, check the following information:

- Archive_file_size (Total size of all archive files corresponding to an archived chunk)
- Compression_ratio (Compression ratio of an archived chunk)

> **Note**
>
> For details about the adbarchivechunk command, see *adbarchivechunk (Archive Chunk)* in the manual *HADB Command Reference*.
>
> For details about the adbdbstatus command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

11.4.18 Unarchiving chunks (when using an archivable multi-chunk table)

To unarchive chunks in an archivable multi-chunk table (to release chunks from archived state), execute the adbunarchivechunk command. The following shows the procedure.

Procedure:

1. Identify the chunks that you want to release from archived state.

   Identify the chunks that you want to release from archived state by using the value in the archive range column. If you execute the adbunarchivechunk command with the -r and -t options specified, you can identify the IDs of chunks that are included in the value range in the archive range column.

   Specification example

   This example identifies the chunks whose date information in the archive range column includes a date between January 1, 2015 and January 31, 2015.

   ```
   adbunarchivechunk -u ADBUSER01 -p '#HelloHADB_01'
                     -r '2015/01/01-2015/01/31'
                     -t
                     SALESLIST
   ```

   If you execute the adbunarchivechunk command with the -r and -t options specified, KFAA80245-I message is output. In the KFAA80245-I message, you can determine the IDs of the chunks that are included in the date range specified for the -r option.

2. Check the data size after the archived state is released.

   If you release chunks from archived state, the data size increases. Make sure that there is no problem if the data size increases. To check the data size of chunks after they are released from archived state, execute the adbdbstatus command with the -c archivechunk and -n options specified to output the archived chunk summary

information. For the -n option, specify the name of the archivable multi-chunk table in which the chunks to be released from archived state are contained.

In the archived chunk summary information, check the following information about the chunks to be released from archived state:

- Unarchive_table_size (Size of the segments used for storing tables in an unarchived chunk)
- Unarchive_index_size (Size of the segments used for storing indexes in an unarchived chunk)

The sum of the preceding values becomes the guideline for the data size (for table data and index data) of the chunks that have been released from archived state.

3. Release chunks from archived state.

   If there is no problem in the data size after the archived state is released, execute the adbunarchivechunk command to release the chunks from archived state. When executing the preceding command, specify for the -c option the chunk IDs identified in step 1.

   Specification example

   This example releases the chunk whose chunk ID is 5 from archived state.

```
adbunarchivechunk -u ADBUSER01 -p '#HelloHADB_01'
                  -w /home/adbmanager/tmp
                  -z /home/adbmanager/unarchive_file/unarchive_opt_file.txt
                  -c 5
                  SALESLIST
```

> **Note**
>
> When you execute the adbunarchivechunk command, you can specify the -r option, instead of the -c option. In this case, specify the value range in the archive range column for this option to specify the chunks to be released from archived state.

To check whether the chunks in the specified range have been released from archived state, execute the adbdbstatus command with the -c archivechunk and -n options specified to output the archived chunk summary information. For the -n option, specify the name of the archivable multi-chunk table in which the chunks released from archived state are contained. Then, check the following information:

- Range_min (Minimum value for the archive range column)
- Range_max (Maximum value for the archive range column)

To check the data size of chunks that have been released from archived state, execute the adbdbstatus command with the -d used, -c table, and -n options specified to output the usage information. For the -n option, specify the name of the archivable multi-chunk table in which the chunks released from archived state are contained. Then, check the following information:

- Used_segments (Number of segments being used)
- Used_pages (Number of pages being used)

> **Note**
>
> - If you execute the adbunarchivechunk command, the HADB server creates a temporary chunk, which is used by the system to release chunks from archived state. Therefore, even if you perform background import or merge of chunks after executing the adbunarchivechunk command, the

ID of the created chunk will not be the number increased by one from the ID of the chunk previously created.

- For details about the `adbunarchivechunk` command, see *adbunarchivechunk (Unarchive Chunk)* in the manual *HADB Command Reference*.
- For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

11.4.19 Checking archived chunks

You can use the `adbdbstatus` command to check the archived chunks. The following shows the procedure.

Procedure:

1. Execute the `adbdbstatus` command.

   Execute the `adbdbstatus` command to output the archived chunk summary information.

   Specification example

   ```
   adbdbstatus -c archivechunk -n ADBUSER01.SALESLIST
   ```

2. Check the execution result of the `adbdbstatus` command.

   Check the value for `Archive_status` in the execution result of the `adbdbstatus` command.

   The chunks for which `Archived` is displayed for `Archive_status` are archived chunks. The chunks for which nothing is displayed for `Archive_status` are unarchived chunks.

Note

For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

11.4.20 Changing a regular multi-chunk table to an archivable multi-chunk table

To change a regular multi-chunk table to an archivable multi-chunk table, execute the `ALTER TABLE` statement. The following shows the procedure.

Important

You cannot change a regular multi-chunk table to an archivable multi-chunk table if the regular multi-chunk table is a column store table.

Procedure:

1. Check whether there are viewed tables that are to be invalidated.

   If you change a regular multi-chunk table to an archivable multi-chunk table, the viewed tables that depend on the target table are invalidated. Therefore, before you change a regular multi-chunk table to an archivable multi-chunk

table, check whether there are viewed tables that are to be invalidated. To check viewed tables that are to be invalidated (dependent viewed tables), see 11.2.11 Checking dependent viewed tables.

If there is a viewed table that is to be invalidated, after changing a regular multi-chunk table to an archivable multi-chunk table, you need to release the viewed table from invalidation.

2. Check the number of chunks that can be created in a data DB area.

   Use the `adbdbstatus` command to output the DB area summary information of the data DB area in which the regular multi-chunk table is stored. Then, confirm that the value output to the following item is no less than 10:

   - `Creatable_Chunks` (Number of chunks that can be created in a DB area)

   If the value is less than 10, you cannot change the regular multi-chunk table to an archivable multi-chunk table. When the table is changed to an archivable multi-chunk table, the number of chunks is consumed by the location table and the indexes for the location table that are automatically defined by the HADB server.

   > **Note**
   >
   > For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

3. Consider and determine the column that you specify as the archive range column.

   From the columns that compose the regular multi-chunk table, consider and determine a column that you specify as the archive range column. Consider which column to specify as the archive range column based on the explanation in (5) Specification of the archive range column under 5.2.5 Points to consider in defining an archivable multi-chunk table [Row store table].

4. Check whether a range index has been defined for the column that you specify as the archive range column.

   After you determined the column that you specify as the archive range column in step 3, check whether a range index has been defined for the column that you specify as the archive range column. To determine whether a range index is defined, see (27) Finding out the range indexes defined for the archive range column in an archivable multi-chunk table in B.22 Searching a dictionary table.

5. Change the regular multi-chunk table to an archivable multi-chunk table.

   Execute the `ALTER TABLE` statement to change the regular multi-chunk table to an archivable multi-chunk table. Note that how to specify the `ALTER TABLE` statement differs depending on the confirmation result in step 4.

   - When a range index has been defined for the column that you specify as the archive range column

     Do not specify `IN` *DB-area-name* for the chunk-archive specification of the `ALTER TABLE` statement.

     After execution of the `ALTER TABLE` statement is completed, go to step 7.

   - When no range index has been defined for the column that you specify as the archive range column

     Specify `IN` *DB-area-name* for the chunk-archive specification of the `ALTER TABLE` statement. The range index automatically defined by the HADB server is stored in the DB area specified for `IN` *DB-area-name*. The range index is defined for the column that you specify as the archive range column.

     After execution of the `ALTER TABLE` statement is completed, go to step 6.

   > **Note**
   >
   > For details about the `ALTER TABLE` statement, see *ALTER TABLE (alter table definition)* in the manual *HADB SQL Reference*.

6. Re-create the range index automatically defined by the HADB server.

If you specified IN *DB-area-name* for the chunk-archive specification of the ALTER TABLE statement in step 5, execute the adbidxrebuild command for the applicable archivable multi-chunk table. Execute the adbidxrebuild command to re-create the range index automatically defined by the HADB server.

After execution of the adbidxrebuild command is completed, go to step 7.

7. Re-validate viewed tables.

   If there are viewed tables that depend on the target table, release the viewed tables from invalidation. For details about how to release a viewed table from invalid status, see (1) When viewed tables are invalidated by using an ALTER TABLE statement to change the table type in 11.2.8 Releasing a viewed table from invalidation.

Now, change of the regular multi-chunk table to an archivable multi-chunk table is completed.

11.4.21 Changing an archivable multi-chunk table to a regular multi-chunk table

To change an archivable multi-chunk table to a regular multi-chunk table, execute the ALTER TABLE statement.

> **Important**
>
> If archived chunks exist in an archivable multi-chunk table, you cannot change the archivable multi-chunk table to a regular multi-chunk table. You need to release the chunks from archived state.
>
> If archived deletion-pending chunks exist in the archivable multi-chunk table, you cannot release the chunks from archived state by using the adbunarchivechunk command. You need to execute the PURGE CHUNK statement to delete the relevant chunks.

Procedure:

1. Check whether there are viewed tables that are to be invalidated.

   If you change an archivable multi-chunk table to a regular multi-chunk table, the viewed tables that depend on the target table are invalidated. Therefore, before you change an archivable multi-chunk table to a regular multi-chunk table, check whether there are viewed tables that are to be invalidated. To check viewed tables that are to be invalidated (dependent viewed tables), see 11.2.11 Checking dependent viewed tables.

   If there are viewed tables that are to be invalidated, after changing an archivable multi-chunk table to a regular multi-chunk table, you need to release the viewed tables from invalidation.

2. Check whether there are archived chunks and archived, deletion-pending chunks.

   Execute the adbdbstatus command to output the table summary information of the archivable multi-chunk table. Then, check the following items (starting with the value for Archive_chunks).

   - Archive_chunks (Number of archived chunks)

     If the value for Archive_chunks is 0, go to step 3. If the value is 1 or larger, one or more archived chunks exist. Execute the adbunarchivechunk command to release all chunks from archived state.

     Then, re-execute the adbdbstatus command, and confirm that the value for Archive_chunks is 0. If the value is not 0, check the following Pending_delete_chunks:

   - Pending_delete_chunks (Number of deletion-pending chunks in the table)

     If the value for Pending_delete_chunks is 1 or larger and the value for Archive_chunks is 1 or larger, one or more archived, deletion-pending chunks exist. In this case, execute the PURGE CHUNK statement to delete all archived, deletion-pending chunks.

If an archived chunk or an archived, deletion-pending chunk exists, you cannot change the archivable multi-chunk table to a regular multi-chunk table.

Note

- For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.
- For details about the `adbunarchivechunk` command, see *adbunarchivechunk (Unarchive Chunk)* in the manual *HADB Command Reference*.
- For details about the `PURGE CHUNK` statement, see *PURGE CHUNK (delete all rows in a chunk)* in the manual *HADB Command Reference*.

3. Change an archivable multi-chunk table to a regular multi-chunk table.

   Execute the `ALTER TABLE` statement to change the archivable multi-chunk table to a regular multi-chunk table.

   Note

   For details about the `ALTER TABLE` statement, see *ALTER TABLE (alter table definition)* in the manual *HADB SQL Reference*.

4. Release viewed tables from invalidation.

   If there are viewed tables that depend on the target table, release the viewed tables from invalidation. For details about how to release a viewed table from invalid status, see (1) When viewed tables are invalidated by using an ALTER TABLE statement to change the table type in 11.2.8 Releasing a viewed table from invalidation.

Now, change of the archivable multi-chunk table to a regular multi-chunk table is completed.

Note that the range index automatically defined for the archive range column of the archivable multi-chunk table cannot be deleted after the table is changed to a regular multi-chunk table. If you do not need the range index, execute the `DROP INDEX` statement to delete the index. To delete a range index, see 11.3.9 Deleting an index.

Important

Do not execute the `ALTER TABLE` statement when the `adbunarchivechunk` command has been interrupted. If you execute the `ALTER TABLE` statement, you need to change the regular multi-chunk table back to an archivable multi-chunk table. Then, you need to release the state of the `adbunarchivechunk` command being interrupted. The following shows the procedure.

1. Execute the `ALTER TABLE` statement to change the regular multi-chunk table back to an archivable multi-chunk table.
2. For the archivable multi-chunk table you changed back in step 1, execute the `adbunarchivechunk` command with the `--force` option specified. You can release the state of the `adbunarchivechunk` command being interrupted. If the `KFAA50284-E` message is output and the `adbunarchivechunk` command results in an error, the state of the `adbunarchivechunk` command being interrupted is released.
3. Execute the `ALTER TABLE` statement to change the archivable multi-chunk table you changed back in step 1 to a regular multi-chunk table.

11.4.22 Deleting a multi-chunk table

To delete a multi-chunk table, execute the `DROP TABLE` statement. For details about the `DROP TABLE` statement, see *DROP TABLE (delete a table)* in *Definition SQL* in the manual *HADB SQL Reference*.

Note that, if you delete an archivable multi-chunk table, the following are also deleted:

- Archive files
- Range index defined for the archive range column
- Location table
- Indexes defined for the location table

11.4.23 Operation taking background import and chunks into consideration (Example 1: Adding and deleting data on a regular basis)

This subsection provides an operation example that takes into consideration background import and chunks. When you use background import, you can repeat data addition and deletion on a regular basis, as described below.

In the following description, the operation of Company A is used as an example.

■ Operation example of Company A

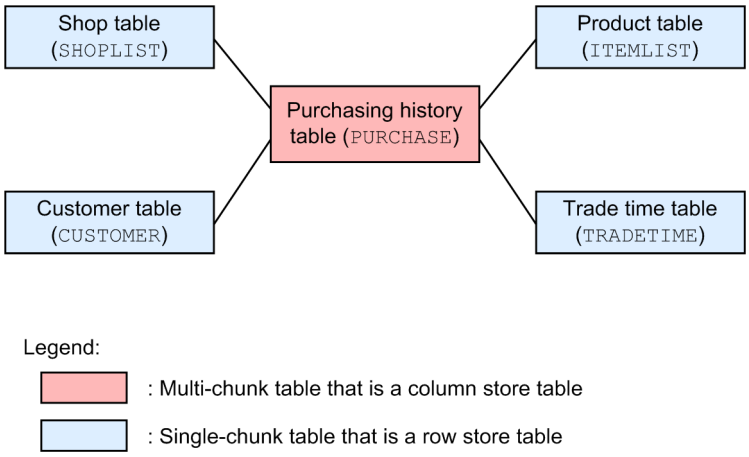
Company A, which has about 1,000 branch stores across the country, has adopted an analysis system for analyzing product purchasing histories. The database of Company A's analysis system is managed in the following manner:

- The data of purchase histories of about 1,000 branch stores across the country is centrally managed by using a purchasing history table in the database of the analysis system.
- The database of the analysis system also contains the shop table and product table that are used to manage the master data other than the purchasing history data, in addition to the purchasing history table.
- Purchasing history data is collected from all branch stores once a day, and is collectively imported into the purchasing history table.
- The system analyzes the product purchasing history data while importing the data into the purchasing history table.
- Product purchasing history analysis can target a period of six months, three months, one month, one week, and one day.
- To prevent degradation of the system's retrieval performance, every month, the purchasing history data chunks of the immediately preceding month are merged into a single chunk (to suppress increases in the number of chunks).
- The analysis processing of Company A requires the purchasing history data of the past six months. Therefore, the purchasing history table, which stores purchasing history data, stores the data collected in the past six months.
- The data older than the data collected in the last six months is backed up and then deleted to suppress increase of the data size of the purchasing history table.
- The purchasing history table data is backed up every month.
- If the purchasing history data has incorrect information, correct it.

■ Database schema model of the analysis system

The following figure shows the database schema model of the analysis system used by Company A.

Figure 11-25: Database schema model of Company A's analysis system



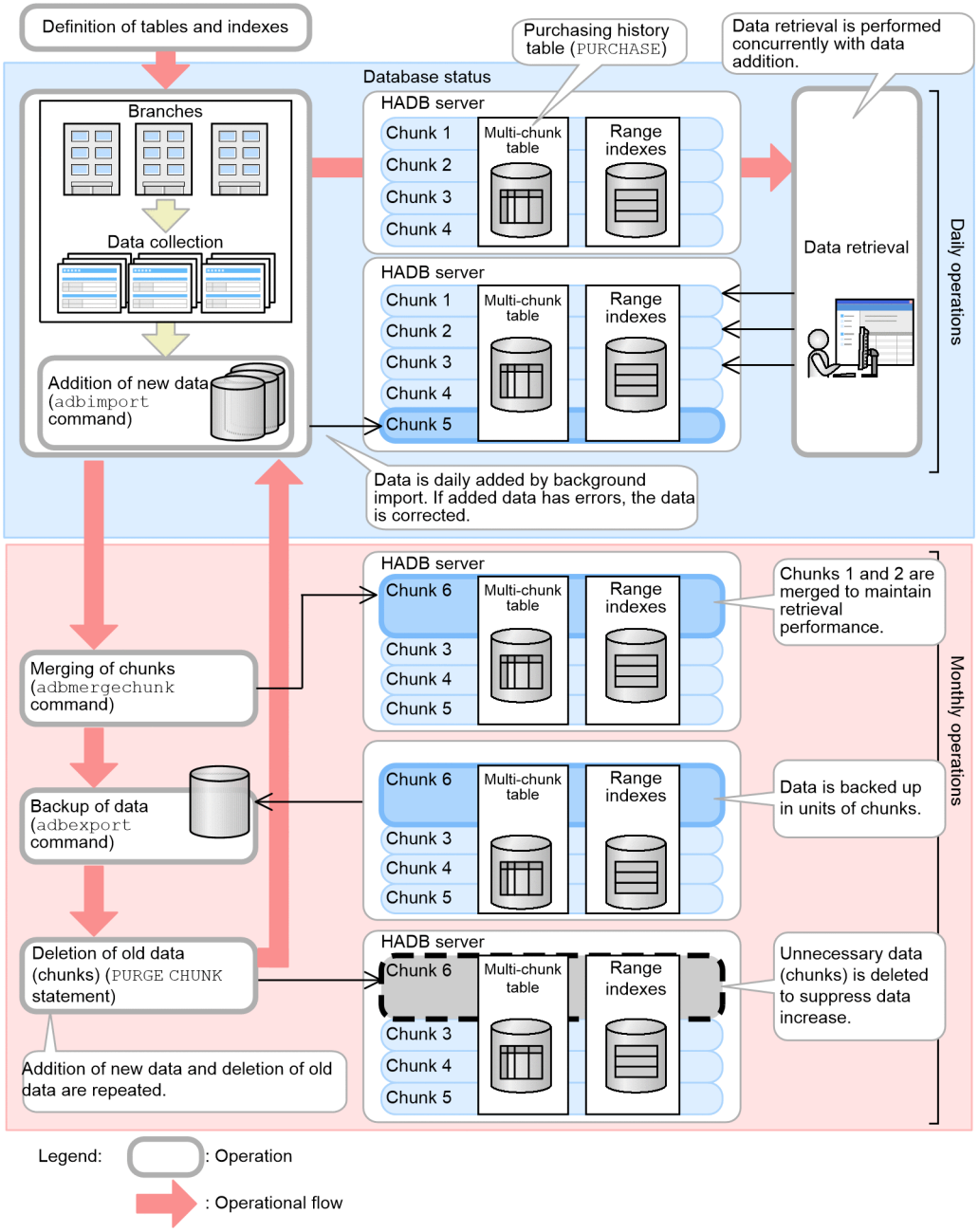
Explanation:

- In the preceding star schema, the purchasing history table (PURCHASE) functions as a fact table. The purchasing history table imports data every day, and is used for retrieval during analysis even while data is being imported. To enable simultaneous execution of data import and retrieval, the purchasing history table is defined as a multi-chunk table. In addition, the purchasing history table is also defined as a column store table because value summation for a specific field frequently occurs (for example, to obtain the average or total) during analysis. (3) Adding new data (background import) and the following subsections describe operations on the purchasing history table.
- To manage the master data other than the purchasing history data, the shop table (SHOPLIST), product table (ITEMLIST), customer table (CUSTOMER), and trade time table (TRADETIME) are defined. These tables function as dimension tables in the star schema. These tables are defined as a single-chunk table that is a row store table.

■ Overview of operations related to the purchasing history table (PURCHASE)

The following figure shows an overview of operations related to the purchasing history table.

Figure 11-26: Overview of operations related to the purchasing history table



(1) Enabling the updated-row columnizing facility

Because the UPDATE statement is sometimes used to update the data of the purchasing history table (PURCHASE), execute the adbcolumnize command to enable the updated-row columnizing facility.

■ Command execution example

```
adbcolumnize --start
```

Note

If the UPDATE or INSERT statement is executed for a column store table, the updated or added data is stored in row store format. This might degrade the retrieval performance for the column store table. If the

updated-row columnizing facility is enabled, HADB automatically converts the data from row store format to column store format, thus preventing degradation of retrieval performance for column store tables.

(2) Defining the database of the analysis system

Define the purchasing history table (PURCHASE) to centrally manage the purchasing history data of all branch stores. Also, define range indexes for the purchasing history table.

In addition, define the following four tables to manage the master data other than the purchasing history data:

- Shop table (SHOPLIST)
- Product table (ITEMLIST)
- Customer table (CUSTOMER)
- Trade time table (TRADETIME)

(a) Defining the purchasing history table (PURCHASE)

Define the purchasing history table (PURCHASE) as follows:

- Define the purchasing history table as a multi-chunk table because the table will be subject to background import.
- Define the purchasing history table as a column store table because the table will be used as a fact table of a star schema.
- Store the purchasing history table in DB area TBLDAT_001.
- Set the maximum number of chunks in the purchasing history table to 44. The following shows how this value was estimated.

■ Estimating the maximum number of chunks in the purchasing history table

The purchasing history table will store the data of the past six months. This is the basis for estimating the maximum number of chunks to be specified in the chunk specification of the CREATE TABLE statement.

Assuming 31 days in a month, background import is performed 31 times a month, and therefore the number of chunks needed is 31. For the remaining five months, the number of chunks is kept low by grouping data into a single chunk every month in order to maintain a high level of retrieval performance.

Assuming a safety factor of 1.2, the maximum number of chunks that can be created is as follows:

```
maximum-number-of-chunks-to-be-created = ↑(31 + 5) × 1.2 (safety factor)↑ = 44
```

> **Note**
>
> If the number of chunks increases, this negatively impacts retrieval performance. Therefore, we recommend that you design and operate your database system to minimize the number of chunks that will be used.

Based on this estimate, specify 44 as the maximum number of chunks. Specify this value for CHUNK of the chunk specification of the CREATE TABLE statement. The following shows an example of specifying the CREATE TABLE statement to define the purchasing history table.

■ **Example of specifying the `CREATE TABLE` statement**

```
CREATE TABLE "PURCHASE"
            ("SHOP_ID"     INTEGER,
             "TRADE_TIME"  TIMESTAMP,
             "ITEM_ID"     INTEGER,
             "ITEM_NAME"   VARCHAR(100),
             "CUSTOMER_ID" INTEGER,
             "PURCHASE_ID" CHAR(8))
    IN "TBLDAT_001"
    CHUNK=44                                ...1
    STORAGE FORMAT COLUMN                   ...2
```

Explanation:

1. This specification sets the maximum number of chunks. This specification defines the purchasing history table as a multi-chunk table.
2. This specification defines the purchasing history table as a column store table.

For details about the `CREATE TABLE` statement, see *CREATE TABLE (define a table)* in *Definition SQL* in the manual *HADB SQL Reference*.

(b) Defining range indexes for the purchasing history table (PURCHASE)

Define range indexes for the purchasing history table (`PURCHASE`). Define the range indexes so that they are stored in DB area `RIXDAT_001`.

Define range indexes for the trade time (`TRADE_TIME` column). The following shows an example of specifying the `CREATE INDEX` statement to define range indexes.

■ **Example of specifying the `CREATE INDEX` statement**

```
CREATE INDEX "TRADE_TIME_RIX"
   ON "PURCHASE"("TRADE_TIME") IN "RIXDAT_001"
      EMPTY INDEXTYPE RANGE
```

For details about the `CREATE INDEX` statement, see *CREATE INDEX (define an index)* in *Definition SQL* in the manual *HADB SQL Reference*.

(c) Defining tables other than the purchasing history table (PURCHASE)

Define also the following tables, which are required to manage the master data other than the purchasing history data:

- Shop table (`SHOPLIST`)
- Product table (`ITEMLIST`)
- Customer table (`CUSTOMER`)
- Trade time table (`TRADETIME`)

Define the preceding tables as follows:

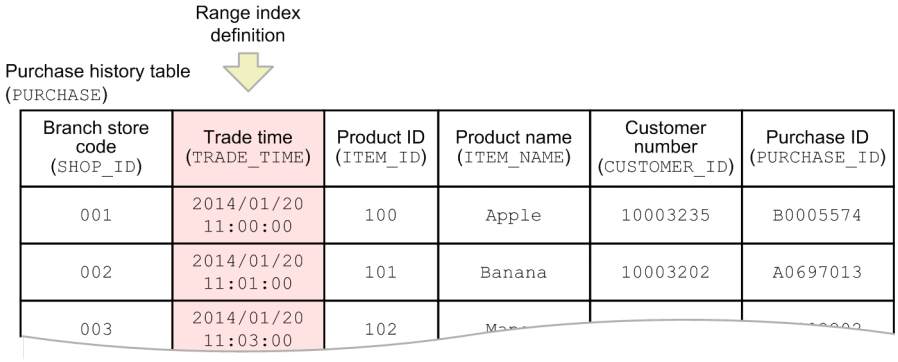
- Define the preceding tables as single-chunk tables because they will not be subject to background import.
- Define the preceding tables as row store tables because they will be used as dimension tables of a star schema.
- Define the preceding tables so that they are stored in DB areas `TBLDAT_002` to `TBLDAT_005`, respectively.

- B-tree indexes must be defined for some columns of the preceding tables. The following lists the relevant columns. Define the B-tree indexes so that they are stored in DB areas IDXDAT_002 to IDXDAT_005, respectively.
  - Branch store code (SHOP_ID column) of the shop table (SHOPLIST)
  - Product ID (ITEM_ID column) of the product table (ITEMLIST)
  - Customer number (CUSTOMER_ID column) of the customer table (CUSTOMER)
  - Trade time (TRADE_TIME column) of the trade time table (TRADETIME)

(d) Table used in this operation example

The following figure shows the purchasing history table (PURCHASE) used in this operation example.

Figure 11-27: Purchasing history table (PURCHASE) used in this operation example



Branch store code (SHOP_ID)	Trade time (TRADE_TIME)	Product ID (ITEM_ID)	Product name (ITEM_NAME)	Customer number (CUSTOMER_ID)	Purchase ID (PURCHASE_ID)
001	2014/01/20 11:00:00	100	Apple	10003235	B0005574
002	2014/01/20 11:01:00	101	Banana	10003202	A0697013
003	2014/01/20 11:03:00	102			

(3) Adding new data (background import)

Once a day, new purchasing history data collected from branch stores all over the country is added to the purchasing history table (PURCHASE).

New data is added using background import. The group of data that is stored in the multi-chunk table in a single background import operation is managed together with its indexes as a group (chunk). For details about a chunk, see 2.14.2 Managing data in data-import units (chunks).

> **Note**
>
> By managing data in units of chunks, you can acquire backups and delete data on a chunk-by-chunk basis. When you perform background import or merging chunks, you can add a comment to a chunk. Adding a comment to a chunk makes it easier to identify a certain chunk you want to manage.

To perform background import, use the adbimport command with the -b option specified. The following shows an example of executing the adbimport command.

■ Command execution example

```
adbimport -u ADBUSER01 -p '#HelloHADB_01' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file01.txt
          -b -m 'January 2014_daily'
          PURCHASE
          /home/adbmanager/imp_file/imp_data_path01.txt
```

Explanation:

Use background import to import the contents of the input data file, set up in the input data path file (`/home/adbmanager/imp_file/imp_data_path01.txt`), into the purchasing history table (`PURCHASE`).

For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*. Also, see 11.4.2 Storing data in a multi-chunk table (background import).

(4) Correcting the data imported by background import

If you find errors in the data that was imported by background import in (3) Adding new data (background import) and the amount of data to be corrected is small, use the `UPDATE` or `DELETE` statement to correct the data. The following shows an example of correcting one record by using the `UPDATE` statement.

■ **Example of executing an SQL statement**

```
UPDATE "PURCHASE" SET "ITEM_ID"=103 WHERE "PURCHASE_ID"='A0015309'
```

Explanation:

The preceding SQL statement changes the product ID (`ITEM_ID`) to `103` for the record whose purchase ID (`PURCHASE_ID`) is `A0015309` in the purchase history table (`PURCHASE`).

For details about the `UPDATE` statement, see *UPDATE (update rows)* in the manual *HADB SQL Reference*.

If the amount of data to be corrected is large, we recommend that you correct the contents of the input data file and execute the `adbimport` command to replace the current data with the corrected data. For details about how to replace the current data with the corrected data, see (4) Operational example (replacing the data in a chunk) in 11.4.12 Changing the chunk status.

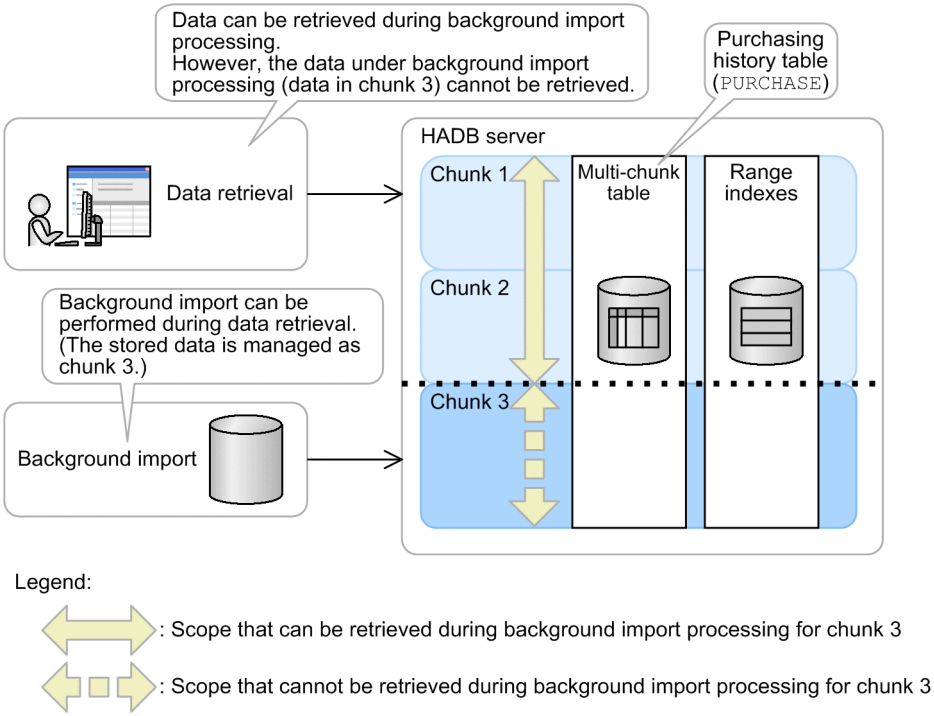
(5) Retrieving data

You can also retrieve data from the purchasing history table (`PURCHASE`) while data is being added to the table by background import.

By using background import to add data, you can import data even if another user is retrieving data from the same database (data can be imported during data retrieval).

The following figure shows the relationship between background import and data retrieval.

Figure 11-28: Relationship between background import and data retrieval

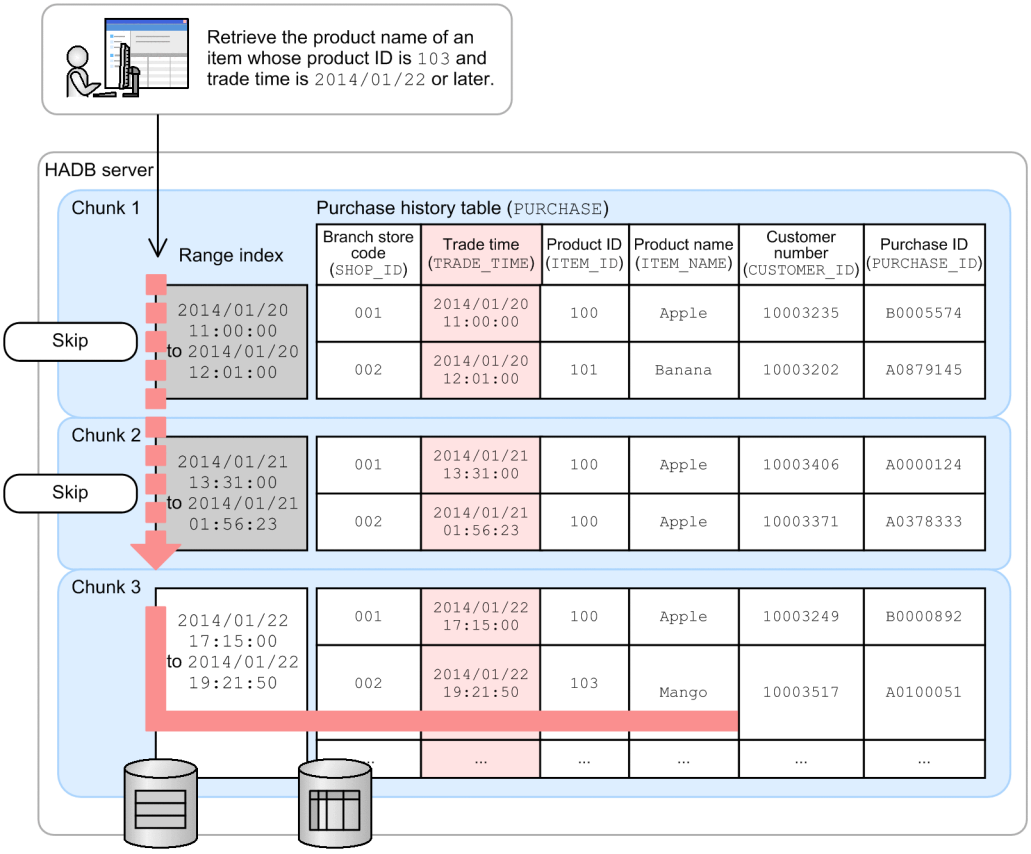


The purchasing history data stored in the purchasing history table is analyzed for six-month, three-month, one-month, one-week, and one-day periods.

If range indexes are defined for the purchasing history table, which is a multi-chunk table, each range index holds the maximum and minimum values (a range) of the values existing in a key field within the chunk. Since chunks that store data not included in the search condition are skipped during retrieval, retrieval efficiency improves.

The following figure shows the processing that takes place when a range index is defined for the purchasing history table.

Figure 11-29: Processing that takes place when a range index is defined for the purchasing history table



Explanation:

In this case, a range index is defined for the trade time (TRADE_TIME column). Therefore, during retrieval using the trade time, retrieval of chunks that do not store trade time data not included in the search condition is skipped.

(6) Merging chunks

By merging the data chunks of a month into one chunk in the purchasing history table (PURCHASE), you can prevent retrieval performance degradation that tends to result from an increase in the number of chunks. In the example shown later, one month's worth of data chunks (data chunks for January 2014) are merged into a single chunk.

(a) Identifying the chunks to be merged

To determine the chunk IDs of the chunks to be merged, retrieve the system table STATUS_CHUNKS as explained below.

The following shows an SQL statement example.

■ SQL statement example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
   WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='PURCHASE'
      AND "CHUNK_COMMENT"='January 2014_daily'
```

■ Retrieval result

```
CHUNK_ID
--------------------
```

```
                  1
           (omitted)
                 31
```

Explanation:

In this example, the chunk IDs (1 to 31) of the chunks to be merged are identified by using the chunk comment ('January 2014_daily'), added during background import, as the key during retrieval of the STATUS_CHUNKS table.

Check the chunks to be merged and their chunk IDs based on the explanation in 11.4.8 Checking the chunk status and the number of chunks created.

(b) Checking whether chunk reorganization is necessary

For the merge-target chunks identified in (a) Identifying the chunks to be merged, you can check whether reorganization of the chunks is necessary. To do this, use the adbdbstatus command to obtain *information about the need for reorganization*.

The following shows an example of executing the command.

■ Command execution example

```
adbdbstatus -d reorginfo -n ADBUSER01.PURCHASE -c 1-31
```

Explanation:

In this example, the command obtains the information about the need for reorganization in terms of the chunk IDs of the merge-target chunks (1 to 31).

The information about the need for reorganization (Reorganization_necessity) is output for each chunk. If Reorganization_necessity is Recommended, perform reorganization of the chunk. For details about the procedure for reorganizing chunks, see (c) Reorganizing chunks.

After reorganizing chunks, merge them into a single chunk as described in (d) Merging chunks.

(c) Reorganizing chunks

As a result of the check in (b) Checking whether chunk reorganization is necessary, if you find that there are chunks for which the need for reorganization (Reorganization_necessity) is Recommended, reorganize them. This subsection describes the chunk reorganization procedure in an example case where the chunk with the chunk ID 2 needs to be reorganized.

■ Procedure for reorganizing chunks

1. Stop the commands, jobs, and application programs that update the purchase history table (PURCHASE).
2. Search the system table STATUS_CHUNK, and check the status and comment of the chunk with the chunk ID 2.

   ■ Example of executing an SQL statement

   ```
   SELECT CHUNK_ID, CHUNK_COMMENT, CHUNK_STATUS
     FROM "MASTER"."STATUS_CHUNKS"
       WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='PURCHASE'
         AND "CHUNK_ID"=2
   ```

   ■ Execution result of the SQL statement

   ```
   CHUNK_ID    CHUNK_COMMENT CHUNK_STATUS
   ----------- ------------- ------------------------
             2 daily         Normal
   ```

3. Use the adbexport command to output the data of the chunk with the chunk ID 2.

   Execute the command twice: once by specifying the export option file (/home/adbmanager/exp_file/exp_opt_file01.txt), and once by specifying the output data path file (/home/adbmanager/exp_file/exp_filepath.txt).

   ■ Command execution example

```
adbexport -u ADBUSER01 -p '#HelloHADB_01'
          -z /home/adbmanager/exp_file/exp_opt_file01.txt
          -n PURCHASE
          -c 2
          /home/adbmanager/exp_file/exp_filepath.txt
```

   ■ Content of the output data path file (/home/adbmanager/exp_file/exp_filepath.txt)

```
/home/adbmanager/exp_file/purchase_data0.csv
/home/adbmanager/exp_file/purchase_data1.csv
/home/adbmanager/exp_file/purchase_data2.csv
/home/adbmanager/exp_file/purchase_data3.csv
/home/adbmanager/exp_file/purchase_data4.csv
/home/adbmanager/exp_file/purchase_data5.csv
/home/adbmanager/exp_file/purchase_data6.csv
/home/adbmanager/exp_file/purchase_data7.csv
/home/adbmanager/exp_file/purchase_data8.csv
/home/adbmanager/exp_file/purchase_data9.csv
```

4. Use the adbimport command to import the output data.

   Execute the command twice: once by specifying the import option file (/home/adbmanager/imp_file/imp_opt_file01.txt), and once by specifying the input data path file (/home/adbmanager/exp_file/exp_filepath.txt).

   For the input data path file, use the output data path file, as is, that you used when executing the adbexport command. For the -m option, specify the string 'daily' of the chunk comment that you checked in step 2.

   ■ Command execution example

```
adbimport -u ADBUSER01 -p '#HelloHADB_01' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file01.txt
          -b --status wait
          -m 'daily'
          PURCHASE
          /home/adbmanager/exp_file/exp_filepath.txt
```

5. Use the adbchgchunkstatus command to change the status of the chunk.

   Change the status of the reorganization-target chunk (chunk with the chunk ID 2) and the chunk that you created in step 4 (chunk with the chunk ID 32).

   ■ Command execution example

```
adbchgchunkstatus -u ADBUSER01 -p '#HelloHADB_01'
                  -w 2 -n 32 PURCHASE
```

   When the preceding command is executed, the chunk with the chunk ID 2 is placed in wait status, and the chunk with the chunk ID 32 is placed in normal status.

6. Use the PURGE CHUNK statement to delete the reorganization-target chunk (chunk with the chunk ID 2).

   ■ Example of executing an SQL statement

```
PURGE CHUNK PURCHASE WHERE CHUNKID=2
```

7. Start the commands, jobs, and application programs that update the purchase history table (PURCHASE).

(d) Merging chunks

To merge chunks, use the `adbmergechunk` command. The following shows an execution example of the `adbmergechunk` command.

■ Command execution example

```
adbmergechunk -u ADBUSER01 -p '#HelloHADB_01' -g 2
              -w /home/adbmanager/tmp
              -z /home/adbmanager/merge_file/merge_opt_file01.txt
              -m 'January 2014'
              -c 1-31
              PURCHASE
```

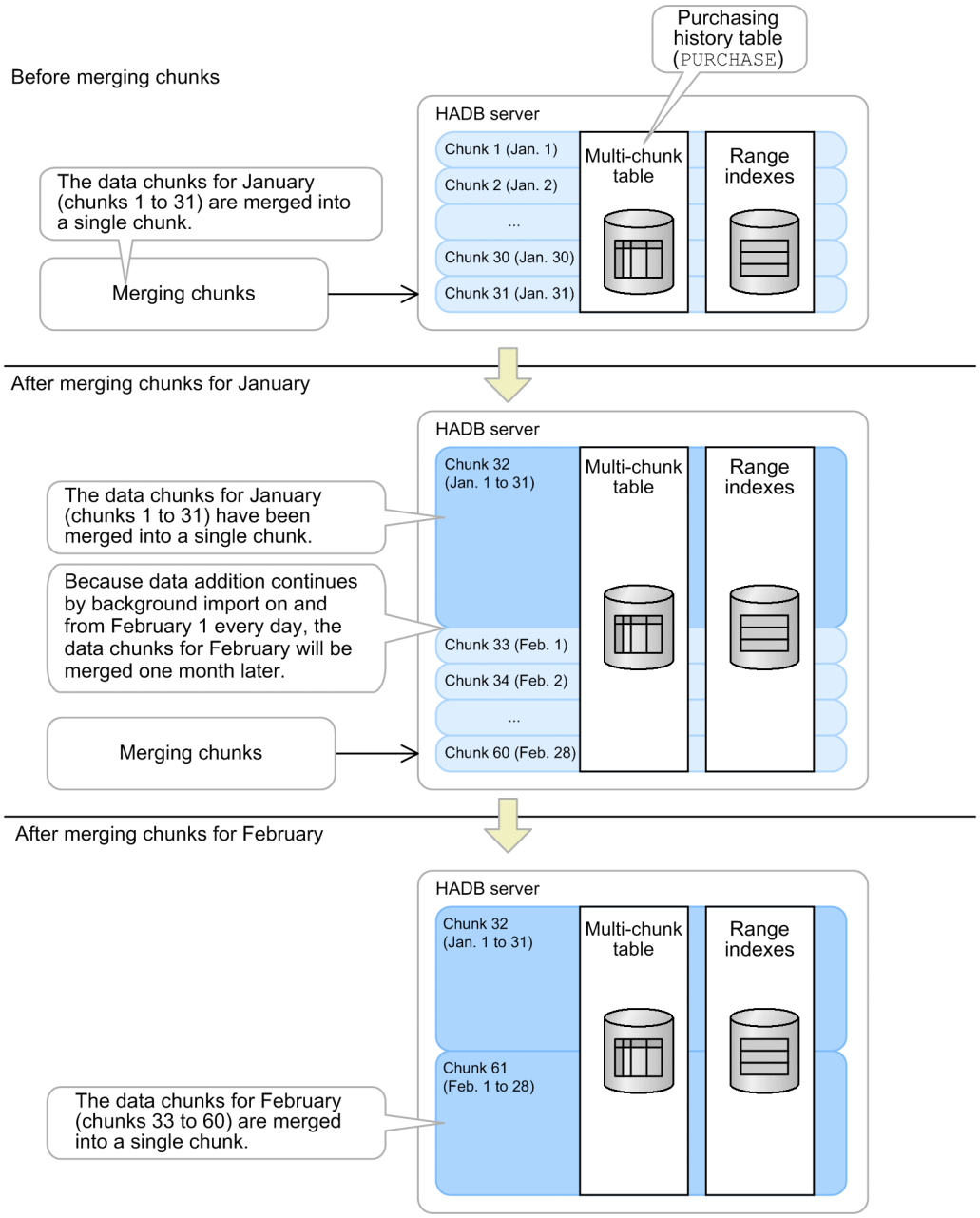
Explanation:

Using the value specified in the merge chunk option file (`/home/adbmanager/merge_file/merge_opt_file01.txt`) as the input information, perform chunk merging on the purchasing history table (`PURCHASE`) as specified in the `-c` option of the `adbmergechunk` command.

In this example, the data for January 2014 is grouped together by merging chunks with chunk IDs `1` through `31` into a single chunk.

The following figure shows an example of chunk merging.

Figure 11-30: Chunk merging example



Chunks with chunk IDs of `1` through `31` are automatically deleted when the `adbmergechunk` command terminates normally.

> **Note**
>
> Merging chunks can improve retrieval performance. There is also a limit to the number of chunks that can be managed by a table or DB area. If the number of chunks is insufficient, you can reduce the number by merging multiple chunks into a single chunk.

You can merge chunks even when another user is retrieving the merge-target chunks.

For details about the `adbmergechunk` command, see *adbmergechunk (Merge Chunks)* in the manual *HADB Command Reference*. Also, see 11.4.9 Merging chunks (to reduce the number of chunks).

(e) Checking whether reorganization of merged chunks is necessary

In this subsection, you merge the chunks of January 2014 into one chunk, and then execute the `adbdbstatus` command to check the information about the need for reorganization (`Reorganization_necessity`) of the chunk. The following shows the procedure.

Procedure

1. Check the chunk ID of the merge-result chunk.

   ■ Example of executing an SQL statement

   ```
   SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
       WHERE "TABLE_SCHEMA"='ADBUSER' AND "TABLE_NAME"='PURCHASE'
         AND "CHUNK_COMMENT"='January 2014'
   ```

   ■ Execution result of the SQL statement

   ```
    CHUNK_ID
    ----------------
                  32
   ```

   Explanation:

   By executing the preceding SQL statement, you can check the chunk ID of the chunk into which the chunks of January 2014 have been merged. In this example, the chunk ID of the merge-result chunk is 32.

2. Check the information about the need for reorganization.

   ■ Command execution example

   ```
   adbdbstatus -d reorginfo -n ADBUSER01.PURCHASE -c 32
   ```

   This command outputs the information about the need for reorganization (`Reorganization_necessity`) of the chunk whose chunk ID is 32. If `Reorganization_necessity` is `Recommended`, perform reorganization of this chunk.

(7) Backing up data

To prevent data size of the purchasing history table (`PURCHASE`) from increasing, the purchasing history table is used to store the data for only the past six months. The data older than six months is backed up and then deleted. A backup is created once a month.

The following describes the detailed procedure.

(a) Identifying the chunks with the earliest creation date and time

Chunks that are older than 6 months (chunks with the earliest creation date and time) are identified in the purchasing history table (`PURCHASE`). To identify these chunks in the purchasing history table, retrieve the system table `STATUS_CHUNKS` as described below.

The following shows an SQL statement example.

■ SQL statement example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
   WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='PURCHASE'
      AND "CREATE_TIME"=(SELECT MIN("CREATE_TIME")
         FROM "MASTER"."STATUS_CHUNKS"
            WHERE TC."TABLE_SCHEMA"='ADBUSER01'
               AND TC."TABLE_NAME"='PURCHASE')
```

■ **Retrieval result**

```
 CHUNK_ID
 --------------------
                   32
KFAA96404-I 1 rows were selected.
```

Explanation:

When the STATUS_CHUNKS table is retrieved, the result shows that the chunk with chunk ID 32 is the chunk with the earliest creation date and time (older than 6 months) in this example.

> **Note**
>
> When chunks are merged, the earliest creation date and time among the merge-source chunks is stored in the STATUS_CHUNKS table.

For details about how to retrieve data from the STATUS_CHUNKS table, see C.9 Searching system tables.

(b) Backup acquisition

Now that the chunk ID of the chunk with the earliest creation date and time in the purchasing history table (PURCHASE) has been identified as 32, back up the data in that chunk in a CSV file.

To make a backup, use the adbexport command. The following shows an execution example of the adbexport command:

■ **Command execution example**

```
adbexport -u ADBUSER01 -p '#HelloHADB_01'
          -z /home/adbmanager/exp_file/exp_opt_file01.txt
          -n PURCHASE
          -c 32
          /home/adbmanager/exp_file/exp_data_path01.txt
```

■ Content of the output data path file (/home/adbmanager/exp_file/exp_data_path01.txt)

```
/home/adbmanager/exp_backup/backup0001.csv
/home/adbmanager/exp_backup/backup0002.csv
/home/adbmanager/exp_backup/backup0003.csv
```

Explanation:

The data contained in the chunk with chunk ID 32 is backed up in CSV files (/home/adbmanager/exp_backup/backup0001.csv, /home/adbmanager/exp_backup/backup0002.csv, and /home/adbmanager/exp_backup/backup0003.csv).

In this example, since chunks are merged every month, one month's worth of data is backed up.

> **Note**
>
> Although you can back up the files into a single file, distributing the files into multiple files can improve performance by distributing the output-related overhead.

For details about the adbexport command, see *adbexport (Export Data)* in the manual *HADB Command Reference*. Also, see 11.4.5 Exporting data in units of chunks.

(8) Deleting old data (chunks)

To prevent the size of the purchasing history table (PURCHASE) from increasing, delete any data that is older than six months.

Delete the chunk with the earliest creation date and time from the purchasing history table (PURCHASE). To delete a chunk, execute the PURGE CHUNK statement with a chunk ID specified in the search condition.

■ SQL statement example

```
PURGE CHUNK "PURCHASE" WHERE CHUNKID=32
```

Explanation:

The data contained in the chunk with chunk ID 32 is deleted.

In this example, since chunks are merged every month, one month's worth of data is deleted.

For details about the PURGE CHUNK statement, see *PURGE CHUNK (delete all rows in a chunk)* in *Data Manipulation SQL* in the manual *HADB SQL Reference*. Also, see 11.4.6 Deleting data in units of chunks.

(9) Repeatedly adding new data, merging, and deleting old data

By repeatedly performing background import, merging chunks, and deleting chunks using the PURGE CHUNK statement, you can continue to analyze purchasing histories without increasing the volume of data managed by HADB.

11.4.24 Operation taking background import and chunks into consideration (Example 2: Using chunks in wait status)

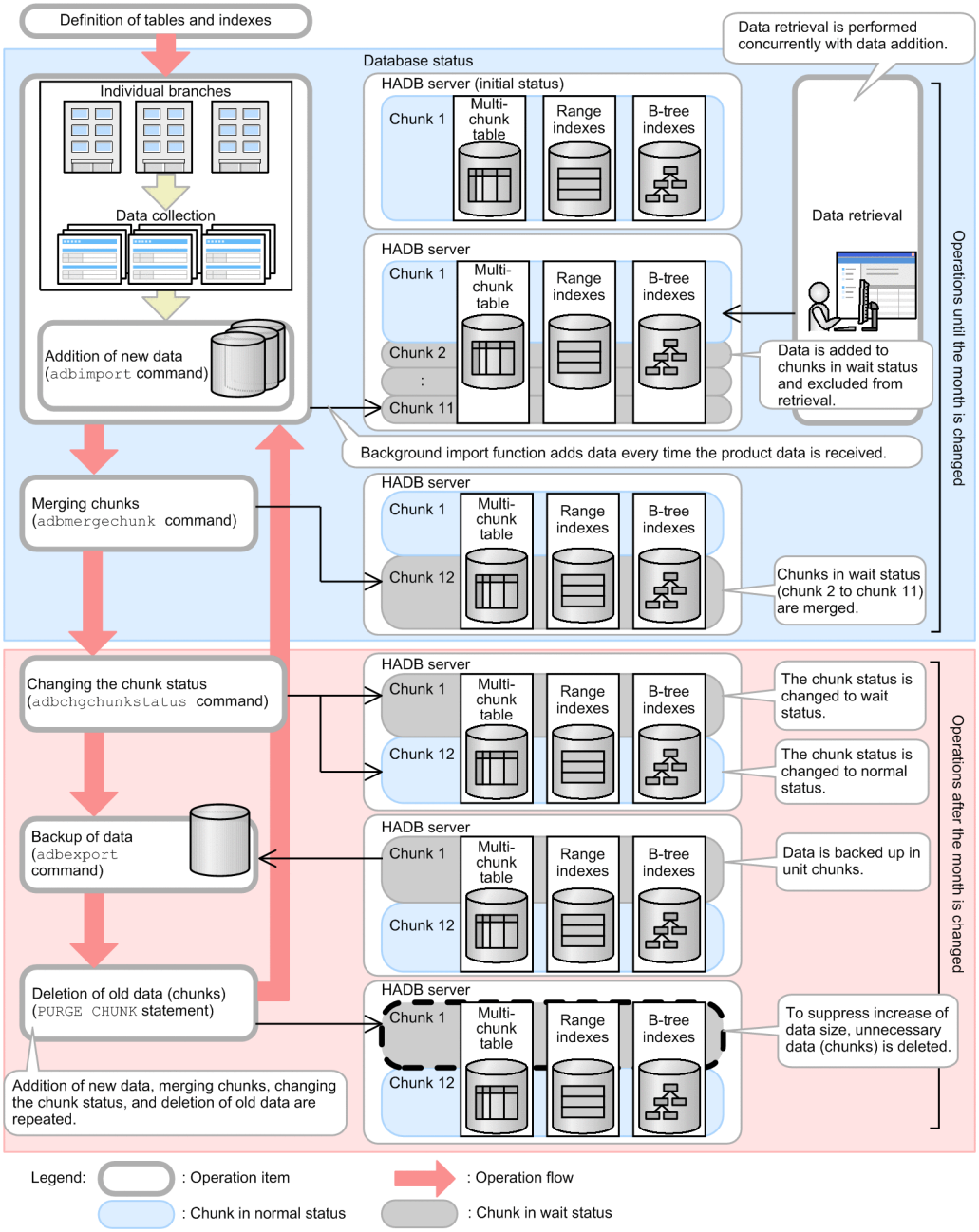
This subsection provides an operation example of background import that uses chunks in wait status. When you use chunks in wait status, you can retrieve data or prevent it from being retrieved at a desired time, as explained below.

Explanations in this example assume that operations are taking place in the following system:

- A main office collects data on products handled by shops nationwide from 10 branch offices, and analyzes the collected product data.
- The product data collected by the main office is managed centrally using a product table.
- Since the product data changes every month, the coming month's product data is sent from each branch office by the 25th of each month.
- The next month's product data is imported into the database as it arrives at the main office.
- The imported product data for the next month needs to be excluded from analysis until the next month begins.
- The product data for the current month is analyzed even as the next month's product data is being imported into the database.
- When the next month begins, the product data that was excluded from analysis becomes subject to analysis. Conversely, the product data that had been subject to analysis is now excluded from analysis.
- The database retains the product data that is now excluded from analysis for 6 months.
- To prevent the amount of data from becoming unmanageable, product data older than six months is backed up and then deleted.

The following figure shows the flow of operations in this system.

Figure 11-31: Flow of operations in this system (usage of chunks in wait status)



(1) Defining a multi-chunk table and indexes

To manage product data centrally, you need to define a product table (ITEMLIST) and indexes.

(a) Defining the product table (ITEMLIST)

Define the product table (ITEMLIST) to be stored in the DB area TBLDAT_002. Also, define the product table as a multi-chunk table to perform background import.

In this operation example, the following number of chunks is required:

- The chunk containing the current month's product data that is to be analyzed (one chunk)
- The chunks containing product data sent from branch offices (11 chunks)

Because there are 10 branch offices, 10 chunks are needed to import the data of those branch offices. In addition, one chunk is needed to merge those 10 chunks. Therefore, 11 chunks are needed in total.

- The chunks containing product data that are not subject to analysis (six chunks)

Assuming a safety factor of 1.2, the maximum number of chunks that can be created is as follows:

```
maximum-number-of-chunks-to-be-created = ↑(1 + 11 + 6) × 1.2 (safety factor)↑ = 22
```

> **Note**
>
> If the number of chunks increases, this negatively impacts retrieval performance. Therefore, we recommend that you design and operate your database system to minimize the number of chunks that are used.

Based on this estimate, specify `22` as the maximum number of chunks. Specify this value for `CHUNK` of the chunk specification in the `CREATE TABLE` statement. The following shows an SQL statement example.

■ SQL statement example

```
CREATE TABLE "ITEMLIST"
      ("SHOP_ID" INT,"DELIVERY_TIME" TIMESTAMP,
       "ITEM_ID" INT, "ITEM_NAME" VARCHAR(100))
    IN "TBLDAT002" CHUNK=22
```

For details about the `CREATE TABLE` statement, see *CREATE TABLE (define a table)* in *Definition SQL* in the manual *HADB SQL Reference*.

(b) Defining indexes for the product table (ITEMLIST)

Define a B-tree index and a range index for the product table. Define these indexes so that they will be stored in DB areas `IDXDAT_002` and `RIXDAT_002`, respectively.

Define a B-tree index for the product ID (`ITEM_ID` column) and a range index for the delivery time (`DELIVERY_TIME` column). The following shows an SQL statement example.

■ SQL statement example

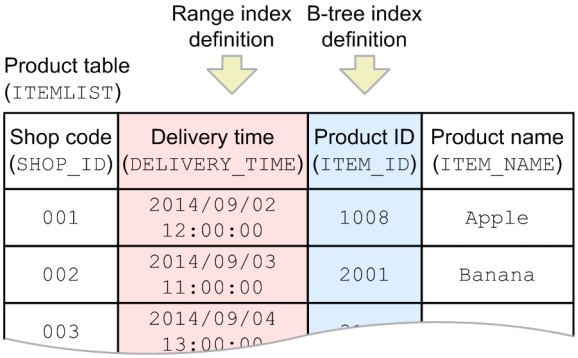
```
CREATE INDEX "ITEM_ID_IDX"
   ON "ITEMLIST"("ITEM_ID") IN "IDXDAT_002"
      EMPTY

CREATE INDEX "DELIVERY_TIME_RIX"
   ON "ITEMLIST"("DELIVERY_TIME") IN "RIXDAT_002"
      EMPTY INDEXTYPE RANGE
```

For details about the `CREATE INDEX` statement, see *CREATE INDEX (define an index)* in *Definition SQL* in the manual *HADB SQL Reference*.

(c) Table used in this operation example

The following figure shows a portion of the table used in this operation example.

Figure 11-32: Portion of the table used in this operation example



(2) Adding the next month's product data to chunks in wait status (background import)

The next month's product data is sent from each branch office by the 25th of each month. At the main office, the next month's product data is added to the product table (ITEMLIST) as it arrives from the branch offices.

Background import is used to add the next month's product data. During this process, the next month's product data needs to be kept excluded from analysis until the next month begins. Therefore, the data is added as chunks in wait status.

The group of data that is stored in the multi-chunk table in a single background import operation is managed together with its indexes as a group (chunk). For details about a chunk, see 2.14.2 Managing data in data-import units (chunks).

Note

By managing data in units of chunks, you can acquire backups and delete data on a chunk-by-chunk basis. When you perform background import or merge chunks, you can add a comment to a chunk. Adding a comment to a chunk makes it easier to identify a certain chunk you want to manage.

To perform background import to create a chunk in wait status, use the adbimport command with the -b and --status wait options specified. The following shows an example of executing the adbimport command.

■ Command execution example

```
adbimport -u ADBUSER01 -p '#HelloHADB_01' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file02.txt
          -b --status wait
          -m 'October 2014'
          ITEMLIST
          /home/adbmanager/imp_file/imp_data_path02.txt
```

Explanation

Use background import to import the contents of the input data file, set up in the input data path file (/home/adbmanager/imp_file/imp_data_path02.txt), into the product table (ITEMLIST).

Whenever the next month's product data arrives from the ten branch offices, background import is performed. As a result, ten chunks in wait status are created.

For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*. Also, see 11.4.4 Temporarily excluding data to be imported to a multi-chunk table from retrieval (creating a chunk in wait status).

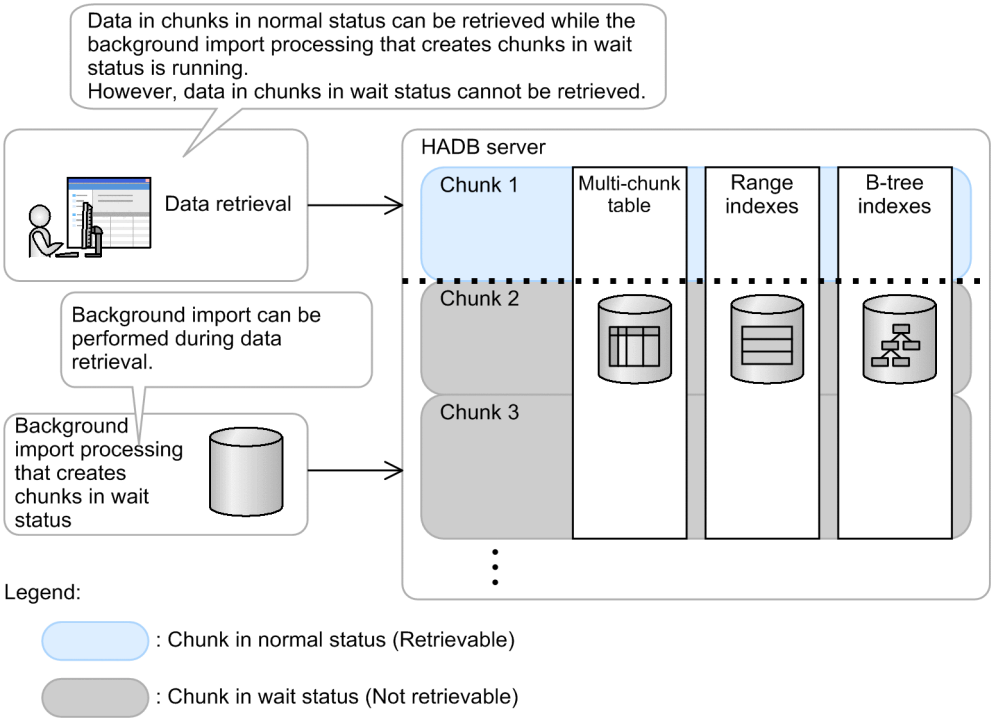
(3) Retrieving data

Even while the next month's product data is being added to the database, the current month's product data is retrieved and analyzed.

By using background import to add data, you can import data even if another user is retrieving data from the same database (data can be imported during data retrieval).

The following figure shows the relationship between background import and data retrieval.

Figure 11-33: Relationship between background import that creates chunks in wait status and data retrieval



Although the current month's product data can be analyzed, the next month's product data cannot be analyzed until the new month begins because it is excluded from analysis (chunks in wait status cannot be retrieved).

(4) Merging chunks in wait status (to reduce the number of chunks)

Performing background import whenever the next month's product data arrives from the branch offices creates multiple chunks in wait status. When these chunks in wait status are changed to chunks in normal status to be included in the analysis, the increase in the number of chunks might degrade the retrieval performance.

Therefore, by merging the multiple chunks in wait status into a single chunk, you can prevent retrieval performance degradation that tends to result from an increase in the number of chunks. In this example, all product data that has been sent from the branch offices is first added to the database and is then merged into a single chunk.

(a) Identifying the chunks in wait status to be merged

To identify the chunk IDs of the chunks in wait status to be merged, retrieve the system table STATUS_CHUNKS as explained below.

The following show an SQL statement example.

■ SQL statement example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
   WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='ITEMLIST'
      AND "CHUNK_COMMENT"='October 2014'
```

■ Retrieval result

```
 CHUNK_ID
 --------------------
                    2
           (omitted)
                   11
```

Explanation

In this example, the chunk IDs (2 to 11) of the chunks to be merged are identified by using the chunk comment ('October 2014'), added during background import, as the key during retrieval of the STATUS_CHUNKS table.

Check the chunks to be merged and their chunk IDs based on the explanation in 11.4.8 Checking the chunk status and the number of chunks created.

(b) Merging chunks in wait status

To merge chunks in wait status, use the adbmergechunk command. The following shows an execution example of the adbmergechunk command.

■ Command execution example

```
adbmergechunk -u ADBUSER01 -p '#HelloHADB_01' -g 2
              -w /home/adbmanager/tmp
              -z /home/adbmanager/merge_file/merge_opt_file02.txt
              -m 'October 2014_merge'
              -c 2-11
              ITEMLIST
```

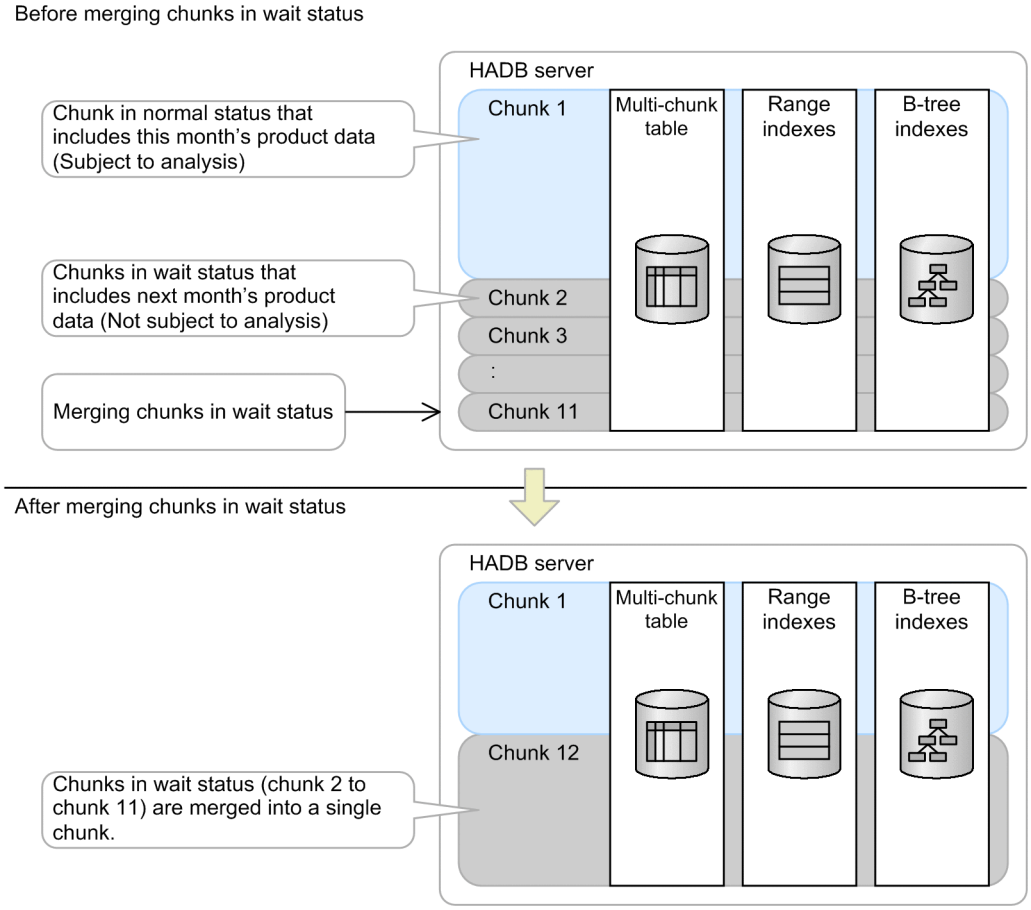
Explanation

Using the values specified in the merge chunk option file (/home/adbmanager/merge_file/merge_opt_file02.txt) as the input information, perform chunk merging on the product table (ITEMLIST) as specified in the -c option of the adbmergechunk command.

In this example, the next month's product data (the data for October 2014) is grouped together by merging chunks with chunk IDs 2 through 11 into a single chunk in wait status.

The following figure shows an example of merging chunks in wait status.

Figure 11-34: Example of merging chunks in wait status



Chunks with chunk IDs `2` through `11` are automatically deleted when the `adbmergechunk` command terminates normally.

Note

Merging chunks in wait status can prevent degradation in retrieval performance when these chunks are changed to chunks in normal status. There is also a limit to the number of chunks that can be managed by a table or DB area. If the number of chunks is insufficient, you can reduce the number by merging multiple chunks into a single chunk.

For details about the `adbmergechunk` command, see *adbmergechunk (Merge Chunks)* in the manual *HADB Command Reference*. Also, see 11.4.9 Merging chunks (to reduce the number of chunks).

(5) Changing the chunk status (switching the product data to be analyzed)

When the new month begins, the product data that was excluded from analysis is made subject to analysis. Conversely, the product data that had been subject to analysis is now excluded from analysis.

When the new month begins, the chunk status is changed as follows:

- Chunks in normal status containing the product data that was subject to analysis are changed to chunks in wait status to exclude them from analysis.
- Chunks in wait status containing the product data that was excluded from analysis are changed to chunks in normal status to make them subject to analysis.

In this example, the product data for September 2014 is now excluded from analysis, and the product data for October 2014 is made subject to analysis.

(a) Identifying the chunk ID of the chunk whose status is to be changed

To identify the chunk ID of the chunk whose status is to be changed, retrieve the system table STATUS_CHUNKS as explained in the following.

First, identify the chunk ID of the product data for September 2014.

■ SQL statement example (changing a chunk in normal status to a chunk in wait status)

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
   WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='ITEMLIST'
      AND "CHUNK_COMMENT"='September 2014_merge'
```

■ Retrieval result

```
 CHUNK_ID
 --------------------
                    1
```

Explanation

In this example, the chunk ID of the chunk containing the product data for September 2014 is identified using the chunk comment ('September 2014_merge') that was added during chunk merging, as the key during retrieval of the STATUS_CHUNKS table.

Next, identify the chunk ID of the product data for October 2014.

■ SQL statement example (changing a chunk in wait status to a chunk in normal status)

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
   WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='ITEMLIST'
      AND "CHUNK_COMMENT"='October 2014_merge'
```

■ Retrieval result

```
 CHUNK_ID
 --------------------
                   12
```

Explanation

In this example, the chunk ID of the chunk containing the product data for October 2014 is identified using the chunk comment ('October 2014_merge') that was added during chunk merging, as the key during retrieval of the STATUS_CHUNKS table.

(b) Changing the chunk status

To change the chunk status, use the adbchgchunkstatus command. Chunk statuses are changed as described in this subsection.

- Changing a chunk in normal status to a chunk in wait status
  Below, the chunk containing the product data for September 2014 (chunk ID: 1) is the one whose status is changed.
- Changing a chunk in wait status to a chunk in normal status
  Below, the chunk containing the product data for October 2014 (chunk ID: 12) is the one whose status is changed.

The following is an execution example of the adbchgchunkstatus command.

■ **Command execution example**

```
adbchgchunkstatus -u ADBUSER01 -p '#HelloHADB_01'
                  -w 1 -n 12 ITEMLIST
```

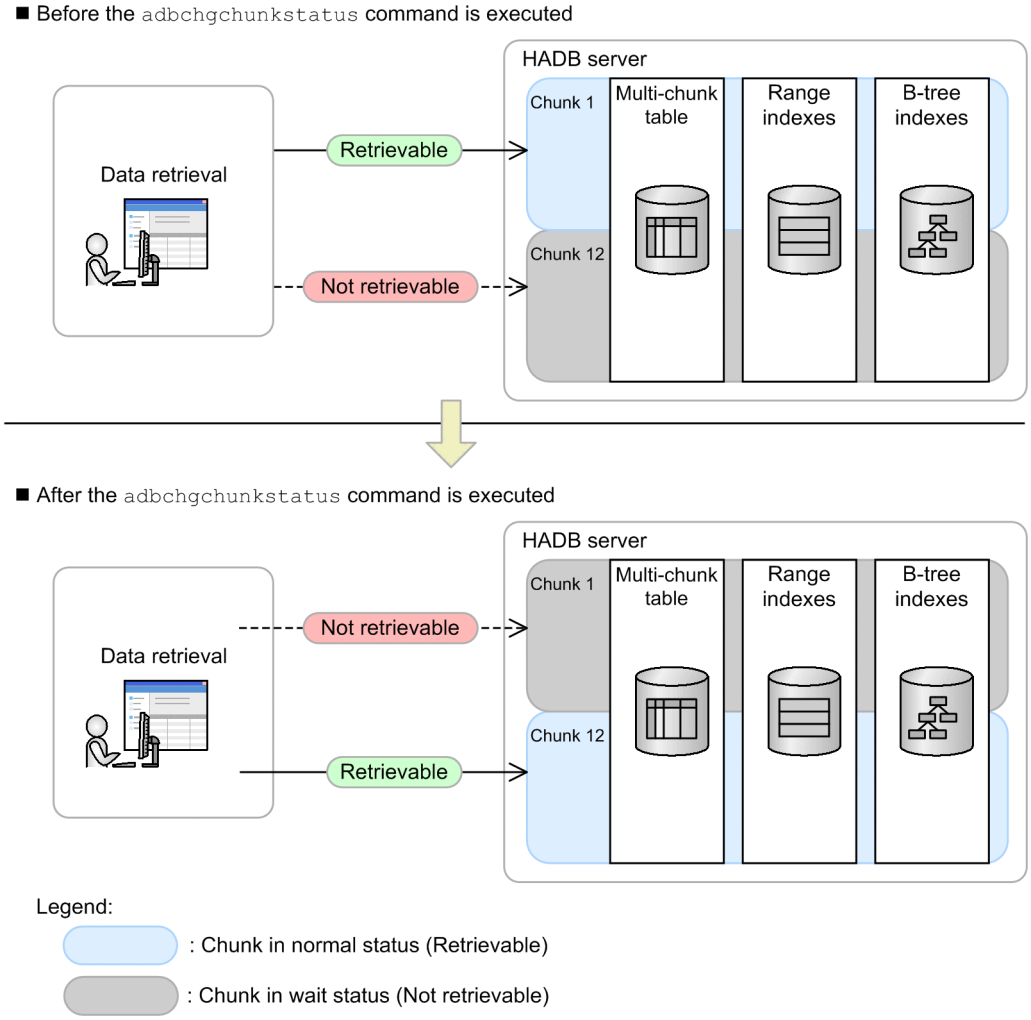
Explanation

The chunk status in the product table (`ITEMLIST`) is changed as specified in the `-w` and `-n` options.

In this example, the chunk with chunk ID `1` is changed to wait status, and the chunk with chunk ID `12` is changed to normal status.

The following figure shows an example of changing chunk statuses.

Figure 11-35: Example of changing chunk statuses



Explanation

The chunk containing the product data for September 2014 (chunk ID: `1`) is excluded from analysis, and the chunk containing the product data for October 2014 (chunk ID: `12`) becomes subject to analysis.

For details about the `adbchgchunkstatus` command, see *adbchgchunkstatus (Change Chunk Status)* in the manual *HADB Command Reference*. Also, see 11.4.12 Changing the chunk status.

(6) Backing up data

To prevent the amount of data from becoming unmanageable, data older than 6 months is backed up and then deleted. The database is backed up every month.

(a) Identifying the chunks with the earliest creation date and time

Chunks that are older than 6 months (chunks with the earliest creation date and time) are identified in the product table (ITEMLIST). To identify these chunks in the product table, retrieve the system table STATUS_CHUNKS as described below.

The following shows an SQL statement example.

■ SQL statement example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
   WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='ITEMLIST'
      AND "CREATE_TIME"=(SELECT MIN("CREATE_TIME")
         FROM "MASTER"."STATUS_CHUNKS"
            WHERE TC."TABLE_SCHEMA"='ADBUSER01'
               AND TC."TABLE_NAME"='ITEMLIST')
```

■ Retrieval result

```
 CHUNK_ID
--------------------
                   1
```

Explanation

When the STATUS_CHUNKS table is retrieved, the result shows that the chunk with chunk ID 1 is the chunk with the earliest creation date and time (older than 6 months).

> **Note**
>
> When chunks are merged, the earliest creation date and time among the merge-source chunks is stored in the STATUS_CHUNKS table.

For details about how to retrieve data from the STATUS_CHUNKS table, see C.9 Searching system tables.

(b) Making a backup

Now that the chunk ID of the chunk with the earliest creation date and time in the product table (ITEMLIST) has been identified as 1, back up the data in that chunk in a CSV file.

To make a backup, use the adbexport command. The following shows an execution example of the adbexport command.

■ Command execution example

```
adbexport -u ADBUSER01 -p '#HelloHADB_01'
          -z /home/adbmanager/exp_file/exp_opt_file02.txt
          -n ITEMLIST
          -c 1
          /home/adbmanager/exp_file/exp_data_path02.txt
```

■ **Content of the output data path file (/home/adbmanager/exp_file/exp_data_path02.txt)**

```
/home/adbmanager/exp_backup/backup0101.csv
/home/adbmanager/exp_backup/backup0102.csv
/home/adbmanager/exp_backup/backup0103.csv
```

Explanation

The data contained in the chunk with chunk ID `1` is backed up in CSV files (`/home/adbmanager/exp_backup/backup0101.csv`, `/home/adbmanager/exp_backup/backup0102.csv`, and `/home/adbmanager/exp_backup/backup0103.csv`).

In this example, since chunks are merged every month, one month's worth of data is backed up.

> **Note**
>
> Although you can back up the files into a single file, distributing the files into multiple files can improve performance by distributing the output-related overhead.

For details about the `adbexport` command, see *adbexport (Export Data)* in the manual *HADB Command Reference*. Also, see 11.4.5 Exporting data in units of chunks.

(7) Deleting old data (chunks)

To prevent the amount of data from becoming unmanageable, delete data that is older than 6 months.

Delete the chunk with the earliest creation date and time from the product table (`ITEMLIST`). To delete a chunk, execute the `PURGE CHUNK` statement with a chunk ID specified in the search condition.

■ **SQL statement example**

```
PURGE CHUNK "ITEMLIST" WHERE CHUNKID=1
```

Explanation

The data contained in the chunk with chunk ID `1` is deleted.

In this example, since chunks are merged every month, one month's worth of data is deleted.

For details about the `PURGE CHUNK` statement, see *PURGE CHUNK (delete all rows in a chunk)* in *Data Manipulation SQL* in the manual *HADB SQL Reference*. Also, see 11.4.6 Deleting data in units of chunks.

(8) Repeatedly adding new data, merging, changing status, and deleting old data

By repeatedly performing background import to create chunks in wait status, by merging chunks in wait status, changing chunk statuses, and by deleting chunks using the `PURGE CHUNK` statement, you can continue to analyze only the product data of a specific month without increasing the overall volume of data managed by HADB.

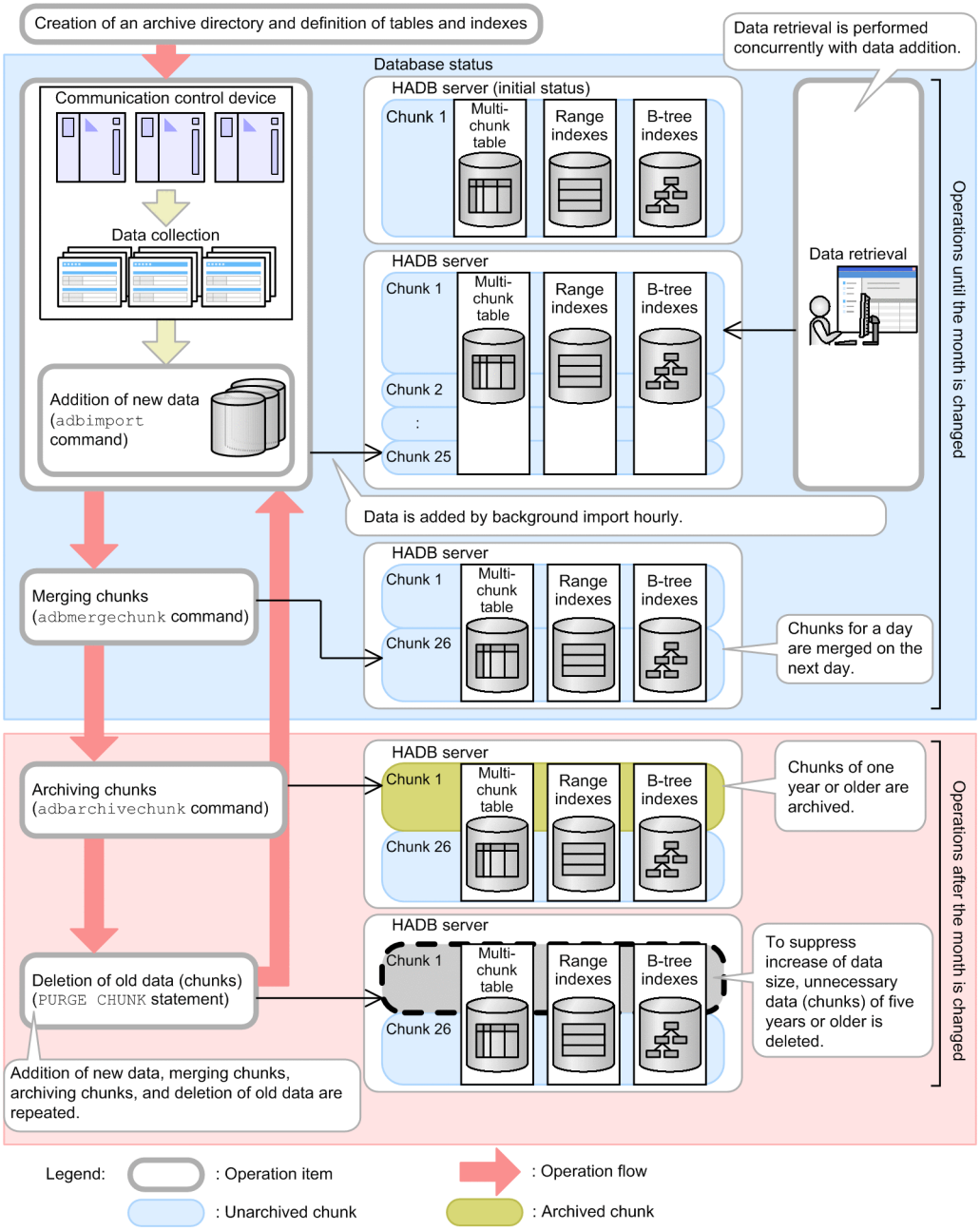
11.4.25 Operation taking archive of chunks into consideration

This subsection describes, by using the following example, how to suppress increase of database size by archiving old data that is less frequently retrieved. The prerequisites for this operation are as follows:

- The sensor data that is generated from nationwide communication control devices (information about communication traffic volume and fault sign detection) is collected in the main office. Then, the sensor data is retrieved and analyzed to determine whether communication control devices need to be replaced or increased.
- The sensor data collected by the main office is managed centrally using the communication control device log table.
- The sensor data is imported to the database hourly.
- A group of sensor data (for a day), each of which is imported hourly, is merged into a single chunk (chunk for a day) when the day is changed. Also, a group of sensor data (for a month) is merged into a single chunk (chunk for a month) when the month is changed. Eventually, chunks are managed in units of months.
- Sensor data is retrieved even while sensor data is being imported into the database. The sensor data that can be retrieved is the data for the last 24 hours (daily unit), for the last seven days (weekly unit), and for the last one month (monthly unit). Based on the retrieval result, whether to replace or increase communication control devices is determined.
- As a result of retrieval of sensor data, if the communication traffic volume exceeds the threshold value, data for the last five years is analyzed by using the applicable day and the previous and next days as keys. Communication traffic volume might exceed the threshold value approximately once a month.
- Because the sensor data imported to the database over a year ago is seldom retrieved, it is archived in unit chunks. To reduce the database size, sensor data is archived when the month is changed.
- The database retains five years' worth of data. The data that was imported more than 5 years ago is deleted in unit chunks when the month is changed.

The following figure shows the flow of operations in this system.

Figure 11-36: Flow of operations in this system (using archived chunks)



(1) Creating an archive directory

In this operation example, to reduce the data size, data older than one year is archived on a chunk basis. To archive chunks, an archive directory is required for storing archive files. Before starting operations, create an archive directory.

Create an archive directory by using the following procedure.

Procedure:

1. Log on to the OS as a superuser.
   Log on to the server machine's OS as a superuser.
2. Configure a file system for the archive directory.
   Execute an OS command to configure a file system for the archive directory.

The following example configures /dev/vg_hadb/hadb_archivedir, which has been created as an LV for the archive directory, in the file system ext4.

▪ **Command execution example**

```
mkfs -t ext4 /dev/vg_hadb/hadb_archivedir
```

3. Create the mount point for the archive directory.

   Execute an OS command to create the mount point for the archive directory. The following shows an execution example.

   ▪ **Command execution example**

```
mkdir -p /HADB/archive
```

4. Activate the VG that contains the file system for the archive directory.

   Execute an OS command to activate the VG that contains the file system for the archive directory. The following shows an execution example.

   ▪ **Command execution example**

```
vgchange -a y /dev/vg_hadb
```

5. Mount the file system for the archive directory.

   Execute an OS command to mount the file system for the archive directory. The following shows an execution example.

   ▪ **Command execution example**

```
mount /dev/vg_hadb/hadb_archivedir /HADB/archive -t ext4 -o defaults,noatime,_netd
ev4
```

6. Change the owner of the archive directory.

   Execute an OS command to change the owner of the archive directory. Specify the HADB administrator and HADB administrators group that use the archive directory.

   The following shows an execution example in which the user name of the HADB administrator is adbmanager and the HADB administrators group is adbgroup.

   ▪ **Command execution example**

```
chown adbmanager.adbgroup /HADB/archive
```

(2) Defining an archivable multi-chunk table

After creating an archive directory, define an archivable multi-chunk table and indexes.

In this operation example, the communication control device log table (INSTRUMENT_LOG) and indexes are defined to manage sensor data centrally.

(a) Defining the communication control device log table (INSTRUMENT_LOG)

Define the communication control device log table (INSTRUMENT_LOG) to be stored in the DB area TBLDAT_003. Also, define the communication control device log table as an archivable multi-chunk table to perform background import and archive of chunks.

In this operation example, the following number of chunks are required:

- Chunks that manage the sensor data hourly added by background import (24 chunks)

  The number of chunks for a day is 24 because data is added hourly by background import.

- Chunks that manage the sensor data for individual days (32 chunks)

  The total number of chunks is 32 because one chunk is needed as the merge-target chunk, in addition to the chunks for a month (for 31 days).
- Chunks that manage the sensor data for individual months (61 chunks)

  The total number of chunks is 61 because one chunk is needed as the merge-target chunk, in addition to the chunks for five years (60 months).

Assuming a safety factor of 1.2, the maximum number of chunks that can be created is as follows:

Maximum number of chunks that can be created = ↑(24 + 32 + 61) x 1.2↑ = 141

Note

If the number of chunks increases, this negatively impacts retrieval performance. Therefore, we recommend that you design and operate your database system to minimize the number of chunks that will be used.

Based on this estimate, specify `141` as the maximum number of chunks. Specify this value for `CHUNK` of the chunk specification in the `CREATE TABLE` statement.

For the collection date and time column (`C_TIME`) of the communication control device log table (`INSTRUMENT_LOG`), you can specify the range for data analysis. Also, data for each chunk does not overlap for this column. Therefore, use this column as the archive range column. Store the range index automatically defined for the archive range column, in the DB area `IDXDAT_003` in which other indexes defined for the communication control device log table (`INSTRUMENT_LOG`) are also stored.

Specify the following for the chunk-archive specification in the `CREATE TABLE` statement.

- Archive range column name (`C_TIME`)
- Name of the DB area that stores the range index automatically defined (`IDXDAT_003`)
- Created archive directory (`/HADB/archive`)

The following shows an example of the `CREATE TABLE` statement.

▪ Example of the CREATE TABLE statement

```
CREATE TABLE "INSTRUMENT_LOG"
            ("SID" INTEGER,
             "C_TIME" TIMESTAMP,
             "TRAFFIC" INTEGER,
             "THROUGH" INTEGER,
             "RETRY" INTEGER,
              ...)
  IN "TBLDAT_003"
  CHUNK= 141
  ARCHIVABLE RANGECOLUMN="C_TIME" IN "IDXDAT_003"
  ARCHIVEDIR='/HADB/archive'
```

For details about the `CREATE TABLE` statement, see *CREATE TABLE (define a table)* in *Definition SQL* in the manual *HADB SQL Reference*.

(b) Defining indexes for the communication control device log table (INSTRUMENT_LOG)

Define B-tree indexes for the communication control device log table. Store the B-tree indexes in the DB area IDXDAT_003.

Here, define a B-tree index for the device ID column (SID). The following shows an example of the CREATE INDEX statement.

▪ Example of the CREATE INDEX statement

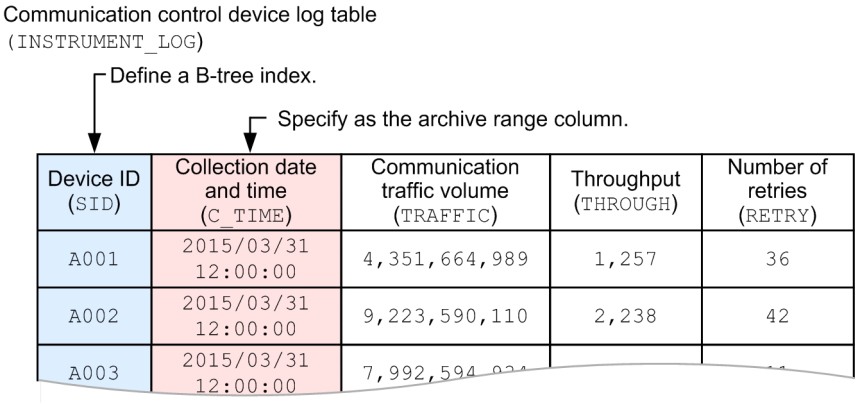
```
CREATE INDEX "SID_IDX"
   ON "INSTRUMENT_LOG"("SID") IN "IDXDAT_003"
   EMPTY
```

For details about the CREATE INDEX statement, see *CREATE INDEX (define an index)* in *Definition SQL Statements* in the manual *HADB SQL Reference*.

(c) Table used in this operation example

The following figure shows a portion of the table used in this operation example.

Figure 11-37: Portion of the table used in this operation example



(3) Adding new sensor data (background import)

Sensor data is generated in real time from communication control devices. The generated sensor data is added hourly to the communication control device log table (INSTRUMENT_LOG).

Sensor data is added by using background import. The group of data that is stored in the archivable multi-chunk table in a single background import operation is managed together with its indexes as a group (chunk). For details about a chunk, see 2.14.2 Managing data in data-import units (chunks).

> **Note**
>
> By managing data in units of chunks, you can acquire backups and delete data on a chunk-by-chunk basis. When you perform background import or merging chunks, you can add a comment to a chunk. Adding a comment to a chunk makes it easier to identify a certain chunk you want to operate on.

Execute the adbimport command with the -b option specified to perform background import. The following shows an example of executing the adbimport command.

▪ **Command execution example**

```
adbimport -u ADBUSER01 -p '#HelloHADB_01' -k "'" -s , -g 10
          -w /home/adbmanager/tmp
          -z /home/adbmanager/imp_file/imp_opt_file03.txt
          -b
          -m '2015/03/31'
          INSTRUMENT_LOG
          /home/adbmanager/imp_file/imp_data_path03.txt
```

Explanation:

Use background import to import the contents of the input data file, set up in the input data path file (`/home/adbmanager/imp_file/imp_data_path03.txt`), into the communication control device log table (`INSTRUMENT_LOG`).

Specify the comment to be added to a chunk for the `-m` option, to add the sensor data generated on March 31, 2015, by using background import.

Perform background import hourly. As a result, 24 chunks are created for one day (for 24 hours) .

For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*. Also, see 11.4.2 Storing data in a multi-chunk table (background import).

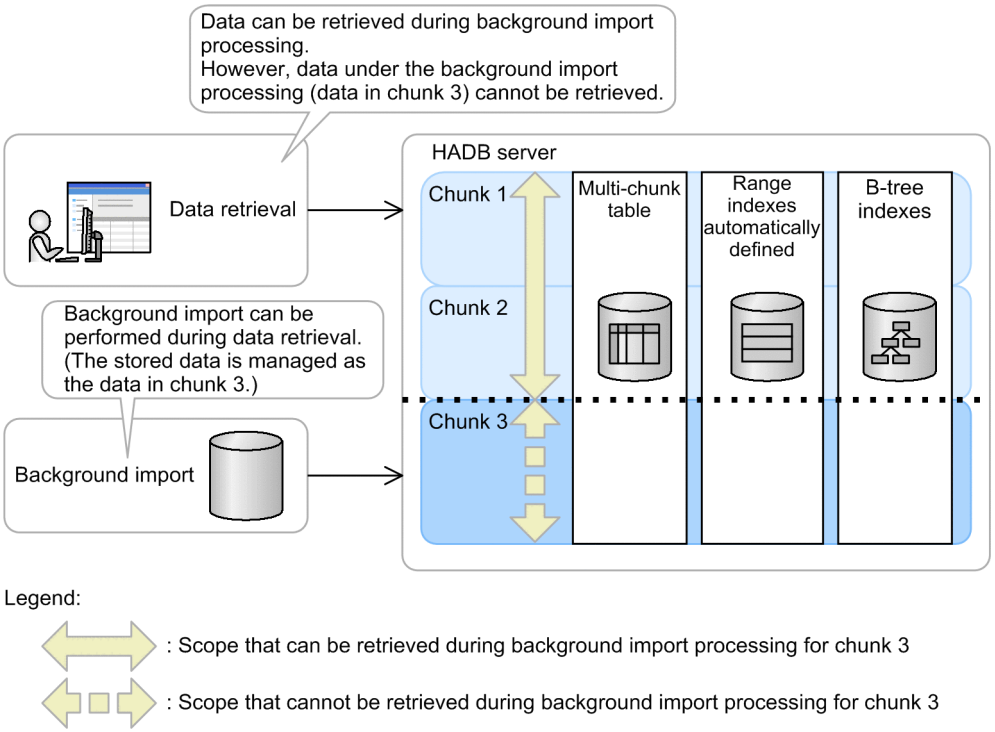
(4) Retrieving data

You can retrieve data in an archivable multi-chunk table even while background import is performed.

By using background import to add sensor data, you can import data even if another HADB user is retrieving data from the same database (data can be imported during data retrieval).

The following figure shows the relationship between background import and data retrieval.

Figure 11-38: Relationship between background import and data retrieval



(5) Merging chunks (to reduce the number of chunks)

If the number of chunks increases, retrieval performance might decline. Therefore, you need to regularly merge chunks. In this operation example, sensor data is hourly imported by background import, so 24 chunks are created per day.

In this operation example, multiple chunks are merged into a single chunk at the following timing. Eventually, chunks are managed in units of months.

- Chunks of 24 per day (24-hour period) are merged into a single chunk.
- Chunks of 31 per month (31-day period) are merged into a single chunk.

The following shows an example of merging data for a day (for March 31, 2015) into a single chunk.

(a) Identifying the chunks to be merged

To identify the chunk IDs of the chunks to be merged, retrieve the system table `STATUS_CHUNKS` as follows. The following shows an SQL statement example.

▪ SQL statement example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
   WHERE "TABLE_SCHEMA"='ADBUSER01' AND "TABLE_NAME"='INSTRUMENT_LOG'
      AND "CHUNK_COMMENT"='2015/03/31'
```

▪ Retrieval result

```
 CHUNK_ID
 --------------------
                   92
           (Omission)
                  115
```

Explanation:

In this example, the `STATUS_CHUNKS` table is retrieved by using the comment added to the chunk during background import (`'2015/03/31'`) as the key. Then, the chunk ID of the chunk to be merged (`92` to `115`) is identified.

Check the chunks to be merged and their chunk IDs based on the explanation in 11.4.8 Checking the chunk status and the number of chunks created.

(b) Merging chunks

Execute the `adbmergechunk` command to merge chunks. The following shows an example of executing the `adbmergechunk` command.

▪ Command execution example

```
adbmergechunk -u ADBUSER01 -p '#HelloHADB_01' -g 2
              -w /home/adbmanager/tmp
              -z /home/adbmanager/merge_file/merge_opt_file03.txt
              -m '2015/03/31'
              -c 92-115
              INSTRUMENT_LOG
```

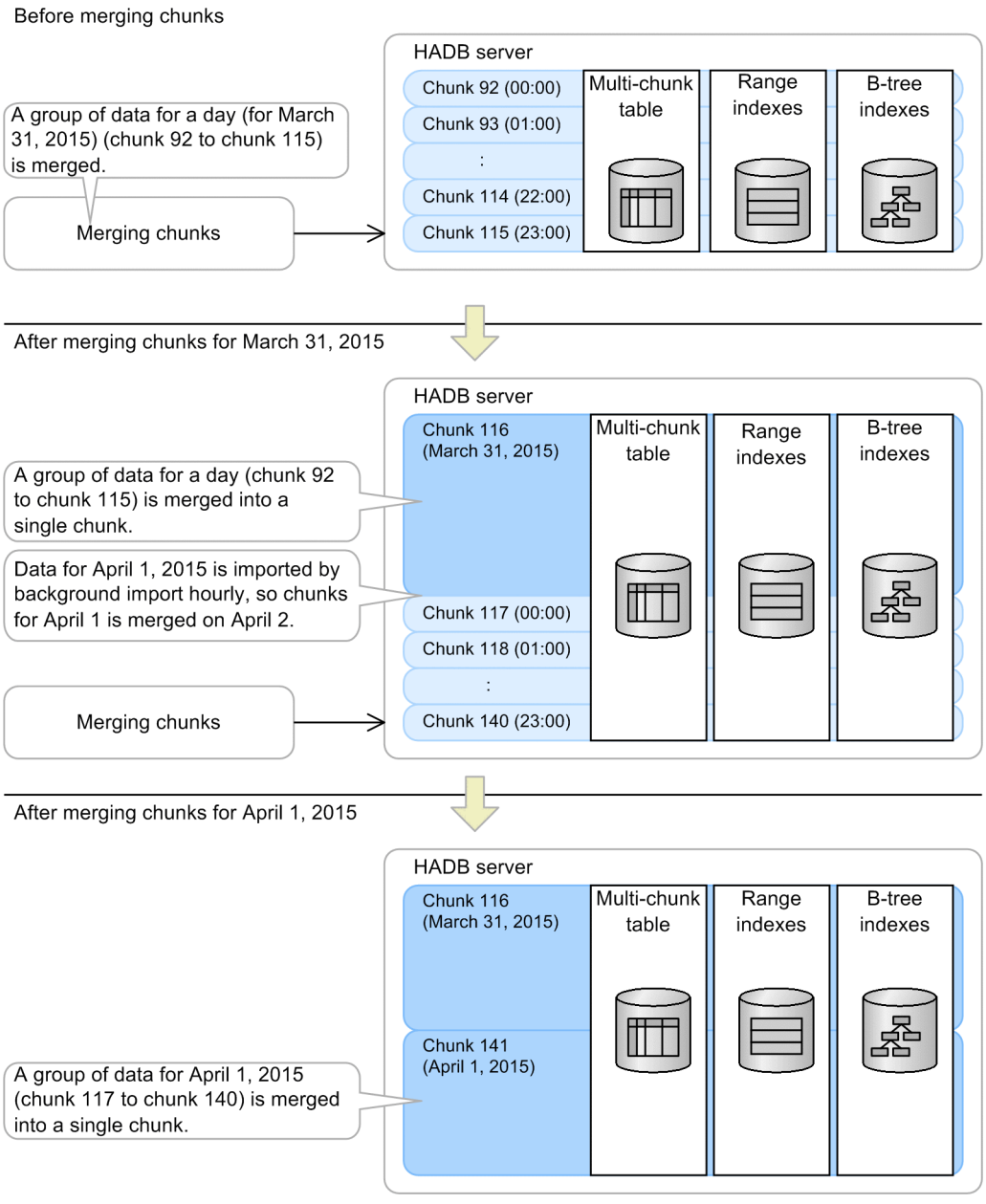
Explanation:

For the communication control device log table (`INSTRUMENT_LOG`), chunks specified for the `-c` option are merged by using the value specified for the merge chunk option file (`/home/adbmanager/merge_file/merge_opt_file03.txt`) as the input information.

In this example, the chunks whose chunk IDs are from 92 to 115 are merged into a single chunk in wait status as the sensor data for a day (for March 31, 2015).

The following figure shows an example of merging chunks.

Figure 11-39: Example of merging chunks



The chunks whose chunk IDs are from `92` to `115` are automatically deleted when the `adbmergechunk` command terminates normally.

Note

By merging chunks, you can prevent retrieval performance from declining. There is a limit on the number of chunks that can be managed by a table or a DB area. If the number of chunks being used reaches the upper limit of the number of chunks that can be managed by a table or a DB area, merge the multiple chunks into a single chunk. By merging chunks, you can reduce the number of chunks being used.

For details about the `adbmergechunk` command, see *adbmergechunk (Merge Chunks)* in the manual *HADB Command Reference*. Also, see 11.4.9 Merging chunks (to reduce the number of chunks).

(6) Archiving chunks (compressing data in unit chunks)

In this operation example, data imported to the database more than a year ago is seldom retrieved, so it is archived (compressed) in unit chunks. By archiving chunks, you can reduce the database size.

Important

Retrieving data in an archived chunk requires longer time than retrieving data in an unarchived chunk due to the following reasons:

- Expanding data takes time.
- Data retrieval by using indexes is not available.

As is performed in this operation example, archive only the data that is seldom retrieved but needs to be retained for a long time. For details, see 2.15.2 When the chunk archiving function is to be used.

(a) Executing archive of chunks

Execute the `adbarchivechunk` command to archive chunks. In this example, the chunks that manage the sensor data for March in 2014 are archived. The following shows an example of executing the `adbarchivechunk` command.

▪ Command execution example

```
adbarchivechunk -u ADBUSER01 -p '#HelloHADB_01' -g 2
                -z /home/adbmanager/arc_file/archive_chunk03.def
                -r 2014/03/01-2014/03/31
                INSTRUMENT_LOG
```

Explanation:

Specify for the `-r` option of the `adbarchivechunk` command the range of dates for the chunks to be archived (`2014/03/01-2014/03/31`). The chunks for which the value range in the archive range column is included in the date range specified for the `-r` option are archived. In this operation example, the collection date and time column (`C_TIME`) of the communication control device log table (`INSTRUMENT_LOG`) is specified as the archive range column. Therefore, based on the value in the collection date and time column (`C_TIME`), the chunks for March, 2014 are archived.

(b) Checking the execution result of archiving

Execute the `adbdbstatus` command to check the status of archived chunks. By executing the `adbdbstatus` command to output the archived chunk summary information, you can check the status of archived chunks. The following shows an example of executing the `adbdbstatus` command.

▪ **Command execution example**

```
adbdbstatus -c archivechunk
            -n ADBUSER01.INSTRUMENT_LOG
            -r 2014/03/01-2014/03/31
```

Explanation:

In this example, the summary information of archived chunks (for March, 2014) is output. From the following output items, you can check the number of rows stored in an archived chunk, the data size of an archived chunk, and the compression ratio of an archived chunk.

- `Rows` (Number of rows stored in an archived chunk)
- `Archive_file_size` (Total size of all archive files corresponding to an archived chunk)
- `Unarchive_table_size` (Size of segments that were used to store a table in the chunk before being archived)
- `Unarchive_index_size` (Size of segments that were used to store an index in the chunk before being archived)
- `Compression_ratio` (Compression ratio of an archived chunk)

(7) Deleting old data (chunks)

To suppress increase of data size, delete data (archived chunks) that was added more than five years ago.

From the communication control device log table (`INSTRUMENT_LOG`), delete old chunks whose data was added more than five years ago. Check the chunk IDs of the chunks to be deleted, and then execute the `PURGE CHUNK` statement to delete the chunks.

Here, the chunks for April, 2010 will be deleted.

(a) Checking the chunk ID

Execute the `adbdbstatus` command to check the chunk IDs of the chunks to be deleted. By executing the `adbdbstatus` command to output the archived chunk summary information, you can check the chunk IDs of the chunks to be deleted. The following shows an example of executing the `adbdbstatus` command.

▪ **Command execution example**

```
adbdbstatus -c archivechunk
            -n ADBUSER01.INSTRUMENT_LOG
            -r 2010/04/01-2010/04/30
```

Explanation:

In this example, the summary information of archived chunks (for April, 2010) is output. From the following output items, you can check the chunk IDs of the chunks to be deleted and the free space that is secured in the archive directory after the chunks are deleted.

- `Chunk_ID` (Chunk ID)
- `Archive_file_size` (Total size of all archive files corresponding to an archived chunk)

(b) Deleting chunks

Delete chunks based on the chunk IDs that you checked by using the `adbdbstatus` command. To delete chunks, specify the chunk IDs for the search condition of the `PURGE CHUNK` statement. The following shows an example of the `PURGE CHUNK` statement.

▪ **SQL statement example**

```
PURGE CHUNK "INSTRUMENT_LOG" WHERE CHUNKID=4
```

Explanation:

This example deletes the data contained in the chunk whose chunk ID is `4`.

In this example, because chunks are merged in units of month, data for one month is deleted.

For details about the `PURGE CHUNK` statement, see *PURGE CHUNK (delete all rows in a chunk)* in *Data Manipulation SQL* in the manual *HADB Command Reference*. Also, see 11.4.6 Deleting data in units of chunks.

(8) Repeating addition, merge, and archive of new data and deletion of old data

By repeating the following operations, you can suppress the size of data to be retained for a long time, and continue analyzing sensor data:

- Executing background import
- Merging chunks
- Archiving chunks
- Deleting chunks by using the `PURGE CHUNK` statement

11.5 Collecting cost information

Cost information is collected in order to improve data-search efficiency. This information is related to search methods, and it is collected from base tables, B-tree indexes, and text indexes by using the `adbgetcst` command. The collected cost information is stored in system tables and used during retrievals.

The following is a specification example of the `adbgetcst` command.

■ Specification example

`USER01` collects cost information from table `T1`:

```
adbgetcst -u USER01 -t T1
```

The following bullet points explain the timing for collecting cost information with the `adbgetcst` command.

■ Timing for collecting cost information

We recommend that you collect cost information whenever you perform either of the following tasks:

- Storing data in a base table by executing the `adbimport` command

  Regardless of whether the creation mode or the addition mode is used, we recommend that you collect cost information whenever you execute the `adbimport` command to store data in a base table.

- Adding, updating, or deleting a large amount of data

  We recommend that you collect cost information whenever you have used the `INSERT`, `UPDATE`, or `DELETE` statement to add, update, or delete a large amount of data.

- Rebuilding indexes by executing the `adbidxrebuild` command

  We recommend that you collect cost information whenever you have used the `adbidxrebuild` command to rebuild a B-tree index or text index.

We recommend that you also collect cost information from time to time if you have added, updated, or deleted small amounts of data multiple times, even if you don't process large amounts of data all at once.

When cost information is collected, all rows of the target table are retrieved, which means that it will take a considerable amount of time for the collection processing to be completed.

For details about the `adbgetcst` command, see *adbgetcst (Collect Cost Information)* in the manual *HADB Command Reference*.

11.6 Managing HADB users

This section explains how to manage HADB users.

11.6.1 Creating HADB users

The CREATE USER definition SQL statement is used to create HADB users.

The CREATE USER statement must be executed by an HADB user possessing the DBA privilege and CONNECT privilege.

The following shows an example of creating an HADB user.

Specification example

Create an HADB user with the following authorization identifier (user ID) and password:

- Authorization identifier: ADBUSER02
- Password: #HelloHADB_02

```
CREATE USER "ADBUSER02" IDENTIFIED BY '#HelloHADB_02'
```

For the password to specify when creating an HADB user, we recommend that you use a minimum of eight characters.

For details about the specification rules for authorization identifiers and passwords of HADB users, see 9.4.2 Authorization identifier specification rules and 9.4.3 Password specification rules.

Important

No privileges have been granted to the created HADB user. Grant the necessary user privileges based on the explanation in 11.7.1 Granting user privileges and the schema operation privilege to HADB users.

11.6.2 Changing an HADB user's password

To change an HADB user's password, use the ALTER USER definition SQL statement. The following shows an example of changing the password of an HADB user.

Specification example

The password of HADB user ADBUSER02 is changed to #HelloHADB_03.

```
ALTER USER "ADBUSER02" IDENTIFIED BY '#HelloHADB_03'
```

Important

- An HADB user who does not have the DBA privilege can change only his or her own password (that of the HADB user whose authorization identifier was used for the current connection to the HADB server). Such a user cannot change the password of another HADB user.
- An HADB user who has the DBA privilege can change their own password. They can also change the passwords of other HADB users. However, they cannot change the password of an HADB user who has the audit privilege.

Note

- When changing the password of an HADB user, we recommend that you use a minimum of eight characters for the new password.
- We also recommend that you change passwords regularly.
- For details about the specification rules for passwords of HADB users, see 9.4.3 Password specification rules.

11.6.3 Deleting HADB users

To delete an HADB user, use the DROP USER definition SQL statement. The following shows an example of deleting an HADB user.

Specification example

HADB user ADBUSER02 is deleted.

```
DROP USER "ADBUSER02" CASCADE
```

An HADB user who has the DBA privilege and the CONNECT privilege can use the DROP USER statement to delete HADB users.

Note

- Before deleting an HADB user, we recommend that you execute the adbchgsrvmode --offline command to change the operation mode of the HADB server to the offline mode. Delete an HADB user only after disabling connection requests from HADB clients.
- You cannot delete an HADB user who has the audit privilege. To delete such a user, you must first have their audit privilege revoked. For details about how to revoke an audit privilege, see (2) Deleting auditors (revoking audit privileges) in 12.4.1 Adding, deleting, and changing auditors (granting or revoking audit privileges).

For details about the DROP USER statement, see *DROP USER (delete an HADB user)* in *Definition SQL* in the manual *HADB SQL Reference*.

Important

Because the schemas and tables that are owned by the HADB user to be deleted are also deleted, the following might be affected:

- The viewed tables that depend on the table to be deleted by execution of the DROP USER statement (viewed tables for other schemas) are deleted or invalidated.
- The foreign keys that use as the referenced table the table to be deleted by execution of the DROP USER statement (foreign keys for other schemas) are also deleted.

For details about deleting schemas, see 11.9.2 Deleting a schema. If the base table defined in the schema is non-updatable, temporary work files created by the interrupted command might remain.

If the HADB user to be deleted granted access privileges to other HADB users, those access privileges are canceled.

11.7 Managing user privileges and the schema operation privilege

This section explains how to manage user privileges and the schema operation privilege.

For details about how to manage audit privileges, see 12.4.1 Adding, deleting, and changing auditors (granting or revoking audit privileges).

11.7.1 Granting user privileges and the schema operation privilege to HADB users

To grant user privileges and the schema operation privilege, use the GRANT definition SQL statement.

The GRANT statement must be executed by an HADB user possessing the DBA privilege and CONNECT privilege.

The following shows examples of granting user privileges and the schema operation privilege to an HADB user.

Specification example 1

The DBA privilege, CONNECT privilege, and schema definition privilege are granted to the HADB user ADBUSER02.

```
GRANT DBA,CONNECT,SCHEMA TO "ADBUSER02"
```

Specification example 2

The CONNECT privilege and schema definition privilege are granted to HADB users ADBUSER03 and ADBUSER04.

```
GRANT CONNECT,SCHEMA TO "ADBUSER03","ADBUSER04"
```

For details about the GRANT statement, see *GRANT (grant privileges)* in *Definition SQL* in the manual *HADB SQL Reference*.

> **Important**
>
> An HADB user cannot have both the audit admin privilege and the DBA privilege. This means that you cannot grant the DBA privilege to an HADB user who has audit admin privilege.

11.7.2 Checking the user privileges and schema operation privilege granted to an HADB user

To check which user privileges and schema operation privilege have been granted to an HADB user, search the SQL_USERS dictionary table.

An HADB user who has the DBA privilege can check the user privileges and schema operation privilege that all HADB users have. An HADB user who does not have the DBA privilege can check only his or her own user privileges and schema operation privilege (the privileges of the HADB user with the authorization identifier that was used for the current connection to the HADB server).

The following is an example of checking the user privileges and schema operation privilege that are granted to HADB users.

Specification example

An HADB user who has the DBA privilege and CONNECT privilege checks the user privileges and schema operation privilege granted to all created HADB users.

```
SELECT "USER_NAME","DBA_PRIVILEGE","CONNECT_PRIVILEGE","SCHEMA_PRIVILEGE"
  FROM "MASTER"."SQL_USERS"
```

If Y is specified for "DBA_PRIVILEGE", "CONNECT_PRIVILEGE", and "SCHEMA_PRIVILEGE" in the search result, the HADB user has the corresponding user privileges and schema operation privilege. If N is displayed, the HADB user does not have the corresponding user privileges and schema operation privilege.

For details about dictionary table SQL_USERS, see B.13 Content of SQL_USERS.

11.7.3 Revoking an HADB user's user privileges and schema operation privilege

You can use the REVOKE definition SQL statement to revoke user privileges and the schema operation privilege that were granted to an HADB user.

The REVOKE statement must be executed by an HADB user who has the DBA privilege and CONNECT privilege.

The following are examples of revoking user privileges and the schema operation privilege that were granted to HADB users.

Specification example 1

The DBA privilege, CONNECT privilege, and schema definition privilege that were granted to the HADB user ADBUSER02 are revoked.

```
REVOKE DBA,CONNECT,SCHEMA FROM "ADBUSER02" CASCADE
```

Important

Because, if the schema definition privilege is revoked, the schemas and tables that are owned by the target HADB user are also deleted, the following might be affected:

- The viewed tables that depend on the table to be deleted by execution of the REVOKE statement (viewed tables for other schemas) are deleted or invalidated.
- The foreign keys that use as the referenced table the table to be deleted by execution of the REVOKE statement (foreign keys for other schemas) are also deleted.

For details about deleting schemas, see 11.9.2 Deleting a schema. If the base table defined in the schema is non-updatable, temporary work files created by the interrupted command might remain.

Specification example 2

The CONNECT privilege and schema definition privilege that were granted to HADB users ADBUSER03 and ADBUSER04 are revoked. However, if ADBUSER03 and ADBUSER04 own schemas, the execution of the REVOKE statement is halted.

```
REVOKE CONNECT,SCHEMA FROM "ADBUSER03","ADBUSER04" RESTRICT
```

If the REVOKE statement is executed when ADBUSER03 owns schemas but ADBUSER04 does not own schemas, execution of the REVOKE statement is halted on both ADBUSER03 and ADBUSER04.

For details about the REVOKE statement, see *REVOKE (revoke privileges)* in *Definition SQL* in the manual *HADB SQL Reference*.

Important

You cannot revoke the CONNECT privilege or schema definition privilege of an HADB user who has the audit privilege.

11.8 Managing access privileges

This section explains how to manage access privileges.

11.8.1 Granting access privileges to an HADB user

To grant access privileges for a schema object to other HADB users, execute the GRANT statement.

> **Important**
>
> The HADB user who grants access privileges to other users (that is, the HADB user who executes the GRANT statement) needs to have the grant options for those access privileges.

The following examples explain how to grant access privileges to other HADB users.

Example 1:

In this example, the SELECT, INSERT, and EXPORT TABLE privileges for table A.T1 are granted to HADB users ADBUSER01 and ADBUSER02.

```
GRANT SELECT,INSERT,EXPORT TABLE ON "A"."T1" TO "ADBUSER01","ADBUSER02"
```

The HADB user who executes the GRANT statement needs to have the SELECT, INSERT, and EXPORT TABLE privileges with the grant options for table A.T1.

Example 2:

In this example, all access privileges for table A.T1 are granted to HADB user ADBUSER01.

```
GRANT ALL PRIVILEGES ON "A"."T1" TO "ADBUSER01"
```

The HADB user who executes the GRANT statement needs to have all access privileges with the grant options for table A.T1.

> **Note**
>
> If the HADB user who executes the GRANT statement has grant options for only a part of access privileges for table A.T1, executing the preceding GRANT statement grants only the access privileges corresponding to those grant options. For example, if the HADB user who executes the GRANT statement has only the grant options for the SELECT and INSERT privileges, executing the preceding GRANT statement grants only the SELECT and INSERT privileges to HADB user ADBUSER01.

Example 3:

In this example, the SELECT privilege for table A.T1 is granted to all HADB users.

```
GRANT SELECT ON "A"."T1" TO PUBLIC
```

The HADB user who executes the GRANT statement needs to be the owner of table A.T1.

Example 4:

In this example, the SELECT privilege with the grant option for table A.T1 is granted to HADB user ADBUSER01.

```
GRANT SELECT ON "A"."T1" TO "ADBUSER01" WITH GRANT OPTION
```

To grant the access privilege with the grant option, specify WITH GRANT OPTION.

The HADB user who executes the GRANT statement needs to have the SELECT privilege with the grant option for table A.T1.

For details about the GRANT statement, see *GRANT (grant privileges)* in *Definition SQL* in the manual *HADB SQL Reference*.

11.8.2 Checking the access privileges granted to an HADB user

To check which access privileges have been granted to an HADB user, search the SQL_TABLE_PRIVILEGES dictionary table.

The range of HADB users whose access privileges can be checked varies depending on the privileges granted to the HADB user who is checking. For the scope of information in dictionary tables that can be referenced by HADB users, see 2.7.7 Scope of information in dictionary tables and system tables that can be referenced by HADB users.

The following shows an example of checking the access privileges that are granted to HADB users.

Specification example

An HADB user who has the DBA privilege and CONNECT privilege checks the access privileges granted to all created HADB users.

```
SELECT "GRANTOR","GRANTEE","TABLE_SCHEMA","TABLE_NAME",
       "SELECT_PRIVILEGE","INSERT_PRIVILEGE","UPDATE_PRIVILEGE",
       "DELETE_PRIVILEGE","TRUNCATE_PRIVILEGE",
       "REFERENCES_PRIVILEGE","IMPORT_TABLE_PRIVILEGE",
       "REBUILD_INDEX_PRIVILEGE","GET_COSTINFO_PRIVILEGE",
       "EXPORT_TABLE_PRIVILEGE","MERGE_CHUNK_PRIVILEGE",
       "CHANGE_CHUNK_COMMENT_PRIVILEGE","CHANGE_CHUNK_STATUS_PRIVILEGE",
       "ARCHIVE_CHUNK_PRIVILEGE","UNARCHIVE_CHUNK_PRIVILEGE"
  FROM "MASTER"."SQL_TABLE_PRIVILEGES"
```

Explanation:

For example, to check whether the SELECT privilege is granted, see the information displayed in the SELECT_PRIVILEGE column.

- If 'G' is displayed, the SELECT privilege with the grant option is granted for a schema object.
- If 'Y' is displayed, the SELECT privilege with no grant option is granted for a schema object.
- If 'N' is displayed, the SELECT privilege is not granted for a schema object.

The schema name and table identifier of a schema object are displayed in the TABLE_SCHEMA and TABLE_NAME columns respectively.

For details about dictionary table SQL_TABLE_PRIVILEGES, see B.18 Content of SQL_TABLE_PRIVILEGES.

11.8.3 Revoking access privileges that were granted to HADB users

To revoke access privileges that were granted to HADB users, execute the REVOKE statement.

Important

Only the HADB users to which access privileges are granted can revoke those access privileges.

The following examples explain how to revoke access privileges that were granted to HADB users.

Example 1:

In this example, the SELECT privilege for table A.T1 that was granted to HADB user ADBUSER01 is revoked.

```
REVOKE SELECT ON "A"."T1" FROM "ADBUSER01"
```

Example 2:

In this example, all types of access privileges for table A.T1 that were granted to HADB user ADBUSER01 are revoked.

```
REVOKE ALL PRIVILEGES ON "A"."T1" FROM "ADBUSER01" RESTRICT
```

Because RESTRICT is specified in the preceding example, the REVOKE statement returns an error if one of the following conditions is met:

- ADBUSER01 has defined a viewed table that depends on the underlying table A.T1.
- ADBUSER01 has granted an access privilege for table A.T1 to another HADB user (the access privilege for table A.T1 that was granted to ADBUSER01 has a dependent privilege).
- There is a referential constraint that was defined by ADBUSER01 by using table A.T1 as the referenced table.

Example 3:

In this example, the SELECT privilege for table A.T1 that was granted to all HADB users by specifying PUBLIC in the GRANT statement is revoked.

```
REVOKE SELECT ON "A"."T1" FROM PUBLIC
```

For details about the REVOKE statement, see *REVOKE (revoke privileges)* in *Definition SQL* in the manual *HADB SQL Reference*.

Important

- If the access privileges that were revoked have dependent privileges, those dependent privileges are also revoked. For details, see (4) Revoking access privileges in 2.7.5 Access privileges.
- If the SELECT privilege for the underlying table for a viewed table is revoked, all viewed tables that depend on that underlying table are invalidated. If the SELECT privilege with the grant option is revoked, the SELECT privilege that was granted by using the grant option is also revoked. This might cause the relevant viewed tables to be invalidated. For details, see 2.7.6 Access privileges for viewed tables.
- If the REFERENCES privilege for a table is revoked, the referential constraint that was defined by using that REFERENCES privilege is deleted.

11.8.4 Revoking only the grant options for access privileges that were granted to HADB users

To revoke only the grant options for access privileges that were granted to HADB users, execute the REVOKE statement with GRANT OPTION FOR specified.

Important

Only HADB users who have been granted access privileges with the grant options can revoke those grant options for access privileges.

The following example explains how to revoke only the grant options for access privileges that were granted to HADB users.

Example:

In this example, only the grant option for the SELECT privilege for table A.T1 that was granted to HADB user ADBUSER01 is revoked.

```
REVOKE GRANT OPTION FOR SELECT ON "A"."T1" FROM "ADBUSER01"
```

To revoke only the grant option, execute the REVOKE statement with GRANT OPTION FOR specified.

In this case, the SELECT privilege for table A.T1 that was granted to HADB user ADBUSER01 is not revoked.

For details about the REVOKE statement, see *REVOKE (revoke privileges)* in *Definition SQL* in the manual *HADB SQL Reference*.

Important

- If the grant option for a privilege is revoked, all access privileges that were granted by using that grant option are revoked. For details, see (4) Revoking access privileges in 2.7.5 Access privileges.
- If the grant option for the SELECT privilege is revoked, all SELECT privileges that were granted by using that grant option are also revoked. This might cause the relevant viewed tables to be invalidated. For details, see 2.7.6 Access privileges for viewed tables.

11.8.5 Changing the access privileges granted to an HADB user

To change an HADB user's access privileges, you need to revoke the unnecessary access privileges first, and then grant the necessary access privileges.

The procedure for changing access privileges follows.

Procedure

1. Check the access privileges granted to an HADB user.
   Check which access privileges have been granted to an HADB user based on the explanation in 11.8.2 Checking the access privileges granted to an HADB user.
2. Revoke the unnecessary access privileges that were granted to the HADB user.
   Execute the REVOKE statement based on the explanation in 11.8.3 Revoking access privileges that were granted to HADB users.
3. Grant the necessary access privileges to the HADB user.
   Execute the GRANT statement based on the explanation in 11.8.1 Granting access privileges to an HADB user.

11.9 Handling schemas

This section describes handling of schemas.

11.9.1 Defining a schema

To define a schema, you execute the `CREATE SCHEMA` statement.

For details about the `CREATE SCHEMA` statement, see *CREATE SCHEMA (define a schema)* under *Definition SQL* in the manual *HADB SQL Reference*.

11.9.2 Deleting a schema

To delete a schema, you execute the `DROP SCHEMA` statement.

For details about the `DROP SCHEMA` statement, see *DROP SCHEMA (delete a schema)* under *Definition SQL* in the manual *HADB SQL Reference*.

Important

If you delete a schema, all schema objects defined in the schema are also deleted.

If a base table is defined in a schema to be deleted, beforehand, see the explanation in 11.1.15 Deleting base tables. If you delete a schema in which a non-updatable base table is defined, temporary work files created by the interrupted command might remain.

11.10 Handling data DB areas

This section explains the handling of data DB areas and data DB area files.

> **Note**
>
> For details about how to change the data DB area that stores a base table, see 11.1.14 Changing the data DB area that stores a base table.
>
> For details about how to change the data DB area that stores indexes, see 11.3.8 Changing the data DB area that stores indexes.

11.10.1 Adding data DB areas

This subsection explains how to use the `adbmodarea` command to add new data DB areas. For details about the `adbmodarea` command, see *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

A specification example for adding a data DB area is shown below. Only the HADB administrator can add a data DB area.

■ Specification example

A data DB area is added.

```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

When you execute the `adbmodarea` command, you need to create a DB area addition/modification option file in which the `adbaddarea` operand of the DB area addition/modification option is specified.

For details about the `adbaddarea` operand of the DB area addition and modification option, see *Specification format for the adbmodarea command* under *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

If one of the following cases is applicable when you are defining a new base table, B-tree index, text index, or range index, consider adding a data DB area:

- If you are defining a range index

  We recommend that you add a DB area dedicated to the range index.
- If you want to separate a global buffer or data DB area file from the existing data DB area

  Add a data DB area in order to distribute the effect on the existing base table, B-tree index, text index, or range index.
- If there is no data DB area that has a size adequate to store the new base table

> **Note**
>
> **■ Relationship between the added data DB area and allocated global buffer**
>
> If the `-o` option is specified for the `adbbuff` operand in the server definition, a global buffer is automatically allocated to the added data DB area according to the specification in the `-o` option.
>
> Even if the `-o` option is not specified, a global buffer similar to the one that is allocated when the `-o` option is specified is automatically allocated to the added data DB area.

In either case, if the required processing performance is not achieved, specify the global buffer allocated to the added DB area in the adbbuff operand. For details, see 7.2.11 Operands and options related to global buffers (command format) and 8.5.2 Modifying the server definition.

Also, if necessary, re-evaluate the value specified for the adbbuff operand. For details, see 13.2.2 Reducing the SQL statement execution time.

11.10.2 Deleting a data DB area

This subsection explains how to use the adbmodarea command to delete a data DB area. For details about the adbmodarea command, see *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

A specification example for deleting a data DB area is shown below. Only the HADB administrator can delete a data DB area.

■ **Specification example**

A data DB area is deleted.

```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

When executing the adbmodarea command, you need to create a DB area addition/modification option file in which the adbrmarea operand of the DB area addition/modification option is specified.

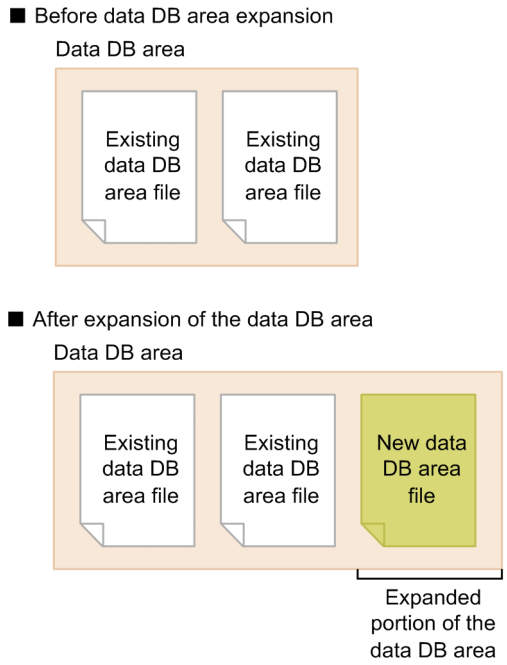
For details about the adbrmarea operand of the DB area addition and modification option, see *Specification format for the adbmodarea command* under *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

11.10.3 Expanding a data DB area (adding a data DB area file)

This subsection explains how to use the adbmodarea command to expand an existing data DB area. For details about the adbmodarea command, see *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

When you want to expand the capacity of an existing data DB area by adding a new data DB area file, execute the adbmodarea command. The following figure shows expansion of a data DB area.

Figure 11-40: Overview of expansion of a data DB area



A specification example for expanding a data DB area is shown below. Only the HADB administrator can expand a data DB area.

■ Specification example

A data DB area is expanded.

```
adbmodarea /HADB/server/conf/adbmodarea.opt
```

When executing the `adbmodarea` command, you need to create a DB area addition/modification option file in which the `adbexpandarea` operand of the DB area addition/modification option is specified.

For details about the `adbexpandarea` operand of the DB area addition and modification option, see *Specification format for the adbmodarea command* under *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

11.10.4 Re-initializing a data DB area

This subsection describes how to re-initialize a specific data DB area.

> **Note**
>
> After data DB areas are created (initialized) during database initialization, you can delete a specific data DB area, and then can add a data DB area with the same name as the deleted one. This operation is called *re-initialization of a data DB area*.

To re-initialize a specific data DB area, you execute the `adbmodarea` command. However, you cannot use the `adbmodarea` command to complete re-initialization in one operation. Therefore, use the following procedure to re-initialize a data DB area.

Procedure

1. Delete the data DB area that you want to re-initialize.
   For details about how to delete a data DB area, see 11.10.2 Deleting a data DB area.
2. Add a data DB area again with the same name as the deleted one.
   For details about how to add a data DB area, see 11.10.1 Adding data DB areas.

For details about the `adbmodarea` command, see *adbmodarea (Add and Change DB Areas)* in the manual *HADB Command Reference*.

Important

Do not use the `adbinit` command when you re-initialize a specific data DB area.

Because the `adbinit` command initializes all DB areas, use the command when you perform database initial setup. If you execute the `adbinit` command when data DB areas already exist, all DB areas, including the ones other than data DB areas, are deleted.

11.10.5 Changing the storage location of data DB area files

This subsection explains how to change the storage location of data DB area files that make up a data DB area.

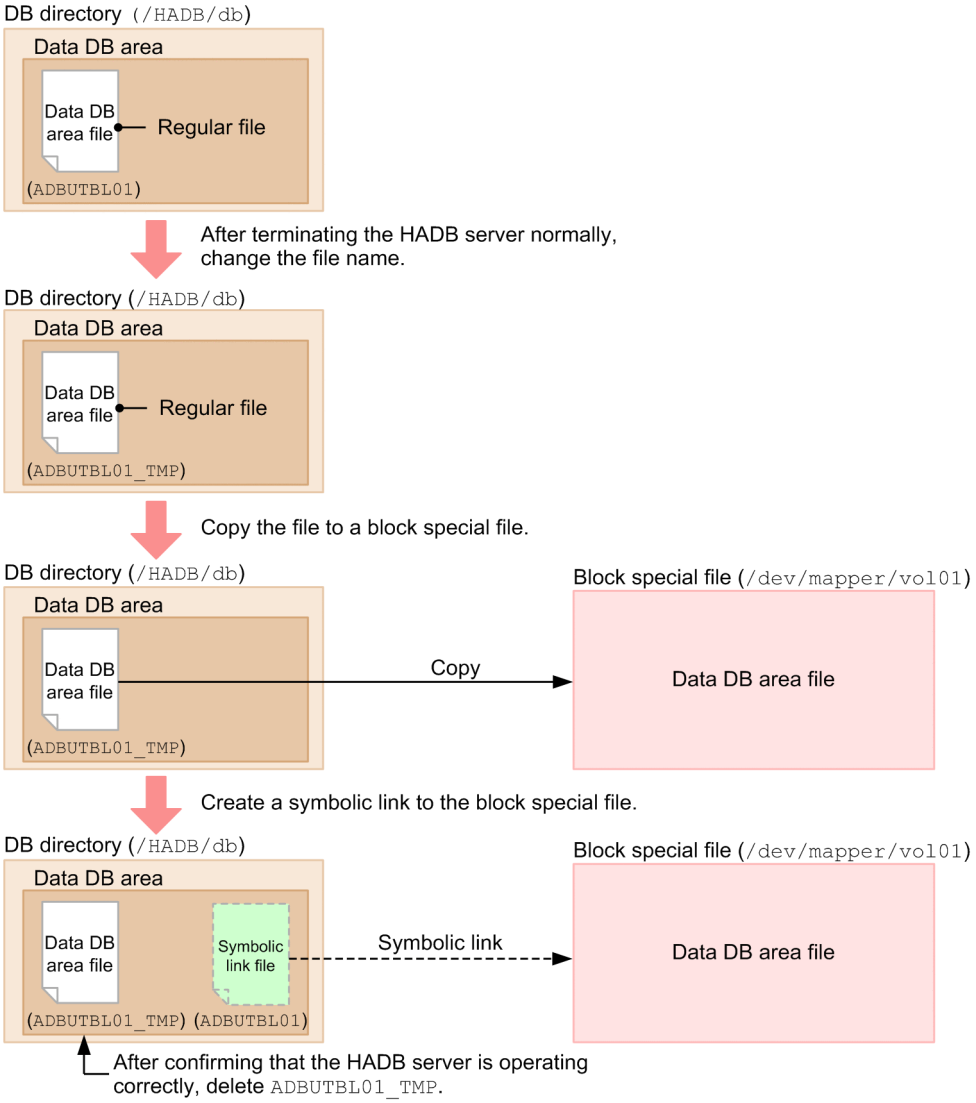
Important

When one data DB area consists of multiple data DB area files, make sure that regular files and block special files are not intermixed as target files when you change the storage location of the data DB area files. Use only one of these types of files.

(1) Changing a data DB area file from a regular file to a block special file

Prepare a block special file of the required size, and then change the data DB area file from a regular file to a block special file.

Figure 11-41: How to change a data DB area file to a block special file



Procedure

1. Terminate the HADB server normally with the `adbstop` command.

   For details about the `adbstop` command, see *adbstop (Terminate the HADB Server)* in the manual *HADB Command Reference*.

2. Rename the data DB area file.

   Rename the data DB area file that is to be changed from a regular file to a block special file. Use the OS's `mv` command to rename the file. The following shows an example:

   • Command execution example

   ```
   mv /HADB/db/ADBUTBL01 /HADB/db/ADBUTBL01_TMP
   ```

   This example changes the file name from `ADBUTBL01` to `ADBUTBL01_TMP`.

3. Copy the renamed data DB area file to a block special file.

   Copy the data DB area file that has been renamed in step 2 to a block special file. Use the OS's `dd` command to copy the data DB area file. The following shows an example:

   • Command execution example

```
dd if=/HADB/db/ADBUTBL01_TMP of=/dev/mapper/vol01 bs=524288
```

This example copies the data DB area file (/HADB/db/ADBUTBL01_TMP), which is a regular file, to a block special file (/dev/mapper/vol01).

4. Create a symbolic link to the block special file using the original file name.

   Create a symbolic link to the block special file to which the data DB area file was copied in step 3. In this case, you need to create the symbolic link to the block special file in the original data DB area file name that was renamed in step 2. Use the OS's ln command to create a symbolic link. The following shows an example:

   • Command execution example

```
ln -s /dev/mapper/vol01 /HADB/db/ADBUTBL01
```

   This example creates a symbolic link to the block special file (/dev/mapper/vol01) using the original file name (/HADB/db/ADBUTBL01).

5. Start the HADB server with the adbstart command.

   For details about the adbstart command, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

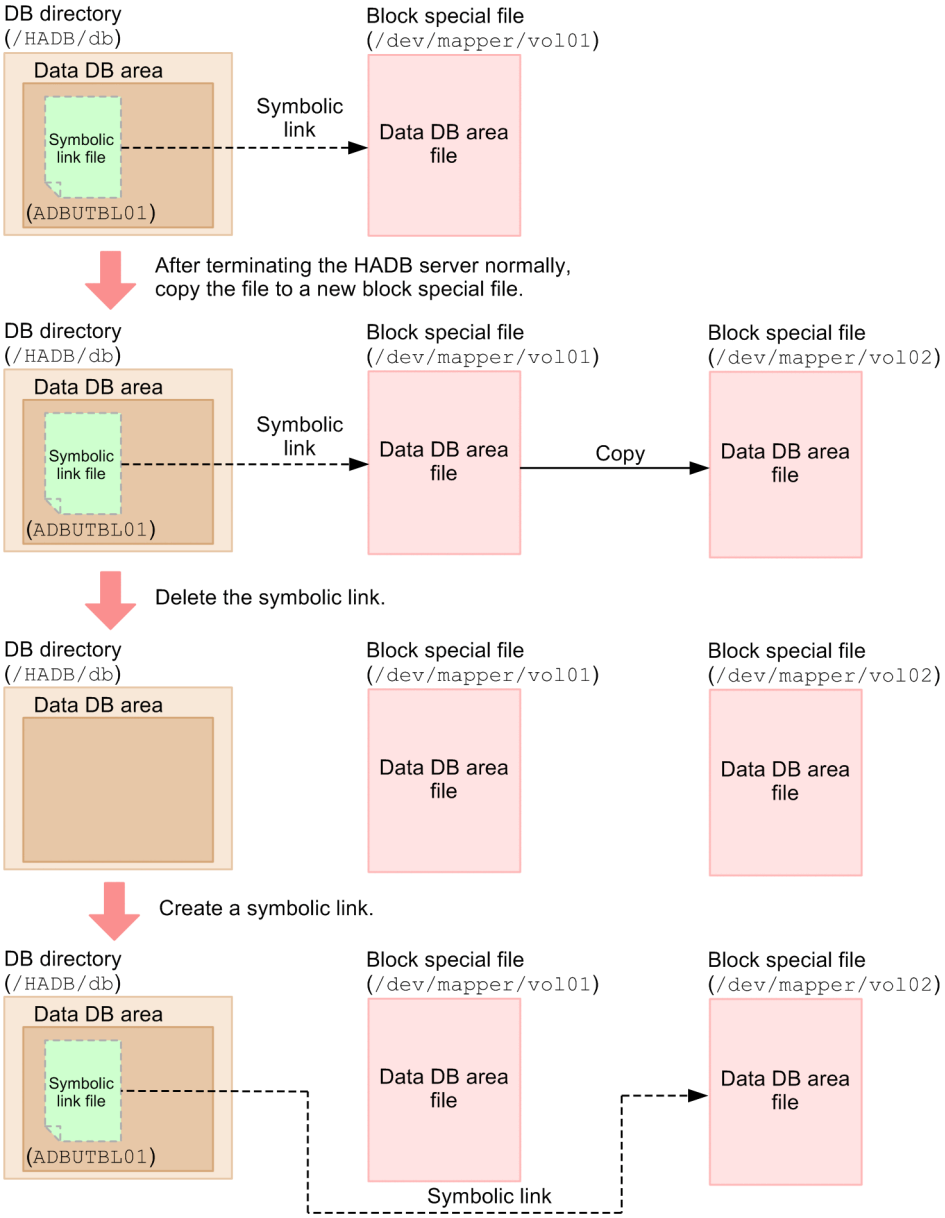
6. After checking the operation of the HADB server, delete the unneeded regular file.

   Confirm that changing the data DB area file from a regular file to a block special file has not resulted in any problems. If there are no problems, the data DB area file (regular file) that was renamed in step 2 is no longer needed. Use an OS command to delete it.

(2) Changing the storage location of a data DB area file (block special file)

Prepare a block special file of the required size, and then change the storage location of the data DB area file (to a new block special file).

Figure 11-42: Changing the storage location of a data DB area file (block special file



Procedure

1. Terminate the HADB server normally with the `adbstop` command.

   For details about the `adbstop` command, see *adbstop (Terminate the HADB Server)* in the manual *HADB Command Reference*.

2. Copy the entity of the data DB area file to a new block special file (change the storage location).

   Copy the entity of the data DB area file (target block special file of the symbolic link) whose storage area is to be changed to a new block special file. Use the OS's `dd` command to copy the entity of the data DB area file. The following shows an example:

   • Command execution example

   ```
   dd if=/dev/mapper/vol01 of=/dev/mapper/vol02 bs=524288
   ```

   This example copies the entity of the data DB area file (`/dev/mapper/vol01`) to another block special file (`/dev/mapper/vol02`).

3. Delete the symbolic link to the original block special file at the storage location before the change.

   To change the storage location of the data DB area file, delete the symbolic link to the source block special file from which data was copied in step 2. Use the OS's `rm` command to delete the source symbolic link file. The following shows an example:

   • Command execution example

   ```
   rm /HADB/db/ADBUTBL01
   ```

   This example deletes the source symbolic link file (`/HADB/db/ADBUTBL01`) to delete the symbolic link to the block special file (`/dev/mapper/vol01`) at the storage location before the change.

4. Create a symbolic link to the block special file at the storage location after the change.

   To change the storage location of the data DB area file, create a symbolic link to the target block special file to which data was copied in step 2. In this case, you need to use the symbolic link file name that was deleted in step 3 to create a symbolic link to the block special file. Use the OS's `ln` command to create the symbolic link. The following shows an example:

   • Command execution example

   ```
   ln -s /dev/mapper/vol02 /HADB/db/ADBUTBL01
   ```

   This example creates a symbolic link to the block special file (`/dev/mapper/vol02`) using the name of the deleted symbolic link file (`/HADB/db/ADBUTBL01`).

5. Start the HADB server with the `adbstart` command.

   For details about the `adbstart` command, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

6. Check the operation of the HADB server.

   Confirm that changing the storage location of the data DB area file (block special file) has not resulted in any problems.

11.10.6 Securing free space in a data DB area

This subsection describes how to secure free space in a data DB area.

> **Important**
>
> Before you begin to secure free space in a data DB area, check whether expansion of the data DB area is possible. For details about how to expand a data DB area, see 11.10.3 Expanding a data DB area (adding a data DB area file).

If expansion of the data DB area is impossible, use Action 1 or 2 (explained later) to secure free space in the data DB area. If you still cannot secure free space in the data DB area by performing Action 1 or 2, perform Action 3.

Action 1

Archive chunks. Compress the data in chunks to secure free space in the data DB area.

For details, see (1) Securing free space in a data DB area (compressing data).

Action 2

Reorganize the base table. Secure free space in the data DB area by deleting invalid areas or compressing data.

For details, see (2) Securing free space in a data DB area (reorganizing the base table).

Action 3

Delete unnecessary data. Delete unnecessary data to secure free space in the data DB area.

For details, see (3) Securing free space in a data DB area (deleting unnecessary data).

(1) Securing free space in a data DB area (compressing data)

If you want to secure free space in a data DB area that contains an archivable multi-chunk table, you can try to archive chunks. If you use the `adbarchivechunk` command, which archives chunks, you might be able to increase the amount of free space in the data DB area by compressing the data in the chunks.

> **Note**
>
> For details about how to archive chunks, see 2.15.1 Overview of the chunk archiving function.

The following shows the procedure for compressing the data in chunks.

Procedure

1. Check the schema name and table identifier of the archivable multi-chunk table.

   Execute the `SELECT` statement shown later to check the schema name and table identifier of the archivable multi-chunk table stored in the data DB area.

   Specification example

   ```
   SELECT "MASTER"."SQL_DIV_TABLE"."TABLE_SCHEMA",
          "MASTER"."SQL_DIV_TABLE"."TABLE_NAME"
     FROM "MASTER"."SQL_DIV_TABLE","MASTER"."SQL_TABLES"
       WHERE "DBAREA_ID" = (SELECT "DBAREA_ID"
                              FROM "MASTER"."SQL_DBAREAS"
                                WHERE "DBAREA_NAME" = ?)
         AND "MASTER"."SQL_DIV_TABLE"."TABLE_SCHEMA"
               = "MASTER"."SQL_TABLES"."TABLE_SCHEMA"
         AND "MASTER"."SQL_DIV_TABLE"."TABLE_NAME"
               = "MASTER"."SQL_TABLES"."TABLE_NAME"
         AND "IS_ARCHIVABLE" = 'Y'
   ```

   If you performed the preceding `SELECT` statements by using the `adbsql` command, you will be prompted to enter the input data for the dynamic parameters. In response to the prompt, enter the DB area name of the data DB area in which you want to secure free space.

2. Determine the chunks to be archived.

   Archive chunks to secure free space in the data DB area. Of the *chunks that are not in the archived state* in the archivable multi-chunk table, determine the chunks to be archived.

   Note that retrieval from archived chunks takes a longer time than retrieval from non-archived chunks. Therefore, we recommend that you archive chunks for which data retrieval occurs less frequently. For example, if retrieval for later created chunks occurs more frequently than retrieval for earlier created chunks, archive earlier created chunks.

   To check the information about the chunks to be archived, execute the `adbdbstatus` command with the `-d used` option specified, and then check the *information about the usage of DB areas, tables, and indexes* in the output results. Alternatively, use the `SELECT` statement to search system table `STATUS_CHUNKS`. For details, see (1) Executing the adbdbstatus command or (3) Searching the STATUS_CHUNKS system table in 11.4.8 Checking the chunk status and the number of chunks created.

3. Check the size of free space that can be secured by archiving chunks.

   By archiving chunks, you can secure free space (of the size that those chunks occupy) in the data DB area. To check the size of free space that can be secured by archiving chunks, execute the `adbdbstatus` command with the `-d`

used option specified. When the *information about the usage of DB areas, tables, and indexes* is output, check the content of Used_segments and Used_pages based on Chunk_ID.

For details, see (1) Executing the adbdbstatus command in 11.4.8 Checking the chunk status and the number of chunks created.

4. Archive chunks to compress data.

   Execute the adbarchivechunk command to archive chunks (compress the data in chunks). For details about how to archive chunks, see 11.4.17 Archiving chunks (when using an archivable multi-chunk table).

(2) Securing free space in a data DB area (reorganizing the base table)

If you have repeatedly performed any of the following operations for a base table, try reorganizing the base table. You might be able to increase the amount of free space in the data DB area.

- Using an SQL statement to perform update or deletion of rows for a row store table
- Using an SQL statement to perform addition, update, or deletion of rows for a column store table
- Using the adbimport command to perform background import for a multi-chunk table

Note

Reasons why the free space in a data DB area can be increased by reorganizing the base table are as follows:

- Repeated update or deletion of rows by using an SQL statement or repeated execution of background import by using the adbimport command might increase invalid row data or non-reusable free space. When you reorganize the base table, such invalid row data or non-reusable free space will be released. This might increase the amount of free space in the data DB area.
- If an SQL statement is used to add rows to a column store table, the rows will be added in row store format rather than in column store format. Data added in row store format is not compressed. When you reorganize the base table or indexes, data will be added in column store format. This might increase the amount of free space in the data DB area because the data added in column store format is compressed.

The following shows the procedure for reorganizing a base table.

Procedure

1. Check the size of free space that can be secured.

   Use the following calculation expression to estimate the size of free space that can be secured by reorganizing the base table.

   Expression for calculating the size of free space that can be secured by reorganizing the base table (megabytes)

```
Size of free space that can be secured by reorganizing the base table =
    space-usage-for-base-table-before-reorganization - space-usage-for-base-tabl
e-after-reorganization
```

   space-usage-for-base-table-before-reorganization (megabytes)

   You can check *space-usage-for-base-table-before-reorganization* by referring to (2) Checking the usage of a base table in 10.9.2 Checking the status and usage of a base table.

space-usage-for-base-table-after-reorganization (megabytes)

Estimate this value by referring to 5.8 Estimating the size of the data DB area. Note that the value obtained from the calculation formula in the section or subsection you reference is in kilobytes. Therefore, you must convert the value in megabytes.

Also note that when a base table is reorganized, the indexes defined for the base table are also reorganized. Use the following calculation expression to estimate the size of free space that can be secured by reorganizing indexes.

Expression for calculating the size of free space that can be secured by reorganizing indexes (megabytes)

```
Size of free space that can be secured by reorganizing indexes =
    space-usage-for-indexes-before-reorganization - space-usage-for-indexes-afte
r-reorganization
```

space-usage-for-indexes-before-reorganization (megabytes)

You can check *space-usage-for-indexes-before-reorganization* by referring to the following subsections:

- (3) Checking the usage of a B-tree index in 10.9.3 Checking the status and usage of a B-tree index.
- (2) Checking the usage of a text index in 10.9.4 Checking the status and usage of a text index.
- (2) Checking the usage of a range index in 10.9.5 Checking the status and usage of range indexes.

space-usage-for-indexes-after-reorganization (megabytes)

Estimate this value by referring to 5.8 Estimating the size of the data DB area. Note that the value obtained from the calculation formula in the section or subsection you reference is in kilobytes. Therefore, you must convert the value in megabytes.

2. Reorganize the base table.

Reorganize the base table by referring to the following subsections. Note that when the base table is reorganized, the indexes defined for the base table are also reorganized. You do not need to obtain the data storage efficiency.

- 11.1.10 Reorganizing a single-chunk table
- 11.4.14 Reorganizing a multi-chunk table: Chunk-based reorganization
- 11.4.15 Reorganizing a multi-chunk table: Reorganization of an entire table
- 11.4.16 Reorganizing a multi-chunk table: Reorganization using a sample shell script

Important

- If the target table is a multi-chunk table that contains unnecessary chunks, try deleting the unnecessary chunks before performing reorganization. For details about how to delete unnecessary chunks, see (3) Securing free space in a data DB area (deleting unnecessary data).
- While a base table is being reorganized, the data in the base table is temporarily deleted. Therefore, you cannot view data in a base table that is being reorganized.
- If the target base table stores a large amount of data, the size of data to be exported during reorganization becomes large. Also, the time required for reorganization becomes longer. If there is not enough space for exporting data or time for reorganization, try securing free space in the data DB area by using another method.

(3) Securing free space in a data DB area (deleting unnecessary data)

This subsection describes how to secure free space in a data DB area by deleting unnecessary data.

If you cannot secure free space in a data DB area by using the methods described in the following subsections, try deleting unnecessary data.

- (1) Securing free space in a data DB area (compressing data)
- (2) Securing free space in a data DB area (reorganizing the base table)

You can secure free space in a data DB area by deleting the following types of unnecessary data:

- Unnecessary indexes
- Unnecessary chunks in a multi-chunk table

Important

Deleting unnecessary data carries the following risks:

- Loss of data due to an incorrect operation
- Degradation in performance of SQL statements and commands

Therefore, before performing such a deletion procedure, read the notes that apply to the procedure.

■ If there are unnecessary indexes in the data DB area in which free space is to be secured

You can secure free space by deleting unnecessary indexes. The following shows the procedure.

Procedure:

1. Determine the indexes to be deleted.

   When you determine the indexes to be deleted, we recommend that you check the usage status of indexes in the *access path statistical information*.

   If SQL tracing is enabled, the *access path statistical information* is output each time an SQL statement is executed, and you can check whether the SQL statement used indexes from that information. Delete indexes that are not used by SQL statements that need to be executed.

   For details about SQL tracing, see 10.11 Running SQL tracing. For details about access path statistical information, see 10.11.3 Examples of output of and output items for access path statistical information.

2. Check the size of free space that can be secured.

   You can increase the free space in the data DB area by the size of space that is used for the indexes to be deleted. You can check the space usage for indexes by referring to the following subsections:

   - (3) Checking the usage of a B-tree index in 10.9.3 Checking the status and usage of a B-tree index.
   - (2) Checking the usage of a text index in 10.9.4 Checking the status and usage of a text index.
   - (2) Checking the usage of a range index in 10.9.5 Checking the status and usage of range indexes.

3. Delete the indexes.

   Delete unnecessary indexes by referring to 11.3.9 Deleting an index.

Notes

- Deleting indexes might degrade the performance of SQL statements that use the indexes.
- In case you need to later redefine indexes that are to be deleted, obtain the definition information that is required to do so before deleting them. For details about how to obtain definition information, see (29) Finding out index definition information in B.22 Searching a dictionary table.

■ If there is a multi-chunk table with unnecessary chunks in the data DB area in which free space is to be secured

You can secure free space by using the `PURGE CHUNK` statement to delete unnecessary chunks in a multi-chunk table. The following shows the procedure.

Procedure:

1. Determine the chunks to be deleted.

   Check the status of each chunk in the multi-chunk table to determine the chunks to be deleted. For example, you can first delete the following types of chunks, which are more likely to be unnecessary:

   - Deletion-pending chunks
   - Unarchived chunks in wait status
   - Chunks that are seldom searched (such as chunks created a long time ago)

   To check the information about chunks, execute the `adbdbstatus` command with the `-d used` option specified, and then check the *information about the usage of DB areas, tables, and indexes* in the output results. Alternatively, use the `SELECT` statement to search system table `STATUS_CHUNKS`. For details, see (1) Executing the adbdbstatus command or (3) Searching the STATUS_CHUNKS system table in 11.4.8 Checking the chunk status and the number of chunks created.

2. Check the size of free space that can be secured.

   You can increase the free space in the data DB area by the size of space that is used for the chunks to be deleted. To check the size of space used for chunks, execute the `adbdbstatus` command with the `-d used` option specified. When the *information about the usage of DB areas, tables, and indexes* is output, check the content of `Used_segments` and `Used_pages` based on `Chunk_ID`.

   For details, see (1) Executing the adbdbstatus command in 11.4.8 Checking the chunk status and the number of chunks created.

3. Delete chunks.

   Delete unnecessary chunks by referring to 11.4.6 Deleting data in units of chunks.

Notes

- If the chunks to be deleted contain data that might need to be accessed later, export the chunks before deletion. For details about how to export the data in a chunk, see 11.4.5 Exporting data in units of chunks.
- You cannot increase the free space in the data DB area by deleting archived chunks.

11.11 Handling the work table DB area

This section explains the handling of the work table DB area and work table DB area files.

11.11.1 Changing the storage location of a work table DB area file

If a work table DB area file is a block special file, you can use the `adb_blk_path_wrk` operand in the server definition to change its storage location. You can also change a work table DB area file from a regular file to a block special file. The following procedure explains how to change the storage location of the work table DB area file.

Procedure

1. Terminate the HADB server normally with the `adbstop` command.

   For details about the `adbstop` command, see *adbstop (Terminate the HADB Server)* in the manual *HADB Command Reference*.

2. Specify a new block special file for the work table DB area file in the `adb_blk_path_wrk` operand in the server definition.

   For details about the `adb_blk_path_wrk` operand in the server definition, see the description of the adb_blk_path_wrk operand in 7.2.1 Operands related to system configuration (set format).

3. Start the HADB server with the `adbstart` command.

   For details about the `adbstart` command, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

11.11.2 Reducing the size of a work table DB area file

When you execute the SQL statement for creating a work table, the size of the work table might become very large depending on the contents of the SQL statements. If a work table DB area file is a regular file, the size of the regular file might become very large depending on the capacity of the work table, resulting in a shortage of available disk space.

If the size of a work table DB area file becomes too large, you can delete it. For details about how to delete work table DB area files, see (3) Reducing the size of the work table DB area files in 15.3.1 When a free space shortage is caused by an increase in the size of the DB area files.

The work table DB area files are stored under the DB directory (`$DBDIR/ADBWRK`).

> **Important**
>
> When the size of the work table DB area file becomes too large, evaluate the contents of the SQL statements before you execute them.

11.12 Performing operations on buffers

This section explains buffer operations.

11.12.1 Changing the local work table buffer

This subsection explains how to change the number of pages in the local work table buffer while the HADB server is running.

You can use the adbmodbuff command to change the number of pages in the local work table buffer specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the server definition.

The following is the procedure for using the adbmodbuff command to change the number of pages in the local work table buffer.

Procedure

1. Check the value specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the server definition.
   Use the adbls -d lbuf command to check the number of pages in the local work table buffer specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the server definition.
   The following shows an execution example of the adbls -d lbuf command.
   Command specification example

   ```
   adbls -d lbuf
   ```

   When the adbls -d lbuf command is executed, an output result is displayed such as that shown in the following.
   Example of output result of the executed command

   ```
   CNUMBER    PAGE_SIZE PAGE_NUM
   0          256       128 S
   1          256       128 S
   ```

   The number of pages in the local work table buffer specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the server definition is output under PAGE_NUM in the row where CNUMBER is 0.
2. Change the number of pages in the local work table buffer.
   Based on the output you checked in step 1, use the adbmodbuff command to change the number of pages in the local work table buffer specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the server definition.

In this way, you can change the number of pages in the local work table buffer specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the server definition while the HADB server is running.

Note that the content changed by the adbmodbuff command is lost when the HADB server terminates. If you want to retain the changed content for use during the next startup, incorporate the changed content into the server definition. For details about how to modify the server definition, see 8.5.2 Modifying the server definition.

In the following cases, you cannot change the number of pages in the local work table buffer by using the adbmodbuff command:

- **When the adb_dbbuff_wrktbl_clt_blk_num operand in the client definition is specified**
  Because the client definition takes precedence over the server definition, you cannot change the number of pages in the local work table buffer that are used even if you execute the adbmodbuff command.

- **When the export option `adb_export_wrktbl_blk_num` is specified**

  Because the export option takes precedence over the server definition, you cannot change the number of pages in the local work table buffer that are used even if you execute the `adbmodbuff` command.

> **Note**
>
> - For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition, see the description of the adb_dbbuff_wrktbl_clt_blk_num operand in 7.2.2 Operands related to performance (set format).
> - For details about the `adbmodbuff` command, see *adbmodbuff (Change Buffer)* in the manual *HADB Command Reference*.
> - For details about the `adbls -d lbuf` command, see *adbls -d lbuf (Display Local Work Table Buffer Information)* in the manual *HADB Command Reference*.

11.13 Checking client group settings

This section explains how to check information about the client groups and command group that have been set by using the client-group facility.

11.13.1 Identifying the group to which the HADB client that executed an application belongs

To identify the group to which the HADB client that executed an application belongs, execute the `adbls -d cnct` command.

From the information displayed in the `CLIENT_GROUP` column in the execution result, you can identify the group to which the HADB client that executed the application belongs. If the HADB client that executed the application does not belong to any group, nothing is displayed in the `CLIENT_GROUP` column.

For details about the `adbls -d cnct` command, see *adbls -d cnct (Display the Connection Status)* in the manual *HADB Command Reference*.

11.13.2 Identifying the groups to which the HADB clients using real threads belong

Execute the `adbls -d thd` command to identify the groups to which the HADB clients using real threads belong.

Check the information displayed in the `CLIENT_GROUP` column in the execution results to identify the groups to which the HADB clients using real threads belong. For an HADB client that does not belong to any group, nothing is displayed in the `CLIENT_GROUP` column.

For details about the `adbls -d thd` command, see *adbls -d thd (Display the Thread Status)* in the manual *HADB Command Reference*.

11.13.3 Checking the numbers of connections and processing real threads set for each group

Executing the `adbls -d cltgrp` command enables you to check the following:

- Maximum number of concurrent connections that is set for each group
- Guaranteed minimum number of concurrent connections that is set for each group
- Number of current connections to the HADB server for each group
- Maximum number of processing real threads that is set for each group
- Guaranteed minimum number of processing real threads that is set for each group
- Number of processing real threads that are being used by each group
- Trigger value (that is set for each group) for outputting the warning message regarding the maximum number of concurrent connections

- Trigger value (that is set for each group) for resetting the status indicating that the warning message regarding the maximum number of concurrent connections has been output

For details about the `adbls -d cltgrp` command, see *adbls -d cltgrp (Display Information on Client Groups and Command Groups)* in the manual *HADB Command Reference*.

11.14 Operating centralized management of client definitions

This section describes how to use the function for centrally managing client definitions, for the HADB server to collectively manage the client definitions managed by individual HADB clients.

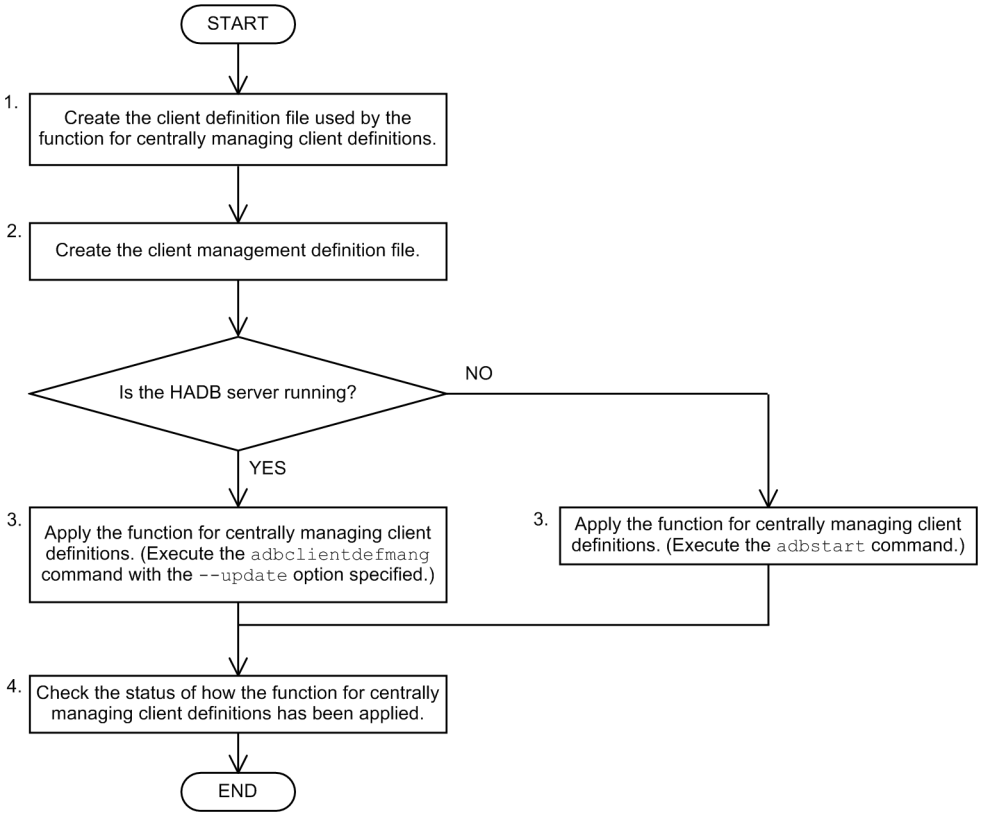
11.14.1 Flow of using the function for centrally managing client definitions

This subsection describes the flow of using the function for centrally managing client definitions.

(1) Flow of newly using the function for centrally managing client definitions

The following figure shows the flow of newly using the function for centrally managing client definitions.

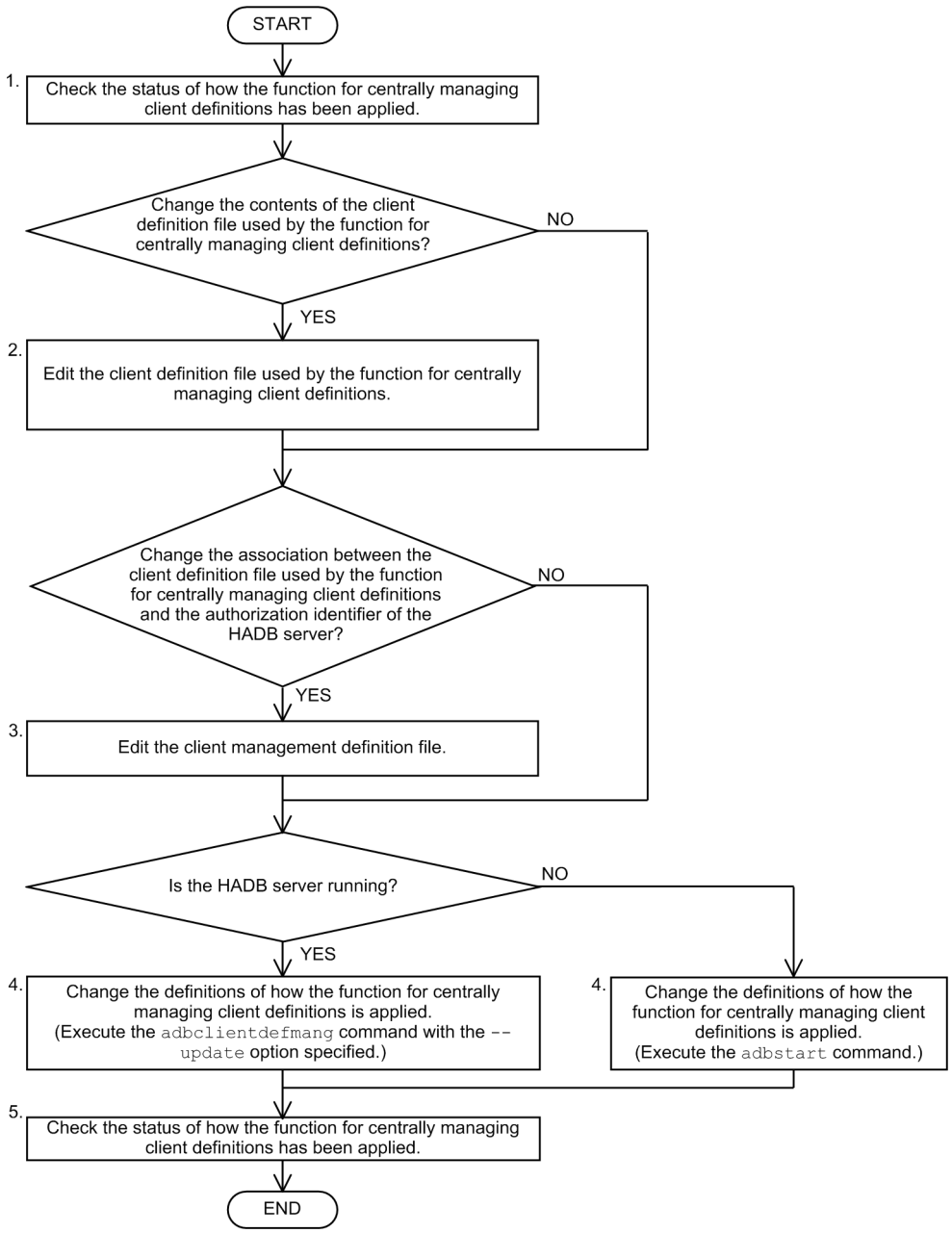
Figure 11-43: Flow of newly using the function for centrally managing client definitions



(2) Flow of changing the definitions of how the function for centrally managing client definitions is applied

The following figure shows the flow of changing the definitions of how the function for centrally managing client definitions is applied.

Figure 11-44: Flow of changing the definitions of how the function for centrally managing client definitions is applied



11.14.2 Files required for the function for centrally managing client definitions

You need to create the following files to use the function for centrally managing client definitions.

Files necessary to be created on the HADB server

- Client definition file used by the function for centrally managing client definitions
- Client management definition file

Files necessary to be created on HADB clients

- Client definition files managed by individual HADB clients

(1) Client definition file used by the function for centrally managing client definitions

The client definition file used by the function for centrally managing client definitions specifies the values for the operands in the client definition that you want to apply to the authorization identifier managed by this function. The applicable authorization identifier connects to the HADB server, according to the definitions in the client definition file used by the function for centrally managing client definitions.

The template of the client definition file used by the function for centrally managing client definitions (`client.def`) is stored in the server directory `$ADBDIR/sample/conf`. Copy the template, and in the new file, specify the individual operands for the client definition. The specification rules for the client definition are the same as ones for the server definition. For details about the specification rules for the server definition, see 7.3 Syntax rules for the server definition.

Note

- For details about individual operands in the client definition, see *Contents of operands in the client definition* in the *HADB Application Development Guide*.
- You do not have to specify the operands in the client definitions that are not applied to the function for centrally managing client definitions. For details about individual operands in the client definitions that are not applied to this function, see *Notes about using the function for centrally managing client definitions* in the *HADB Application Development Guide*.

Important

The default values are assumed for the omitted operands in client definitions. Therefore, if you want to apply the definitions in the client definition file managed by an HADB client, specify the same definitions also in the client definition file used by the function for centrally managing client definitions.

After creating the client definition file to be used by the function for centrally managing client definitions, store the file in the server directory `$ADBDIR/conf`. You can rename the client definition file from the name of the template (`client.def`).

If you want to apply the definitions in different client definition files to multiple authorization identifiers, create as many client definition files as needed that are used by the function for centrally managing client definitions.

(2) Client management definition file

In the client management definition file, the `adbclientmang` operand in the client-managing definition is specified. For the `adbclientmang` operand, specify the following two items to associate them.

Items specified for the `adbclientmang` operand in the client-managing definition:

- Name of the client definition file used by the function for centrally managing client definitions
- Authorization identifier of the HADB user for which you want to apply the definitions in the client definition file

The template of the client management definition file (`adbclientdefmang.def`) is stored in the server directory `$ADBDIR/sample/conf`. Copy the template, and in the new file, specify the `adbclientmang` operand in the client-managing definition.

After creating the client management definition file, store the file in the server directory $ADBDIR/conf. Keep the file name the same as the template (adbclientdefmang.def).

In the following cases, the function for centrally managing client definitions is not applied:

- Nothing is specified in the client management definition file. (The file is an empty file.)
- All definitions in the client management definition file are commented out.

In these cases, the definitions specified in the client definition file managed by the HADB client are applied.

(a) Specification format of the client-managing definition

```
{{
  [adbclientmang
     -f client-definition-file-used-by-the-function-for-centrally-managing-client-def
initions
     -i authorization-identifier[,authorization-identifier]...]
}}
```

The specification rules for the client-managing definition are the same as ones for the server definition. For details about the specification rules for the server definition, see 7.3 Syntax rules for the server definition.

(b) Explanation of specification format of the client-managing definition

●-f *client-definition-file-used-by-the-function-for-centrally-managing-client-definitions*

~<character string> ((1 to 255 bytes))

Specify the name of the client definition file used by the function for centrally managing client definitions.

For this option, specify only the name of the client definition file that is stored in the server directory $ADBDIR/conf. A specification example follows.

Specification example (when the client definition file name is client01.def)

```
-f client01.def
```

You can also specify the same file name for the -f option of multiple adbclientmang operands. A specification example follows. Specification example 1 and 2 are both dealt with as the same definition.

Specification example 1

```
adbclientmang -f client01.def -i ADBUSER01,ADBUSER02
```

Specification example 2

```
adbclientmang -f client01.def -i ADBUSER01
adbclientmang -f client01.def -i ADBUSER02
```

If one of the following conditions applies to the content specified for the -f option, an error occurs:

- The specified file does not exist under the $ADBDIR/conf directory.
- A directory name is specified in addition to a file name.
- There is a mistake in the contents of the client definition file specified for this option.

●-i *authorization-identifier*

~<character string> ((1 to 100 bytes))

Specify the authorization identifier of the HADB user who connects to the HADB server by applying the definitions in the client definition file specified for the -f option.

The authorization identifier specified for this option must be 1 to 100 bytes long. The double quotation marks enclosing the authorization identifier are excluded from the length of the authorization identifier.

For this option, you can specify multiple authorization identifiers by delimiting them with a comma. A specification example follows.

Specification example (when the authorization identifiers of the HADB users are ADBUSER01 and ADBUSER02)

```
-i ADBUSER01,ADBUSER02
```

Important

If there is a lowercase letter in the character string for the authorization identifier, be sure to check the specification rules for authorization identifier. For details about the specification rules for authorization identifiers, see 9.4.2 Authorization identifier specification rules.

If one of the following conditions applies to the content specified for the `-i` option, an error occurs:

- The same authorization identifier is repeatedly specified for the `-i` options.
- The same authorization identifier is specified for the `-i` options of multiple `adbclientmang` operands.
- The total number of authorization identifiers specified for all `-i` options exceeds 30,000.

(3) Client definition files managed by HADB clients

Even when you use the function for centrally managing client definitions, you need to create the client definition files that are managed by individual HADB clients.

Note

For details about how to create the client definition file managed by an HADB client, see *How to create a client definition* in the *HADB Application Development Guide*.

For the client definition file managed by an HADB client, specify the operands in the client definition to which the function for centrally managing client definitions is not applied. Such operands include the host name and port number of the connection-target HADB server. For details about individual operands in the client definitions that are not applied to the function for centrally managing client definitions, see *Notes about using the function for centrally managing client definitions* in the *HADB Application Development Guide*.

For the client definition file managed by an HADB client, you do not have to specify the operands in the client definition to which the function for centrally managing client definitions is applied.

Specify the operands in the client definitions to which the function for centrally managing client definitions is applied, in the client definition file used by the function for centrally managing client definitions. If an operand is omitted, the default value is assumed.

11.14.3 Newly using the function for centrally managing client definitions

The following describes the procedure for newly using the function for centrally managing client definitions.

For an explanation of the flow of newly using the function for centrally managing client definitions, see (1) Flow of newly using the function for centrally managing client definitions in 11.14.1 Flow of using the function for centrally managing client definitions.

Procedure:

1. Create the client definition file used by the function for centrally managing client definitions.

   Create the client definition file used by the function for centrally managing client definitions. Store the created client definition file in the server directory `$ADBDIR/conf`. For details, see (1) Client definition file used by the function for centrally managing client definitions in 11.14.2 Files required for the function for centrally managing client definitions.

2. Create the client management definition file.

   Create the client management definition file. Store the created client management definition file in the server directory `$ADBDIR/conf`. Keep the file name the same as the template (`adbclientdefmang.def`). For details, see (2) Client management definition file in 11.14.2 Files required for the function for centrally managing client definitions.

3. Apply the function for centrally managing client definitions.

   After creating the client definition file used by the function for centrally managing client definitions and the client management definition file, apply this function.

   The way of applying the function for centrally managing client definitions differs, depending on whether the HADB server is running.

   - **When the HADB server is stopped**

     Execute the `adbstart` command to start the HADB server. After the HADB server starts, according to the client definition file and client management definition file that you created, the function for centrally managing client definitions is applied.

     For details about the `adbstart` command, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

   - **When the HADB server is running**

     Execute the `adbclientdefmang` command with the `--update` option specified. After executing the `adbclientdefmang` command, according to the client definition file and client management definition file that you created, the function for centrally managing client definitions is applied.

     For details about the `adbclientdefmang` command, see *adbclientdefmang (Centralized Management of Client Definitions)* in the manual *HADB Command Reference*.

     > **Note**
     >
     > When the function for centrally managing client definitions is applied, this function is not applied to the HADB clients that are connected to the HADB server by using the authorization identifier of the HADB user specified for the `-i` option of the `adbclientmang` operand. In that case, disconnect the HADB clients from the HADB server. Then, when the HADB clients are connected to the HADB server next time, the function for centrally managing client definitions is applied.

4. Check the status of how the function for centrally managing client definitions has been applied.

   Execute the `adbclientdefmang` command to check the status of how the function for centrally managing client definitions has been applied.

   From the execution result of the `adbclientdefmang` command, you can check whether the definitions in the client management definition file you created in step 2 have been applied.

   For details about the `adbclientdefmang` command, see *adbclientdefmang (Centralized Management of Client Definitions)* in the manual *HADB Command Reference*.

11.14.4 Changing the definitions of how the function for centrally managing client definitions is applied

The following describes the procedure for changing the definitions of how the function for centrally managing client definitions is applied.

For an explanation of the flow of changing how the function for centrally managing client definitions is applied, see (2) Flow of changing the definitions of how the function for centrally managing client definitions is applied in 11.14.1 Flow of using the function for centrally managing client definitions.

Procedure:

1. Check the status of how the function for centrally managing client definitions has been applied.

   Execute the `adbclientdefmang` command to check the status of how the function for centrally managing client definitions has been applied. From the execution result of the `adbclientdefmang` command, you can check the definitions in the client management definition file that is currently applied.

   For details about the `adbclientdefmang` command, see *adbclientdefmang (Centralized Management of Client Definitions)* in the manual *HADB Command Reference*.

   If you modify the contents of the client definition file used by the function for centrally managing client definitions, go to step 2.

   If you do not modify the contents of the client definition file used by the function for centrally managing client definitions, go to step 3.

2. Edit the client definition file used by the function for centrally managing client definitions.

   Edit the client definition file that is stored in the server directory `$ADBDIR/conf` and used by the function for centrally managing client definitions. For details, see (1) Client definition file used by the function for centrally managing client definitions in 11.14.2 Files required for the function for centrally managing client definitions.

   If you change the association between the client definition file used by the function for centrally managing client definitions and the authorization identifier of the HADB user, go to step 3.

   If you do not change the association between the client definition file used by the function for centrally managing client definitions and the authorization identifier of the HADB user, go to step 4.

3. Edit the client management definition file.

   Edit the client management definition file (`adbclientdefmang.def`) stored in the server directory `$ADBDIR/conf`. For details, see (2) Client management definition file in 11.14.2 Files required for the function for centrally managing client definitions.

4. Change the definitions of how the function for centrally managing client definitions is applied.

   Change the definitions of how the function for centrally managing client definitions is applied.

   The way of changing the definitions of how the function for centrally managing client definitions is applied differs, depending on whether the HADB server is running.

   - **When the HADB server is stopped**

     Execute the `adbstart` command to start the HADB server. After the HADB server starts, according to the client definition file and client management definition file that you edited, the definitions of how the function for centrally managing client definitions is applied are changed.

     For details about the `adbstart` command, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

   - **When the HADB server is running**

     Execute the `adbclientdefmang` command with the `--update` option specified. After executing the `adbclientdefmang` command, according to the client definition file and client management definition file

that you edited, the definitions of how the function for centrally managing client definitions is applied are changed.

For details about the `adbclientdefmang` command, see *adbclientdefmang (Centralized Management of Client Definitions)* in the manual *HADB Command Reference*.

> **Note**
>
> When the function for centrally managing client definitions is applied, this function is not applied to certain HADB clients. Such HADB clients are connected to the HADB server by using the authorization identifier of the HADB user specified for the `-i` option of the `adbclientmang` operand. For such an HADB client, the function for centrally managing client definitions is applied (after the HADB client is disconnected) when the HADB client is re-connected.

5. Check the status of how the function for centrally managing client definitions has been applied.

   Execute the `adbclientdefmang` command to check the status of how the function for centrally managing client definitions has been applied.

   From the execution result of the `adbclientdefmang` command, you can check whether the definitions in the client management definition file you edited in step 3 have been applied.

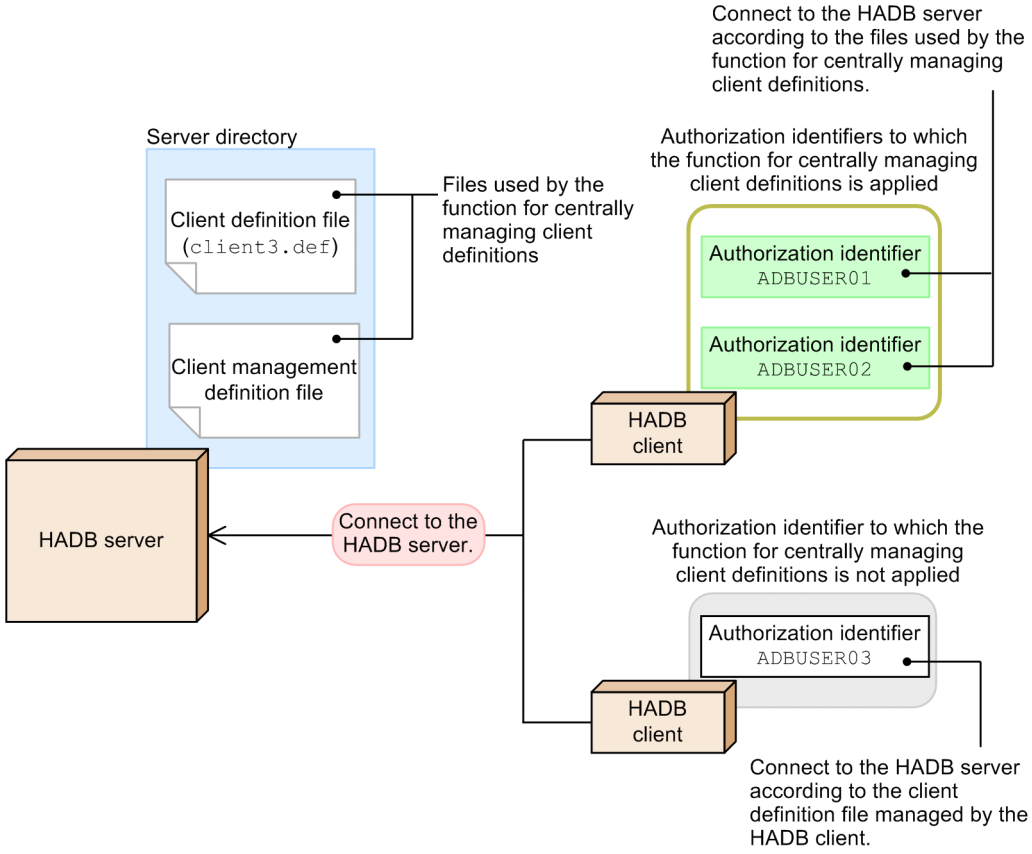
11.14.5 Example of using the function for centrally managing client definitions

This subsection provides an example of using the function for centrally managing client definitions, according to the procedure in 11.14.3 Newly using the function for centrally managing client definitions.

▪ Conditions

- The HADB server is stopped.
- The function for centrally managing client definitions manages the following values specified in the client definition:
  - `adb_sql_exe_max_rthd_num` operand
  - `adb_sql_exe_hashgrp_area_size` operand
  - `adb_sql_exe_hashtbl_area_size` operand
  - `adb_sql_exe_hashflt_area_size` operand
- The following authorization identifiers are applied to the function for centrally managing client definitions:
  - `ADBUSER01`
  - `ADBUSER02`
- The following authorization identifier is not applied to the function for centrally managing client definitions:
  - `ADBUSER03`

Figure 11-45: Example of using the function for centrally managing client definitions



(1) Creating the client definition file to be used by the function for centrally managing client definitions

Create the client definition file to be used by the function for centrally managing client definitions, and in the file, specify individual client definitions.

In this example, create the client definition file (`client3.def`) in the server directory `$ADBDIR/conf`.

Specification example of the client definition file to be used by the function for centrally managing client definitions

```
set adb_sql_exe_max_rthd_num = 12
set adb_sql_exe_hashgrp_area_size = 2400
set adb_sql_exe_hashtbl_area_size = 1000
set adb_sql_exe_hashflt_area_size = 100
```

(2) Creating the client management definition file

Create the client management definition file (`adbclientdefmang.def`) in the server directory `$ADBDIR/conf`, and in the file, specify the `adbclientmang` operand in the client-managing definition.

Specification example of the client management definition file

```
adbclientmang -f client3.def                ...1
              -i ADBUSER01,ADBUSER02        ...2
```

Explanation:

1. Specify the file name (`client3.def`) of the client definition file created in (1) Creating the client definition file to be used by the function for centrally managing client definitions.
2. Specify the authorization identifiers (`ADBUSER01` and `ADBUSER02`) to which the function for centrally managing client definitions is to be applied.

(3) Executing the adbstart command

Execute the `adbstart` command to start the HADB server. After the HADB server starts, according to the contents of the following two files, the function for centrally managing client definitions is applied:

- Client definition file used by the function for centrally managing client definitions
- Client management definition file

> **Note**
>
> If the HADB server is running, execute the `adbclientdefmang` command with the `--update` option specified. The contents in the client management definition file are applied.

(4) Executing the adbclientdefmang command

Execute the `adbclientdefmang` command to check the status of how the function for centrally managing client definitions has been applied.

Execution example of the `adbclientdefmang` command

```
adbclientdefmang -u ADBUSER01         ...1
                 -p '#HelloHADB_01'   ...2
```

Explanation:

1. Specify the authorization identifier of the HADB user who executes the `adbclientdefmang` command.
2. Specify the password for the authorization identifier specified for the `-u` option.

From the execution result of the `adbclientdefmang` command, you can check whether the definitions in the client management definition file you created in (2) Creating the client management definition file have been applied.

Execution result example of the `adbclientdefmang` command

```
authorization-identifier client-definition-file
ADBUSER01                client3.def
ADBUSER02                client3.def
```

Explanation:

You can confirm that the client definition file (`client3.def`) used by the function for centrally managing client definitions has been applied to the authorization identifiers `ADBUSER01` and `ADBUSER02`.

11.14.6 Stopping the use of the function for centrally managing client definitions

The following describes the procedure for stopping the use of the function for centrally managing client definitions.

Procedure:

1. Invalidate the client management definition file.

   Perform one of the following operations on the client management definition file (`adbclientdefmang.def`) stored in the server directory `$ADBDIR/conf`:

   - Deleting the client management definition file
   - Emptying the client management definition file (so that nothing is specified in the file)
   - Commenting out all definitions in the client management definition file

2. Stop the use of the function for centrally managing client definitions.

   Stop the use of the function for centrally managing client definitions based on the invalidated client management definition file.

   The way of stopping the use of the function for centrally managing client definitions differs, depending on whether the HADB server is running.

   - **When the HADB server is stopped**

     Execute the `adbstart` command to start the HADB server. After the HADB server starts, according to the invalidated client management definition file, the use of the function for centrally managing client definitions is stopped.

     For details about the `adbstart` command, see *adbstart (Start the HADB Server)* in the manual *HADB Command Reference*.

   - **When the HADB server is running**

     Execute the `adbclientdefmang` command with the `--update` option specified. After the `adbclientdefmang` command is executed, according to the invalidated client management definition file, the use of the function for centrally managing client definitions is stopped.

     For details about the `adbclientdefmang` command, see *adbclientdefmang (Centralized Management of Client Definitions)* in the manual *HADB Command Reference*.

     > **Note**
     >
     > The operation to stop the use of the function for centrally managing client definitions does not take effect in the following case: When the HADB client is connected to the HADB server with the authorization identifier of the HADB user specified for the `-i` option of the `adbclientmang` operand. In such a case, the use of the function for centrally managing client definitions can be stopped after the HADB client is disconnected from the HADB server, and then re-connected to the HADB server.

11.15 Handling of data retrieval from CSV files

This section explains how to retrieve data from CSV files.

11.15.1 Preparatory tasks

Complete the following preparatory tasks.

(1) Preparing the CSV files

Perform the following:

- The CSV files need to satisfy any of the following conditions. Confirm that the CSV files satisfy the condition.
  - Uncompressed CSV files
  - CSV files compressed in GZIP format by using the OS's `gzip` command
  - CSV files that were exported by the `adbexport` command and compressed in GZIP format
- Confirm that the data format of the CSV files observes the rules. For details about the rules for the data format of CSV files, see *Rules for CSV files* under *Rules* under *ADB_CSVREAD function* in *System-defined functions* in *Constituent Elements* in the manual *HADB SQL Reference*. If the data format of a CSV file is invalid, an error will occur during data retrieval processing.
- Store the CSV files in the file system of the server machine on which the HADB server is running.
- Make sure that the absolute path name of a CSV file consists of no more than 510 bytes.

(2) Granting the read privilege for the CSV files and their directory

Grant the following privileges to the HADB administrator so that the HADB server can open the CSV files:

- Read privilege for the CSV files
- Read and execute privileges for the directory storing the CSV files

11.15.2 Data retrieval method

Execute the `SELECT` statement with the `ADB_CSVREAD` function specified to retrieve data in the CSV files.

Example:

This example retrieves data in the following CSV files that have been compressed into GZIP format:

- `/dir/file1.csv.gz`
- `/dir/file2.csv.gz`
- `/dir/file3.csv.gz`

```
SELECT * FROM TABLE(ADB_CSVREAD(
               MULTISET['/dir/file1.csv.gz','/dir/file2.csv.gz','/dir/file3.csv.gz'
],
              'COMPRESSION_FORMAT=GZIP;
               FIELD_NUM=1,2;
               ENCLOSING_CHAR=";
```

```
            DELIMITER_CHAR=,;'))
        AS "T1" ("C1" INTEGER, "C2" VARCHAR(32))
```

Explanation

This example specifies the absolute path names of the CSV files in the `MULTISET` argument (underlined portion) of the `ADB_CSVREAD` function. For details about other arguments and how to specify the `ADB_CSVREAD` function, see *ADB_CSVREAD function* in *System-defined functions* in *Constituent Elements* in the manual *HADB SQL Reference*.

Important

When an SQL statement that specifies the `ADB_CSVREAD` function is executed, the HADB server opens the specified CSV files and reads the data in the files. You must not edit the CSV files while the SQL statement is executing.

Note

- When data in CSV files is read, one processing real thread is allocated per CSV file.
- You can also use a table subquery to specify the CSV files that are specified in the `ADB_CSVREAD` function. For details, see *ADB_CSVREAD function* in *System-defined functions* in *Constituent Elements* in the manual *HADB SQL Reference*.

11.16 Performing synonym search operations

This section describes preparatory tasks for synonym search operations, how to perform synonym search operations, and how to manage synonym dictionaries.

For an overview of synonym search, see 2.17.3 Synonym search.

If you perform synonym search operations when the multi-node function is being used, also read the explanation in 16.24 Performing synonym search operations (when using the multi-node function).

11.16.1 Preparing for synonym search operations

Before performing synonym search operations, perform the following preparatory tasks.

(1) Considering defining text indexes

When you perform synonym search operations, performance improvement is expected if you use text indexes. Therefore, we recommend that you define text indexes for the columns in which text data (document data) is stored.

▪ Cases in which no effect can be expected even if text indexes are defined

If there are only a few types of characters in the text data, performance improvement cannot be expected even if text indexes are defined. For example, this case includes when the text data consists only of numbers.

Also, when you perform a synonym search operation, if you specify as the search-target character string a simple and short character string (such as `a` or `0`), performance improvement cannot be expected even if text indexes are defined. In some cases, text indexes might not be used.

For details about the points to consider when defining text indexes, see 5.4.1 Points to consider in determining the columns to be defined for a text index.

▪ Relationship with correction search

Synonym search operations can be performed together with correction search operations. To perform synonym search and correction search operations at the same time, when you define a text index in the `CREATE INDEX` statement, specify `CORRECTIONRULE` (notation-correction-search text-index specification).

For details about correction search, see 2.17.1 Correction search.

You can select whether to apply correction search in the specification of the scalar function `CONTAINS` when you perform a synonym search operation. Therefore, if whether to apply correction search is uncertain in the preparation step, we recommend that you define text indexes that support correction search.

> **Important**
>
> If the character encoding used on the HADB server is `Shift-JIS` (if the value specified for the environment variable `ADBLANG` is `SJIS`), correction search operations cannot be performed.

Even if you do not define text indexes, you can perform synonym search and correction search operations at the same time. However, from the viewpoint of performance, we recommend that you define text indexes.

■ Relationship with word-context search

Synonym search operations can be performed together with word-context search operations. To perform synonym search and word-context search operations at the same time, when you define a text index in the `CREATE INDEX` statement, specify `TEXT WORDCONTEXT` for `INDEXTYPE` (define a text index for a word-context search).

For details about word-context search, see 2.17.4 Word-context search.

You can select whether to apply word-context search in the specification of the scalar function `CONTAINS` when you perform a synonym search operation. Therefore, if whether to apply word-context search is uncertain in the preparation step, we recommend that you define text indexes for word-context search.

Even if you do not define text indexes, you can perform synonym search and word-context search operations at the same time. However, from the viewpoint of performance, we recommend that you define text indexes.

(2) Defining text indexes

To define a text index, perform the following procedure.

Procedure:

1. Execute the `CREATE INDEX` statement to define a text index.
2. Execute the `adbidxrebuild` command to create index data for the text index.

(3) Estimating the size required for the directory for storing synonym dictionary files

If you execute the `adbsyndict` command (which registers synonym dictionaries), synonym dictionary files that store information about individual synonym dictionaries are created. Estimate the size required for the directory for storing these synonym dictionary files. For the approximate size required for the directory for storing synonym dictionary files, see 11.16.12 Checking the free space required for the directory for storing synonym dictionary files.

To estimate in detail the size required for the directory for storing synonym dictionary files, see the following parts of this manual:

- 6.14 Estimating the size of the synonym dictionary file directory
- 6.21.6 Estimating the size of the temporary work file for executing the adbsyndict command

The total size determined from the preceding estimation is the size required for the directory for storing the synonym dictionary files.

Important

If free space for the directory for storing synonym dictionary files is insufficient, you need to change the directory for storing synonym directory files. If you change the directory for storing synonym dictionary files, you need to reregister all synonym dictionaries. Therefore, estimate the size of the directory so that the disk (on which the directory is created) will have enough free space.

(4) Creating the directory for storing synonym dictionary files

Create the directory for storing synonym dictionary files. From the performance point of view, we recommend that you prepare a file system in which only the directory for storing synonym dictionary files is created.

When you create the directory for storing synonym dictionary files, note the following.

- The following directories cannot be the directory for storing synonym dictionary files:
  - Server directory
  - Subordinate directories of the server directory
  - Directories that have the server directory as a subordinate directory

- DB directory
- Subordinate directories of the DB directory
- Directories that have the DB directory as a subordinate directory
- Root directory

The following shows example directories that can be and that cannot be the directory for storing synonym dictionary files when the DB directory is `/HADB/db`.

Example of the directory for storing synonym dictionary files		Reason
Example of the directory that can be the directory for storing synonym dictionary files	`/HADB/syndict`	None.
Example of the directories that cannot be the directory for storing synonym dictionary files	`/HADB/db`	The directory in the left column cannot be the directory for storing synonym dictionary files because it is the DB directory.
	`/HADB/db/synonym`	The directory in the left column cannot be the directory for storing synonym dictionary files because it is a subordinate directory of the DB directory.
	`/HADB`	The directory in the left column cannot be the directory for storing synonym dictionary files because it has the DB directory as a subordinate directory.

- Do not use, as the directory for storing synonym dictionary files, the directory that was used for storing installation data when the HADB server was installed.

Note

When you back up a database, the directory for storing synonym dictionary files is also backed up. For details about the backup acquisition method, see 10.3 Backing up a database.

(5) Assigning permissions to the directory for storing synonym dictionary files

Use the following procedure to assign permissions to the directory for storing synonym dictionary files.

Procedure:

1. To the directory for storing synonym dictionary files, assign the permissions that allow reading, writing, and execution by the HADB administrator.
2. To all directories included in the path of the directory for storing synonym dictionary files, assign permissions that allow execution by the HADB administrator.

Example:

If the directory for storing synonym dictionary files is `/HADB/syndict`:

- Read, write, and execution permissions must be assigned to `/HADB/syndict`.
- Execution permission must be assigned to `/` and `/HADB`.

(6) Modifying the server definition

For the `adb_syndict_storage_path` operand in the server definition, specify the name of the directory for storing synonym dictionary files. For details about the `adb_syndict_storage_path` operand, see adb_syndict_storage_path in 7.2.8 Operands related to synonym search (set format).

If you specify a symbolic link to the storage directory for synonym dictionary files in the `adb_syndict_storage_path` operand, a check is performed to determine whether the absolute path name generated after the symbolic link is resolved conforms to the rules explained in (4) Creating the directory for storing synonym dictionary files.

> **Note**
>
> For details about the procedure for modifying the server definition, see 8.5.2 Modifying the server definition.

(7) Creating a synonym list definition file

Create a file in which a list of synonyms to be registered in a synonym dictionary is written. This file is called a *synonym list definition file*. When you perform a synonym search operation, the synonyms written in the synonym list definition file are retrieved in a batch.

▪ Specification example of a synonym list definition file

```
database,data bank,DB↓
application server,AP server↓
```

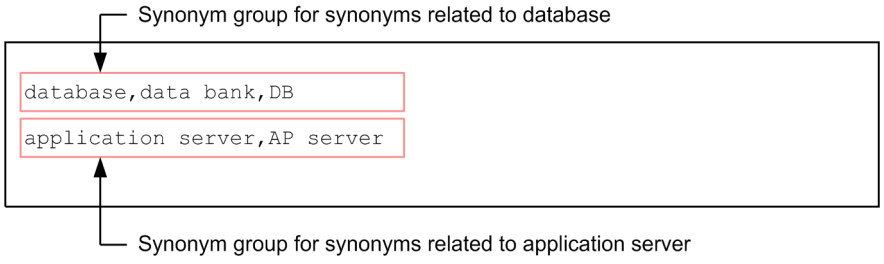
Legend: ↓: Line feed

When you create a synonym list definition file as in the preceding example, if you specify `database`, `data bank`, or `DB` as the search-target character string, all of `database`, `data bank`, and `DB` become the search-target character string.

If you specify `application server` or `AP server` as the search-target character string, both of `application server` and `AP server` become the search-target character string.

▪ Specification rules for a synonym list definition file

- Individual synonyms are delimited by `,`.
- The synonyms subject to batch search are written on one line. A group of synonyms written on one line is called a *synonym group*. In the following example, two synonym groups are specified.



> **Note**
>
> Only the basic specification rules for a synonym list definition file are explained here. For details about the specification rules for a synonym list definition file, see 11.16.15 Specification rules for a synonym list definition file.

This example assumes that the synonym list definition file is saved with the following file name:

- `/home/adbmanager/dictionary1_synonym.txt`

> **Important**
>
> Be sure to save the synonym list definition file that you created here because it is used, for example, when a synonym is added to a synonym dictionary.

(8) Creating a dictionary creation file

Create a file in which information about synonym dictionaries is specified. This file is called a *dictionary creation file*. Specify the dictionary creation file that you create here, as the argument of an option of the `adbsyndict` command that is executed when synonym dictionaries are registered.

▪ Specification example of a dictionary creation file

```
Dictionary1,/home/adbmanager/dictionary1_synonym.txt,CORRECTIONRULE,terms related
to DB and AP server↓
```

Legend: ↓: Line feed

Explanation:

The following four items are specified in a dictionary creation file. Specify individual items by delimiting with a comma (`,`).

- First line:
  Specify a synonym dictionary name.
- Second line:
  Specify the name of the synonym list definition file you created in (7) Creating a synonym list definition file.
- Third line:
  Specify the correction search option. For the correction search option, specify whether to create a synonym dictionary that supports correction search. Specify either of the following:
  - `CORRECTIONRULE`: Specify this when you create a synonym dictionary that supports correction search.
  - `CASESENSITIVE`: Specify this when you create a synonym dictionary that does not support correction search.
- Fourth line:
  Specify a comment for the synonym dictionary.

Note

Only the basic specification rules for a dictionary creation file are explained here. For details about the specification rules for a dictionary creation file, see 11.16.16 Specification rules for a dictionary creation file.

This example assumes that the dictionary creation file is saved with the following file name:

- `/home/adbmanager/dictionary1_information.txt`

Important

Be sure to save the dictionary creation file that is created here because it is used, for example, when a synonym is added to a synonym dictionary.

(9) Registering a synonym dictionary

Execute the `adbsyndict` command to register a synonym dictionary.

Command execution example

```
adbsyndict -m /home/adbmanager/dictionary1_information.txt
```

Explanation:

In the `-m` option, specify the absolute path name of the dictionary creation file you created in (8) Creating a dictionary creation file.

Note

- When registration of synonym dictionaries is completed, the `KFAA51509-I` message is output. (As many messages as the synonym dictionaries for which registration is completed are output.) Identify the synonym dictionary names in the message, and confirm that registration of the synonym dictionaries is completed.
- If you execute the `adbsyndict` command, one or multiple synonym dictionary files are created. If a synonym dictionary supports correction search, three synonym dictionary files are created. If a synonym dictionary does not support correction search, only one synonym dictionary file is created.
- Do not delete or update the synonym dictionary files created in the directory for storing synonym dictionary files by using a method other than the `adbsyndict` command.

(10) Making a backup of the directory for storing synonym dictionary files

In case a failure occurs in a synonym dictionary file, make a backup of the directory for storing synonym dictionary files. If a failure occurs in a synonym dictionary file when you have not made a backup, you need to re-create the synonym dictionary file by using the `adbsyndict` command.

The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

11.16.2 Performing synonym search operations

Perform synonym search operations by using the scalar function CONTAINS. For details about the scalar function CONTAINS, see *CONTAINS* in the manual *HADB SQL Reference*.

(1) Basic search example

In this example, a synonym search operation is performed by using synonym dictionary Dictionary1. The following terms are registered as synonyms in synonym dictionary Dictionary1.

- database, data bank, DB

Example:

In the DOCUMENTS column, document data is stored. In this example, the names (in the TITLE column) of the document data that contains any of database, data bank, or DB are retrieved.

SQL statement to be executed

```
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1","data bank")')>0
```

Explanation:

In scalar function CONTAINS, the synonym dictionary name (Dictionary1) and one of the synonyms registered in Dictionary1 (data bank) are specified. If you execute the preceding SQL statement, the names (in the TITLE column) of certain document datasets are returned as the search result. Such document datasets contain, in their sentence information, any of the synonyms (database, data bank, and DB) registered in the same synonym group in the synonym dictionary.

> **Note**
>
> For details about how to check the synonyms registered in the synonym dictionary, see 11.16.3 Checking the synonyms registered in a synonym dictionary.

(2) Search example in which correction search is applied

In this example, a synonym search operation is performed by using synonym dictionary Dictionary1, which supports correction search. The following terms are registered as synonyms in synonym dictionary Dictionary1.

- database, data bank, DB

Example:

In the DOCUMENTS column, document data is stored. In this example, the names (in the TITLE column) of the document datasets that contain any of database, data bank, DB, DATABASE, Data bank, or db are retrieved.

SQL statement to be executed

```
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1",SORTCODE("data bank"))')>0
```

Explanation:

In the scalar function CONTAINS, the synonym dictionary name (Dictionary1) and one of the synonyms registered in Dictionary1 (data bank) are specified. If you execute the preceding SQL statement, the names (in the TITLE column) of certain document datasets are returned as the search result. Such document

datasets contain, in their sentence information, any of the synonyms (database, data bank, and DB) registered in the same synonym group in the synonym dictionary.
Also, the names (in the TITLE column) of the document datasets that contain the term DATABASE, Data bank, or db are also returned as the search result. This is because the function of correction search is working.

(3) Search example in which two synonym dictionaries are used

The following provides a search example in which two synonym dictionaries are used.

- Synonym dictionary name: Dictionary2
  The following terms are registered as synonyms in synonym dictionary Dictionary2:
  - relational database, RDB, NoSQL database
- Synonym dictionary name: Dictionary3
  The following terms are registered as synonyms in synonym dictionary Dictionary3:
  - relational database, RDB, NoSQL database, No-SQL database, cloud database

Example:

In this example, the names (in the TITLE column) of the document datasets that satisfy all the following conditions are retrieved:

- Document datasets stored in the DOCUMENTS column contain any of the terms relational database, RDB, and NoSQL database.
- Document dataset names stored in the TITLE column contain any of the terms relational database, RDB, NoSQL database, No-SQL database, and cloud database.

```
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary2","RDB")')>0
      AND CONTAINS("TITLE",'SYNONYM("Dictionary3","RDB")')>0
```

(4) Notes

- If the search-target character string specified for scalar function CONTAINS has not been registered in the synonym dictionary, synonym search operations are not performed.
  Example:
  When the synonyms registered in the synonym dictionary are database, data bank, and DB, if the following SELECT statement is executed, a synonym search operation is not performed. In this case, only RDB is the search-target character string.

```
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1",SORTCODE("RDB"))')>0
```

- While a synonym dictionary is being updated (while the adbsyndict command is being executed), you can perform synonym search operations by using the synonym dictionary being updated. Note, however, that whether the search operation uses the synonym dictionary in the state before update or after update differs depending on the timing.

11.16.3 Checking the synonyms registered in a synonym dictionary

By executing the adbsyndict command, you can output a list of synonyms registered in a synonym dictionary.

Example:

In this example, a list of synonyms registered in synonym dictionary `Dictionary1` is output.

Procedure:

1. Execute the `adbsyndict` command to output a list of synonyms registered in synonym dictionary `Dictionary1`.

```
adbsyndict -n Dictionary1 -o /home/adbmanager/dictionary1_output.txt
```

For the `-n` option, specify the synonym dictionary name.

For the `-o` option, specify the file to which a list of synonyms registered in synonym dictionary `Dictionary1` is output.

2. Open the file that was output, and check the list of synonyms.

Output example of a list of synonyms:

```
database,data bank,DB↓
application server,AP server↓
```

Legend: ↓: Line feed

A list of synonyms is output in the same format as a synonym list definition file. As shown in the preceding example, one line indicates the synonyms in one synonym group.

> **Note**
>
> When output of the list of synonyms is completed in step 1, the `KFAA51529-I` message is output. Check the synonym dictionary name in the message to confirm that output of the list of synonyms has been completed.

11.16.4 Checking the information about synonym dictionaries

When you want to check a synonym dictionary name, or when you want to check whether a synonym dictionary supports correction search, search the system table.

(1) Checking the information about a specific synonym dictionary

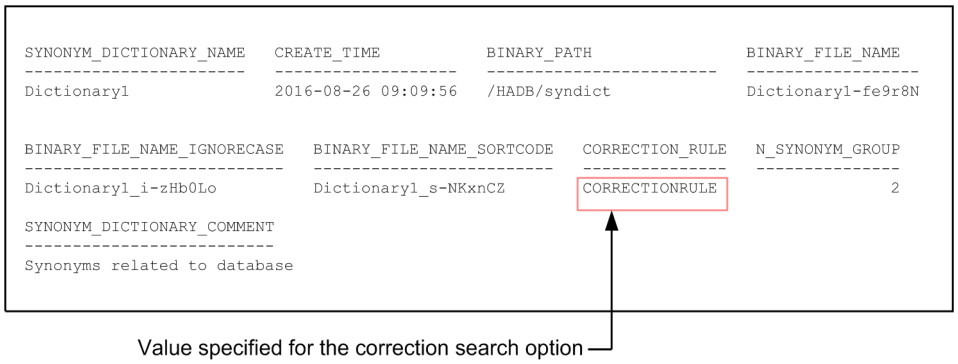
Example:

In this example, whether the synonym dictionary `Dictionary1` supports correction search is checked.

SQL statement to be executed

```
SELECT * FROM "MASTER"."STATUS_SYNONYM_DICTIONARIES"
    WHERE "SYNONYM_DICTIONARY_NAME"='Dictionary1'
```

Execution result example



Explanation:

Because `CORRECTIONRULE` is displayed in the `CORRECTION_RULE` column, synonym dictionary `Dictionary1` supports correction search.

If `CASESENSITIVE` is displayed in the `CORRECTION_RULE` column, the synonym dictionary does not support correction search.

For output items that do not have an explanation, see the explanation of the column information in `STATUS_SYNONYM_DICTIONARIES` in C.6 Content of STATUS_SYNONYM_DICTIONARIES.

The display format of the preceding search result is partially different from the display format of the actual output.

(2) Checking the information about all synonym dictionaries

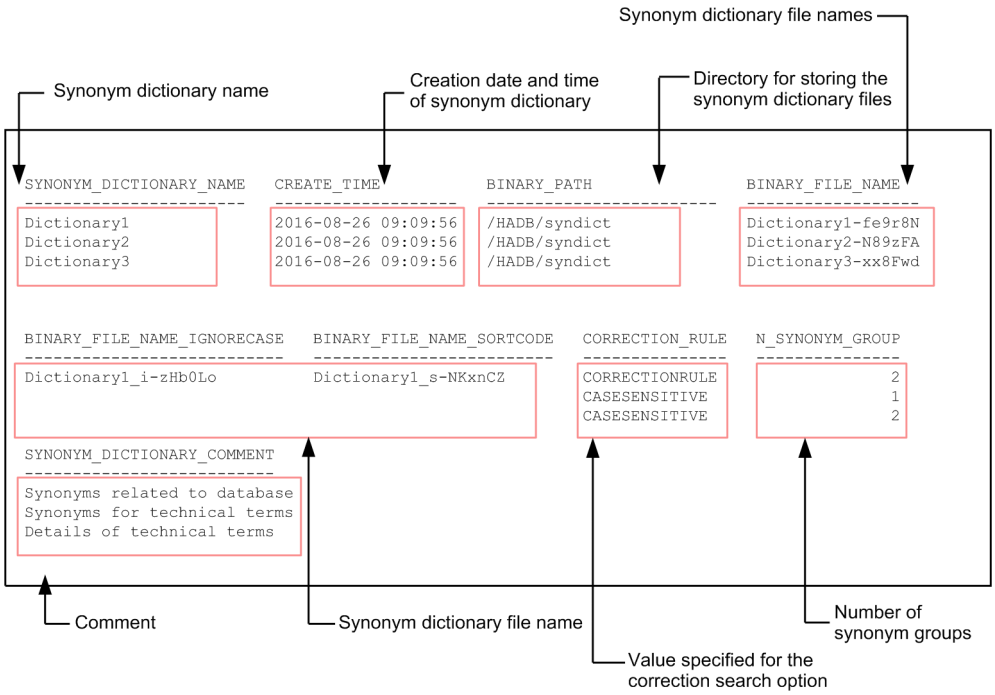
Example:

In this example, the list of synonym dictionaries and information regarding individual synonym dictionaries are checked.

SQL statement to be executed

```
SELECT * FROM "MASTER"."STATUS_SYNONYM_DICTIONARIES"
```

Execution result example



For example, when you display the list of synonym dictionaries and check the synonym dictionary name you specify in the SQL statement, guess the applicable synonym dictionary from the comment.

The display format of the preceding search result is partially different from the display format of the actual output.

11.16.5 Adding synonyms

This subsection describes how to add synonyms to a synonym dictionary.

Example:

Synonyms are added to a synonym group registered in synonym dictionary `Dictionary1`. In this example, `relational database` and `RDB` are added.

Procedure:

1. Modify the contents specified in the saved synonym list definition file.

   <Contents specified in the synonym list definition file>

   ```
   database,data bank,DB,relational database,RDB↓
   application server,AP server↓
   ```

   Legend: ↓: Line feed

   Modify the saved synonym list definition file for `Dictionary1`. In this example, the underlined synonyms are added.

   > **Important**
   >
   > Keep specifying the synonym groups that have no change as they were. In the preceding example, keep specifying the second line (the line that contains "application server") as it was. If you do not specify the second line in the preceding example, the synonym group on the line that contains "application server" will be deleted.

   This example assumes that the synonym list definition file for `Dictionary1` is saved with the following file name:

   - `/home/adbmanager/dictionary1_synonym.txt`

2. Check the contents specified in the saved dictionary creation file.

   <Contents specified in the dictionary creation file>

   ```
   Dictionary1,/home/adbmanager/dictionary1_synonym.txt,CORRECTIONRULE,terms related
   to DB and AP server↓
   ```

   Legend: ↓: Line feed

   Check the contents specified in the saved dictionary creation file for `Dictionary1`. If you need to modify the comment due to addition of synonyms, modify the comment.

   This example assumes that the dictionary creation file is saved with the following file name:

   - `/home/adbmanager/dictionary1_information.txt`

3. Check the free space of the directory for storing the synonym dictionary file.

   For details about how to check the free space, see 11.16.12 Checking the free space required for the directory for storing synonym dictionary files.

4. Execute the `adbsyndict` command to update the synonym dictionary.

```
adbsyndict -m /home/adbmanager/dictionary1_information.txt
```

For the -m option, specify the name of the absolute path to the dictionary creation file that you checked in step 2.

5. Make a backup of the directory for storing synonym dictionary files.

   If you update a synonym dictionary, the relevant synonym dictionary files are updated. In case a failure occurs in a synonym dictionary file, make a backup of the directory for storing synonym dictionary files. If a failure occurs in a synonym dictionary file when you have not made a backup, you need to re-create the synonym dictionary file by using the adbsyndict command.

   The directory for storing synonym dictionary files is the directory specified for the adb_syndict_storage_path operand in the server definition.

Note

- For details about the specification rules for a synonym list definition file, see 11.16.15 Specification rules for a synonym list definition file.
- For details about the specification rules for a dictionary creation file, see 11.16.16 Specification rules for a dictionary creation file.
- If you have lost a synonym list definition file, rebuild the synonym list definition file based on the explanation in 11.16.14 Rebuilding a synonym list definition file (when a synonym list definition file is lost).
- When update of a synonym dictionary is completed in step 4, the KFAA51509-I message is output. (As many messages as the synonym dictionaries for which registration is completed are output.) Identify the synonym dictionary name in the message, and confirm that update of the synonym dictionary is completed.

11.16.6 Deleting synonyms

This subsection describes how to delete synonyms from a synonym dictionary.

Example:

A synonym registered in synonym dictionary Dictionary1 is deleted. In this example, synonym data bank is deleted.

Procedure:

1. Modify the contents specified in the saved synonym list definition file.

   <Contents specified in the synonym list definition file before change>

```
database,data bank,DB↓
application server,AP server↓
```

   <Contents specified in the synonym list definition file after change>

```
database,DB↓
application server,AP server↓
```

   Legend: ↓: Line feed

   Modify the saved synonym list definition file for Dictionary1. In this example, synonym data bank is deleted.

Important

Keep specifying unchanged synonym groups as they were. In the preceding example, keep specifying the second line (the line that contains "application server") as it was. If you do not specify the second line in the preceding example, the synonym group on the line that contains "application server" will be deleted.

This example assumes that the synonym list definition file for `Dictionary1` is saved with the following file name:

- `/home/adbmanager/dictionary1_synonym.txt`

2. Check the contents specified in the saved dictionary creation file.

<Contents specified in the dictionary creation file>

```
Dictionary1,/home/adbmanager/dictionary1_synonym.txt,CORRECTIONRULE,terms related
to DB and AP server↓
```

Legend: ↓: Line feed

Check the contents specified in the saved dictionary creation file. If you need to modify the comment due to deletion of synonyms, modify the comment.

This example assumes that the dictionary creation file is saved with the following file name:

- `/home/adbmanager/dictionary1_information.txt`

3. Check the free space of the directory for storing the synonym dictionary file.

For details about how to check the free space, see 11.16.12 Checking the free space required for the directory for storing synonym dictionary files.

4. Execute the `adbsyndict` command to update the synonym dictionary.

```
adbsyndict -m /home/adbmanager/dictionary1_information.txt
```

For the `-m` option, specify the name of the absolute path to the dictionary creation file that you checked in step 2.

5. Make a backup of the directory for storing synonym dictionary files.

If you update the synonym dictionary, the relevant synonym dictionary files are updated. In case a failure occurs in a synonym dictionary file, make a backup of the directory for storing synonym dictionary files. If a failure occurs in a synonym dictionary file when you have not made a backup, you need to re-create the synonym dictionary file by using the `adbsyndict` command.

The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

Note

- For details about the specification rules for a synonym list definition file, see 11.16.15 Specification rules for a synonym list definition file.
- For details about the specification rules for a dictionary creation file, see 11.16.16 Specification rules for a dictionary creation file.
- If you have lost a synonym list definition file, rebuild the synonym list definition file based on the explanation in 11.16.14 Rebuilding a synonym list definition file (when a synonym list definition file is lost).
- When update of a synonym dictionary is completed in step 4, the `KFAA51509-I` message is output. (As many messages as the synonym dictionaries for which update is completed are output.) Identify the

synonym dictionary name in the message, and confirm that update of the synonym dictionary is completed.

11.16.7 Adding synonym groups

This subsection describes how to add synonym groups to a synonym dictionary.

Example:

In this example, a synonym group is added to synonym dictionary `Dictionary1`. The following synonyms are registered in the synonym group to be added:

- `WWW server,Web server`

Procedure:

1. Modify the contents specified in the saved synonym list definition file.

   <Contents specified in the synonym list definition file>

   ```
   database,data bank,DB↓
   application server,AP server↓
   WWW server,Web server↓
   ```

   Legend: ↓: Line feed

   Modify the saved synonym list definition file for `Dictionary1`. In this example, synonym groups are added.

   > **Important**
   >
   > Keep specifying the unchanged synonym groups as they were. In the preceding example, keep specifying the first line (the line that contains "database") and the second line (the line that contains "application server") as they were. If you do not specify the first and second lines in the preceding example, the synonym group for these lines will be deleted.

   This example assumes that the synonym list definition file for `Dictionary1` is saved with the following file name:

   - `/home/adbmanager/dictionary1_synonym.txt`

2. Check the contents specified in the saved dictionary creation file.

   <Contents specified in the dictionary creation file>

   ```
   Dictionary1,/home/adbmanager/dictionary1_synonym.txt,CORRECTIONRULE,"terms related
    to DB, AP server, and Web server"↓
   ```

   Legend: ↓: Line feed

   Check the contents specified in the saved dictionary creation file. If you need to modify the comment due to addition of synonym groups, modify the comment. In this example, the comment (the underlined part) is modified.

   > **Note**
   >
   > Because `,` is specified in the comment, the comment is enclosed by `"`.

   This example assumes that the dictionary creation file is saved with the following file name:

   - `/home/adbmanager/dictionary1_information.txt`

3. Check the free space of the directory for storing the synonym dictionary file.

For details about how to check the free space, see 11.16.12 Checking the free space required for the directory for storing synonym dictionary files.

4. Execute the `adbsyndict` command to update the synonym dictionary.

```
adbsyndict -m /home/adbmanager/dictionary1_information.txt
```

For the `-m` option, specify the name of the absolute path to the dictionary creation file that you checked in step 2.

5. Make a backup of the directory for storing synonym dictionary files.

If you update the synonym dictionary, the relevant synonym dictionary files are updated. In case a failure occurs in a synonym dictionary file, make a backup of the directory for storing synonym dictionary files. If a failure occurs in a synonym dictionary file when you have not made a backup, you need to re-create the synonym dictionary file by using the `adbsyndict` command.

The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

Note

- For details about the specification rules for a synonym list definition file, see 11.16.15 Specification rules for a synonym list definition file.
- For details about the specification rules for a dictionary creation file, see 11.16.16 Specification rules for a dictionary creation file.
- If you have lost a synonym list definition file, rebuild the synonym list definition file based on the explanation in 11.16.14 Rebuilding a synonym list definition file (when a synonym list definition file is lost).
- When update of a synonym dictionary is completed in step 4, the `KFAA51509-I` message is output. (As many messages as the synonym dictionaries for which update is completed are output.) Identify the synonym dictionary name in the message, and confirm that update of the synonym dictionary is completed.

▪ **Deleting synonym groups**

The procedure for deleting synonym groups is almost the same as the procedure for adding synonym groups that is described here. In step 1 *Modify the contents specified in the saved synonym list definition file*, delete unnecessary synonym groups.

11.16.8 Registering a synonym dictionary

This subsection describes how to register a synonym dictionary.

Example:

In this example, synonym dictionary `Dictionary2` is registered. In `Dictionary2`, the following synonyms are registered. (Only one synonym group is registered.)

- `HADB`, `Hitachi Advanced Database`, `HADB database`

Procedure:

1. Create a synonym list definition file for `Dictionary2`.

<Contents specified in the synonym list definition file>

```
HADB,Hitachi Advanced Database,HADB database↓
```

Legend: ↓: Line feed

This example assumes that the created synonym list definition file is saved with the following file name:

- /home/adbmanager/dictionary2_synonym.txt

2. Creating a dictionary creation file

<Contents specified in the dictionary creation file>

```
Dictionary2,/home/adbmanager/dictionary2_synonym.txt,CORRECTIONRULE,synonym of HAD
B↓
```

Legend: ↓: Line feed

This example assumes that the created dictionary creation file is saved with the following file name:

- /home/adbmanager/dictionary2_information.txt

3. Check the free space of the directory for storing the synonym dictionary file.

For details about how to check the free space, see 11.16.12 Checking the free space required for the directory for storing synonym dictionary files.

4. Execute the adbsyndict command to register a synonym dictionary.

```
adbsyndict -m /home/adbmanager/dictionary2_information.txt
```

For the -m option, specify the name of the absolute path to the dictionary creation file that you created in step 2.

5. Make a backup of the directory for storing synonym dictionary files.

If you register a synonym dictionary, synonym dictionary files are added. In case a failure occurs in a synonym dictionary file, make a backup of the directory for storing synonym dictionary files. If a failure occurs in a synonym dictionary file when you have not made a backup, you need to re-create the synonym dictionary file by using the adbsyndict command.

The directory for storing synonym dictionary files is the directory specified for the adb_syndict_storage_path operand in the server definition.

Note

- For details about the specification rules for a synonym list definition file, see 11.16.15 Specification rules for a synonym list definition file.
- For details about the specification rules for a dictionary creation file, see 11.16.16 Specification rules for a dictionary creation file.
- When registration of a synonym dictionary is completed in step 4, the KFAA51509-I message is output. (As many messages as the synonym dictionaries for which registration is completed are output.) Identify the synonym dictionary name in the message, and confirm that registration of the synonym dictionary is completed.

11.16.9 Deleting synonym dictionaries

This subsection describes how to delete synonym dictionaries.

Example:

In this example, synonym dictionaries Dictionary1 and Dictionary2 are deleted.

Procedure:

1. Create a dictionary deletion file.
   <Contents specified in the dictionary deletion file>

   ```
   Dictionary1↓
   Dictionary2↓
   ```

   Legend: ↓: Line feed
   Specify the names of the synonym dictionaries to be deleted. Specify a synonym dictionary name per line.
   This example assumes that the created dictionary deletion file is saved with the following file name:
   - `/home/adbmanager/dictionary_delete.txt`
2. Execute the `adbsyndict` command to delete synonym dictionaries.

   ```
   adbsyndict -d /home/adbmanager/dictionary_delete.txt
   ```

   For the `-d` option, specify the name of the absolute path to the dictionary deletion file.

▪ Specification rules for a dictionary deletion file

- Input a line feed at the end of each line. Also, input a line feed at the last line of the dictionary deletion file. Use as the linefeed code one from `X'0A'` (`LF`), `X'0D0A'` (`CRLF`), or `X'00'`.
- Create a dictionary deletion file by using the character encoding specified for environment variable `ADBLANG`.
- If there is a line that contains only a line feed, that line is skipped when processing is executed.

> **Note**
>
> When deletion of synonym dictionaries is completed in step 2, the `KFAA51509-I` message is output. (As many messages as the synonym dictionaries for which deletion is completed are output.) Identify the synonym dictionary names in the message, and confirm that deletion of the synonym dictionaries is completed.

11.16.10 Changing a synonym dictionary to support correction search

This subsection describes how to change a synonym dictionary that does not support correction search to a synonym dictionary that supports correction search.

Example:

In this example, synonym dictionary `Dictionary3` that does not support correction search is changed to a synonym dictionary that supports correction search. The comment is also modified.

Procedure:

1. Modify the contents specified in the saved dictionary creation file.
   <Contents specified in the dictionary creation file before change>

   ```
   Dictionary3,/home/adbmanager/dictionary3_synonym.txt,CASESENSITIVE,detailed term↓
   ```

   <Contents specified in the dictionary creation file after change>

```
Dictionary3,/home/adbmanager/dictionary3_synonym.txt,CORRECTIONRULE,"detailed term
,correction support"↓
```

Legend: ↓: Line feed

Modify the contents specified in the saved dictionary creation file for `Dictionary3`. In this example, the following changes are made.

- Third line:

  Specification for the correction search option is modified from `CASESENSITIVE` to `CORRECTIONRULE`.
- Fourth line:

  The comment is modified. Because `,` is specified in the comment, the comment is enclosed by enclosing character `"`.

This example assumes that the dictionary creation file is saved with the following file name:

- `/home/adbmanager/dictionary3_information.txt`

2. Check the free space of the directory for storing the synonym dictionary file.

   For details about how to check the free space, see 11.16.12 Checking the free space required for the directory for storing synonym dictionary files.

3. Execute the `adbsyndict` command to update the synonym dictionary.

```
adbsyndict -m /home/adbmanager/dictionary3_information.txt
```

   For the `-m` option, specify the name of the absolute path to the dictionary creation file that you modified in step 1.

4. Make a backup of the directory for storing synonym dictionary files.

   If you update the synonym dictionary, the relevant synonym dictionary files are updated. In case a failure occurs in a synonym dictionary file, make a backup of the directory for storing synonym dictionary files. If a failure occurs in a synonym dictionary file when you have not made a backup, you need to re-create the synonym dictionary file by using the `adbsyndict` command.

   The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

Note

When update of a synonym dictionary is completed in step 3, the `KFAA51509-I` message is output. (As many messages as the synonym dictionaries for which update is completed are output.) Identify the synonym dictionary name in the message, and confirm that update of the synonym dictionary is completed.

11.16.11 Tuning synonym search function

To tune synonym search function, consider and check the descriptions here.

(1) Considering defining text indexes

If no text index is defined for columns in which document data is stored, consider defining text indexes. By defining text indexes, you can reduce the number of pages to be accessed, so improvement of retrieval performance is expected.

(2) Checking whether text indexes are effectively used

Check the value for `Data_tidx_all_search_cnt` output to the access path statistical information. When this value is no less than 1, as this value is larger, the text indexes are less effectively used. In `Data_tidx_all_search_cnt`, the number of all data searches is output when data in a chunk is retrieved by using text indexes.

If an SQL statement that satisfies either of the following conditions is specified, modifying the SQL statement might improve the performance of synonym search.

- The scalar function `CONTAINS` for which synonym-search specification is specified, and the condition that uses the `LIKE` or `LIKE_REGEX` predicate are specified multiple times by using logical operator `OR`.
- The scalar function `CONTAINS` for which synonym-search specification is specified, is specified multiple times by using logical operator `OR`.

In this case, consider unifying individual retrieval results by specifying `UNION`, or performing retrieval without using a text index.

The following shows an example of modifying an SQL statement.

<Before modification>

```
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1","database")')>0
       OR CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1","application server")')>0
       OR ...
```

<After modification>

- When `UNION` is used:

```
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1","database")')>0
UNION DISTINCT
SELECT "TITLE" FROM "REPORTS"
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1","application server")')>0
UNION DISTINCT ...
```

- When no text index is used:

```
SELECT "TITLE" FROM "REPORTS" /*>> WITHOUT INDEX <<*/
    WHERE CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1","database")')>0
       OR CONTAINS("DOCUMENTS",'SYNONYM("Dictionary1","application server")')>0
       OR ...
```

(3) Checking the access time to synonym dictionary files

When synonym search is performed, the HADB server accesses synonym dictionary files. Therefore, performance improvement for synonym search is expected by creating only the directory for storing synonym dictionary files in the dedicated file system.

Check the value for `Syndict_file_access_time` output to the SQL statement statistical information. In `Syndict_file_access_time`, the time during which the HADB server is accessing the synonym dictionary files

is output. If `Syndict_file_access_time` occupies a large portion of the time for executing SQL statements, consider creating the directory for storing synonym dictionary files in the dedicated file system.

For details about how to change the directory for storing synonym dictionary files, see 11.16.13 Changing the directory for storing synonym dictionary files.

11.16.12 Checking the free space required for the directory for storing synonym dictionary files

When you register or update a synonym dictionary, check whether there is enough free space for the directory for storing synonym dictionary files. If the free space is insufficient, execution of the `adbsyndict` command results in an error.

You can use the following formula to determine the approximate value of the required free space.

Formula (kilobytes)

5 x *A* + 513 x *C* + 27 x *B* + 1,539 x *D*

A:

Total size (kilobytes) of the synonym list definition files for the synonym dictionaries for which `CASESENSITIVE` is specified for the correction search option

B:

Total size (kilobytes) of the synonym list definition files for the synonym dictionaries for which `CORRECTIONRULE` is specified for the correction search option

C:

Number of the synonym dictionaries for which `CASESENSITIVE` is specified for the correction search option

D:

Number of the synonym dictionaries for which `CORRECTIONRULE` is specified for the correction search option

As the result of the preceding calculation, if the free space is insufficient, execute the `adbsyndict` command to reduce the number of synonym dictionaries registered or updated at a time. The preceding formula calculates the free space that is temporarily required. If you reduce the number of synonym dictionaries registered or updated at a time, the free space that is temporarily required is reduced. Therefore, the `adbsyndict` command might be executed.

Note

For example, when you update 10 synonym dictionaries, 10 new synonym dictionary files are created, and then the old synonym dictionary files are deleted. Therefore, 20 synonym dictionary files temporarily exist in the directory for storing synonym dictionary files.

11.16.13 Changing the directory for storing synonym dictionary files

This subsection describes how to change the directory for storing synonym dictionary files.

Procedure:

1. Create the directory for storing synonym dictionary files that is to be used after the change.

For details about how to create this directory, see the following subsections of 11.16.1 Preparing for synonym search operations:

- (3) Estimating the size required for the directory for storing synonym dictionary files
- (4) Creating the directory for storing synonym dictionary files
- (5) Assigning permissions to the directory for storing synonym dictionary files

2. Execute the `adbstop` command to terminate the HADB server normally.
3. Change the value specified for the `adb_syndict_storage_path` operand in the server definition.

   For the `adb_syndict_storage_path` operand, specify the directory you created in step 1.
4. Execute the `adbstart` command to start the HADB server normally.
5. Execute the `adbsyndict` command to reregister all synonym dictionaries.

   Use the synonym list definition file and dictionary creation file that are saved, to reregister all synonym dictionaries.
6. Make a backup of the directory for storing synonym dictionary files.

   If you register a synonym dictionary, the relevant synonym dictionary files are added. In case a failure occurs in a synonym dictionary file, make a backup of the directory for storing synonym dictionary files. If a failure occurs in a synonym dictionary file when you have not made a backup, you need to re-create the synonym dictionary file by using the `adbsyndict` command.

   The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

> **Note**
>
> When registration of a synonym dictionary is completed in step 5, the `KFAA51509-I` message is output. (As many messages as the synonym dictionaries for which registration is completed are output.) Identify the synonym dictionary name in the message, and confirm that registration of the synonym dictionary is completed.

11.16.14 Rebuilding a synonym list definition file (when a synonym list definition file is lost)

By executing the `adbsyndict` command, you can rebuild a synonym list definition file. For example, if you have lost a synonym list definition file, rebuild the synonym list definition file.

Example:

Rebuild a synonym list definition file for synonym dictionary `Dictionary1`.

Procedure:

1. Execute the `adbsyndict` command to rebuild a synonym list definition file for synonym dictionary `Dictionary1`.

```
adbsyndict -n Dictionary1 -o /home/adbmanager/dictionary1_output.txt
```

   For the `-n` option, specify the synonym dictionary name.

   For the `-o` option, specify the output destination file name.
2. Open the synonym list definition file that was rebuilt, and check the contents.

```
database,data bank,DB↓
application server,AP server↓
```

Legend: ↓: Line feed

Note

- In a synonym list definition file that was rebuilt by the `adbsyndict` command, the following points might differ from a synonym list definition file that you created by yourself:
  - Each synonym is enclosed by enclosing characters (double quotation marks).
  - If one synonym is specified twice or more in a synonym group, only one of them is output.
  - The specification order of synonyms in a synonym group might be different.
  - The line break code is `X'0A'` (`LF`).
- When rebuilding of a synonym list definition file is completed in step 1, the `KFAA51529-I` message is output. Check the synonym dictionary name in the message to confirm that rebuilding of a synonym list definition file has been completed.

11.16.15 Specification rules for a synonym list definition file

This subsection describes the specification rules for a synonym list definition file.

(1) Specification example of a synonym list definition file

In this example, the following two synonym groups are registered in a synonym dictionary. The following describes an example of specifying the synonym list definition file.

- `database,data bank,DB`
- `application server,AP server`

<Specification example of a synonym list definition file>

```
database,data bank,DB↓
application server,AP server↓
```

Legend: ↓: Line feed

Explanation:

- If you perform a search by specifying one of `database`, `data bank`, and `DB` as the search-target character string, all of `database`, `data bank`, and `DB` are searched for as synonyms.
- If you perform a search by specifying one of `application server` and `AP server` as the search-target character string, all of `application server`, and `AP server` are searched for as synonyms.

(2) Specification rules for a synonym list definition file

- Specify a synonym group per line.
- Specify individual synonyms by delimiting with a comma (`,`).

- Input a line feed at the end of each line (at the end of each synonym group). Also, input a line feed at the last line of the synonym list definition file. In the preceding example, you need to input a line feed after `application server`.
- Use as the linefeed code one from `X'0A'` (`LF`), `X'0D0A'` (`CRLF`), or `X'00'`.
- If there is a line on which no synonym is specified (a line that has only a line feed), execution of the `adbsyndict` command results in an error.
- Create a synonym list definition file by using the character encoding specified for environment variable `ADBLANG`.
- You can specify a character string consisting of 1 to 1,000 characters for a synonym.
- You can specify 2 to 1,000 synonyms for a synonym group.
- You can specify no more than 1,000,000 synonym groups.
- You cannot register a synonym that contains a horizontal tab `X'09'` (`HT`), line feed character `X'0A'` (`LF`), or `X'00'`.
- You can specify double quotation marks (`"`) as enclosing characters. If a synonym contains `,`, enclose the synonym by the enclosing characters.
- Half-width spaces before and after the synonym character string are not deleted. For example, if "ΔabΔcΔdΔΔ" is specified, "ΔabΔcΔdΔΔ" is registered as a synonym. Δ indicates a half-width space.

(3) Relationship between synonym list definition files and synonym dictionaries

Depending on the search conditions or contents of the search-target document data, you can create multiple synonym dictionaries and use them differently. When you create multiple synonym dictionaries, create a synonym list definition file per synonym dictionary.

For example, if you perform the following retrievals, create two synonym list definition files and two synonym dictionaries:

1. Retrieving sentence data that contains general terms for database
2. Retrieving sentence data by narrowing down the database type (for example, retrieving sentence data regarding relational database, or retrieving sentence data regarding NoSQL database)

■ Specification of the synonym list definition file (`Terminology.txt`) for the preceding item 1

```
database,data bank,DB,relational database,RDB,NoSQL database,No-SQL database,cloud
 database↓
```

Legend: ↓: Line feed

Create synonym dictionary `Terminology` by using this synonym list definition file. When retrieving sentence data that contains general terms for database, use synonym dictionary `Terminology`.

■ Specification of the synonym list definition file (`Terminology_detail.txt`) for the preceding item 2

```
relational database,RDB,relational data base↓
NoSQL database,No-SQL database,cloud database↓
```

Legend: ↓: Line feed

Create synonym dictionary `Terminology_detail` by using this synonym list definition file. When retrieving sentence data regarding relational database or sentence data regarding NoSQL database, use synonym dictionary `Terminology_detail`.

11.16.16 Specification rules for a dictionary creation file

This subsection describes the specification rules for a dictionary creation file.

(1) Specification example of a dictionary creation file

The following describes a specification example of a dictionary creation file used when registering synonym dictionary `Dictionary2`.

<Specification example of a dictionary creation file>

```
Dictionary2,/home/adbmanager/dictionary2_synonym.txt,CORRECTIONRULE,general term↓
```

Legend: ↓: Line feed

Explanation:

The following four items are specified in the dictionary creation file. Specify individual items by delimiting with a comma (`,`).

- *synonym-dictionary-name* (Specify this on the first line.)
- *synonym-list-definition-file-name* (Specify this on the second line.)
- *notation-correction-option* (Specify this on the third line.)
- *comment* (Specify this on the fourth line.)

(2) Contents specified for individual items in a dictionary creation file

▪ First line

synonym-dictionary-name: ~**<character string> ((1 to 120 bytes))**

Specify the synonym dictionary name. We recommend that you specify a simple name for a synonym dictionary name because it is specified in an SQL statement.

You can specify a character string for a synonym dictionary name by using half-width alphabetic characters (`A` to `Z`, and `a` to `z`), half-width numeral characters (`0` to `9`), and underscores (_).

> **Important**
>
> If you specify an already registered synonym directory name, that synonym dictionary is reregistered. Therefore, when you register a new synonym dictionary, be careful not to specify a dictionary name that has already been registered. You can check whether there are already registered synonym dictionaries by retrieving the system table. For details about how to retrieve data from the system table, see (2) Checking the information about all synonym dictionaries in 11.16.4 Checking the information about synonym dictionaries.

▪ Second line

synonym-list-definition-file-name: ~**<OS path name> ((2 to 510 bytes))**

Specify the synonym list definition file name by using the absolute path name.

The following shows the specification rule for a synonym list definition file name:

- Control characters (`0x00` to `0x1f`, and `0x7f`) cannot be specified.

▪ Third line

notation-correction-option:

Specify whether to create a synonym dictionary that supports correction search. Specify either of the following:

- CORRECTIONRULE or CR

  Specify this when you create a synonym dictionary that supports correction search.

- CASESENSITIVE or CS

  Specify this when you create a synonym dictionary that does not support correction search.

You can use a synonym dictionary created by specifying CORRECTIONRULE even when correction search is not performed. Therefore, we recommend that you specify CORRECTIONRULE if you are uncertain whether to perform correction search when creating a synonym dictionary.

Note

- You can select whether to perform correction search in the specification of the scalar function CONTAINS when you perform a synonym search operation.
- If you change a synonym dictionary that does not support correction search to a synonym dictionary that supports correction search, you need to reregister that synonym dictionary.

Important

If the character encoding used on the HADB server is Shift-JIS (if the value specified for the environment variable ADBLANG is SJIS), correction search operations cannot be performed. Therefore, you cannot specify CORRECTIONRULE for the correction search option.

If you omit specifying the correction search option, CASESENSITIVE is assumed.

Example: when specification of the correction search option is omitted:

```
Dictionary3,/home/adbmanager/dictionary3_synonym.txt,,detailed term
```

▪ **Fourth line**

comment: ~<character string> ((0 to 1,024 bytes))

Specify a comment for the synonym dictionary.

If you are uncertain which synonym dictionary should be used for synonym search, display the list of synonym dictionaries to identify the synonym dictionary to be used. When you do so, you can identify the synonym dictionary by checking the comment specified here. Therefore, specify a comment that is relevant to the contents of the synonym dictionary.

Note

- To display the list of synonym dictionaries, retrieve the system table. For details, see (2) Checking the information about all synonym dictionaries in 11.16.4 Checking the information about synonym dictionaries.
- In the list of synonym dictionaries, sets of the synonym dictionary name and the comment specified here are displayed.

The following shows the specification rules for a comment:

- Control characters (0x00 to 0x1f, and 0x7f) cannot be specified.
- You can omit specifying a comment.

Example: when specification of a comment is omitted:

```
Dictionary2,/home/adbmanager/dictionary2_synonym.txt,CORRECTIONRULE,
```

(3) Specification rules for a dictionary creation file

- In a dictionary creation file, you can specify information about multiple synonym dictionaries. For details, see (4) Notes.
- The maximum number of synonym dictionaries that can be registered is 50.
- Information about one synonym dictionary is written on one line.
- Input a line break at the end of each line. Also, input a line break at the last line of the dictionary creation file. Use as the line break code one from `X'0A'` (`LF`), `X'0D0A'` (`CRLF`), or `X'00'`.
- If there is a line that has only a line break, that line is skipped when processing is executed.
- Create a dictionary creation file by using the character encoding specified for environment variable `ADBLANG`.
- You can specify double quotation marks (`"`) as enclosing characters. If a character string (for example, a character string for a comment) to be specified contains `,`, enclose the character string by the enclosing characters.

  Example:

```
Dictionary3,/home/adbmanager/dictionary2_synonym.txt,CORRECTIONRULE,"general term,
related to law"
```

(4) Notes

In a dictionary creation file, you can specify dictionary information about multiple synonym dictionaries as follows. In this case, you can register multiple synonym dictionaries by one execution of the `adbsyndict` command.

Example:

```
Dictionary1,/home/adbmanager/dictionary2_synonym.txt,CORRECTIONRULE,general term↓
Dictionary2,/home/adbmanager/dictionary3_synonym.txt,CASESENSITIVE,detailed term↓
```

Legend: ↓: Line feed

For example, if you perform the following operations, specify multiple synonym dictionaries as is done in the preceding example:

- When adding, modifying, or deleting synonyms in `Dictionary1`, the synonyms in `Dictionary2` also need to be added, modified, or deleted.

If you do not perform such operations, we recommend that you specify the dictionary information about only one synonym dictionary for one dictionary creation file.

11.17 Reorganizing system tables

This section describes reorganization of system tables.

For details about system tables, see C. System Tables.

11.17.1 Reason for reorganizing a system table

When an HADB user executes an SQL statement or command on a base table, the relevant system table (base table) might be automatically updated by the HADB server. During this operation, invalid row data might increase in the system table (base table). Invalid row data is not deleted automatically from the disk.

- If a row in a system table (base table) is deleted, the target row data becomes invalid.
- If a row in a system table (base table) is updated, the row data after update is added as a new row. The row data before update becomes invalid.

If invalid row data continues to increase in a system table (base table), free space of the disk that stores the system table (base table) becomes insufficient. Eventually, you will not be able to perform operations that cause update of the system table (base table). For example, you will not be able to import data by using background import or merge chunks.

Therefore, you need to regularly reorganize a system table (base table) to delete invalid row data. To reorganize a system table (base table), execute the `adbreorgsystemdata` command.

The following table describes the timing of when invalid row data increases in a system table (base table).

Table 11-9: Timing of when invalid row data increases in a system table (base table)

No.	Type of a system table (base table)	Operation that increases invalid row data in a system table (base table)
1	• `STATUS_TABLES` • `STATUS_COLUMNS` • `STATUS_INDEXES`	• Deleting a table or index by a definition SQL statement# • Collecting or deleting cost information by the `adbgetcst` command
2	`STATUS_CHUNKS`	• Deleting a multi-chunk table by a definition SQL statement • Deleting a chunk by the `PURGE CHUNK` statement • Batch deleting data stored in a multi-chunk table, by the `TRUNCATE TABLE` statement • Importing data to a multi-chunk table by using creation mode (when executing the `adbimport` command with the `-d` option specified) • Importing data to a multi-chunk table by using background import (when executing the `adbimport` command for which the `-b` option is specified and the `--status wait` option is not specified) • Importing data to a multi-chunk table by using addition mode and modifying the comment for a chunk (when executing the `adbimport` command with the `-m` option specified) • Re-creating an index by the `adbidxrebuild` command (when executed after the data import to a multi-chunk table by using background import was interrupted) • Merging chunks by the `adbmergechunk` command • Setting, modifying, or deleting the comment for a chunk by the `adbchgchunkcomment` command • Changing the chunk status by the `adbchgchunkstatus` command
3	`STATUS_SYNONYM_DICTIONARIES`	Registering, updating, or deleting a synonym dictionary by the `adbsyndict` command

#

When cost information is collected for the processing-target table or index.

11.17.2 Timing for reorganizing a system table

To avoid the shortage of free disk space due to increase of invalid row data, you need to execute the `adbreorgsystemdata` command to reorganize a system table (base table).

The timing for reorganizing a system table (base table) differs, depending on whether the system-table DB area file is created by using a regular file or block special file.

- **When using a regular file**

  If the free disk space for storing a system table (base table) becomes insufficient due to increase of invalid row data, no warning message is output from the HADB server.

  Therefore, before you start operations, consider and determine the timing for reorganizing the system table (base table) by the following procedure. Then, after you start operations, reorganize the system table (base table) at the timing you determined.

  > **Note**
  >
  > When a block special file is used, warning messages are output from the HADB server at the timing reorganization is required. Therefore, we recommend that you use a block special file when creating a system-table DB area file. For details about the procedure for changing a regular file to a block special file, see (1) Changing a data DB area file from a regular file to a block special file in 11.10.5 Changing the storage location of data DB area files.

  Procedure:

  1. Determine the frequency of reorganizing a system table (base table).

     Execute the `adbreorgsystemdata` command to determine the frequency of reorganizing a system table (base table). The basic guideline is every six months.

  2. Estimate the size of the system-table DB area.

     Estimate the size of the system-table DB area based on the frequency of reorganization that you determined in step 1. For details about estimation, see 5.12 Estimating the size of the system-table DB area.

  3. Create a database based on the estimation result.

     Based on the size of system-table DB area that you determined in step 2, prepare a disk and create a database.

  4. Regularly reorganize the system table (base table).

     After you start operations, based on the frequency of reorganization that you determined in step 1, execute the `adbreorgsystemdata` command.

- **When using a block special file**

  If the free disk space for storing a system table (base table) becomes insufficient due to increase of invalid row data, the following warning messages are output from the HADB server:

  - `KFAA61213-W` message
  - `KFAA61214-W` message

  When the warning messages are output, execute the `adbreorgsystemdata` command to reorganize the system table (base table).

Note that there is no problem if you reorganize a system table (base table) when it is suitable for your operations even if no warning message is output. In that case, we recommend that you execute the adbreorgsystemdata command about every six months.

11.17.3 Reorganizing a system table

To reorganize a system table (base table), execute the adbreorgsystemdata command.

For details about the adbreorgsystemdata command, see *adbreorgsystemdata (Reorganize System Table)* in the manual *HADB Command Reference*.

Important

While a system table (base table) is being reorganized, the following two datasets temporarily exist in the system-table DB area that stores the system table (base table):

- Dataset before reorganization
- Dataset after reorganization

Therefore, if there is not enough free disk space to store these two datasets, you cannot reorganize the system table (base table).

▪ **Execution example of the adbreorgsystemdata command**

In this example, system table (base table) STATUS_CHUNKS is reorganized. For the timeout time for the adbreorgsystemdata command, 300 seconds is specified.

```
adbreorgsystemdata --timeout 300
                   -c table
                   -f /home/adbmanager/reorg_file/unldpath.txt
                   -n STATUS_CHUNKS
```

Note

We recommend that you specify the --timeout option when executing the adbreorgsystemdata command.

Execution of the adbreorgsystemdata command usually finishes in several tens of seconds to a few minutes. However, if a special SQL statement or command is being executed, the adbreorgsystemdata command is placed in wait status. Therefore, execution time of the adbreorgsystemdata command might be longer. While the adbreorgsystemdata command is being executed, you cannot execute the adbimport or adbmergechunk command. For details about the wait status of the adbreorgsystemdata command, see 11.17.4 Reorganization of a system table and lock control.

If longer execution time of the adbreorgsystemdata command causes a problem in system operations, specify the --timeout option. If the adbreorgsystemdata command does not finish within the time specified for the --timeout option, the reorganization processing is canceled.

If the adbreorgsystemdata command is canceled, check the output message, and then re-execute the adbreorgsystemdata command.

11.17.4 Reorganization of a system table and lock control

This subsection describes the relationship between the adbreorgsystemdata command and lock control, and the wait status of the adbreorgsystemdata command.

(1) Relationship between the adbreorgsystemdata command and lock control

When you execute the adbreorgsystemdata command, the HADB server reserves locked resources according to the explanation in 2.10.4 Locked resources that are reserved and their lock modes.

While the adbreorgsystemdata command is being executed, the system-table DB area, which is a locked resource, is reserved in exclusive mode (EX). Therefore, you cannot execute the adbimport and adbmergechunk commands.

If any of the following SQL statements or command is being executed, the adbreorgsystemdata command is placed in wait status. (The KFAA80257-I message is output, and then reorganization processing is interrupted.) After the SQL statement or command that is being executed finishes, the processing for the adbreorgsystemdata command is resumed.

▪ SQL statements and command that place the adbreorgsystemdata command in wait status

- Retrieval SQL statement (that is executed on a system table (base table) or an archivable multi-chunk table#)
- Update SQL statement (that is executed on an archivable multi-chunk table#)
- adbexport command (that is executed on a system table (base table) or an archivable multi-chunk table#)
- adbdbstatus command with the --shared-lock option specified (executed for the system-table DB area)
- adbdbstatus command with the --shared-lock option specified (executed for the system table (base table) that is the target of adbreorgsystemdata command processing)
- adbdbstatus command with the --shared-lock option specified (executed for an index of the system table (base table) that is the target of adbreorgsystemdata command processing)

#

This case includes when the adbreorgsystemdata command is executed on the system table (base table) STATUS_CHUNKS. If you execute the adbreorgsystemdata command on a system table (base table) other than the STATUS_CHUNKS table, the adbreorgsystemdata command is not placed in wait status even while the preceding SQL statement or command is being executed on an archivable multi-chunk table.

If you want to release the wait status of the adbreorgsystemdata command without waiting for the SQL statement or command to finish, see (2) Releasing the wait status of the adbreorgsystemdata command in 11.17.4 Reorganization of a system table and lock control.

(2) Releasing the wait status of the adbreorgsystemdata command

If you want to release the wait status of the adbreorgsystemdata command without waiting for the SQL statement or command that causes the wait status to finish, you need to terminate the SQL statement or command.

To terminate the SQL statement or command that causes the wait status of the adbreorgsystemdata command, perform the following procedure.

Procedure:

1. Execute the adbls -d lock command.

Check the item CID of the following row:

- The row in which the ID of the table on which the adbreorgsystemdata command is executed is output to the item RESOURCEID

For details about the ID of the system table (base table) targeted by the adbreorgsystemdata command, see (1) List of system tables in C.1 System table overview.

2. Execute the adbls -d cnct command.

   Check the item CID of either of the following rows:

   - The row in which command is output to the item CLIENT_TYPE and the time output to the item CONNECT_TIME is earlier than the time the KFAA80256-I message was output. (This message was output as the result of the previous execution of the adbreorgsystemdata command.)
   - The row in which command is not output to the item CLIENT_TYPE and which also satisfies the following condition. The execution time of the SQL statement that is determined based on the value (microseconds) output to the item SQL_ELAPSED_TIME is earlier than the time the KFAA80256-I message was output. (This message was output as a result of the previous execution of the adbreorgsystemdata command.)

3. Execute the adbcancel command.

   The item CID that was checked in step 1 and step 2 indicates the connection ID of the SQL statement or command that caused the wait status of the adbreorgsystemdata command. If you execute the adbcancel command with the connection ID specified, the wait status of the adbreorgsystemdata command will be released.

For details about these commands, see the manual *HADB Command Reference*.

11.17.5 Checking the status and amount of use of system tables

This subsection describes how to check the status, amount of use, and storage efficiency of system tables (base tables).

(1) Checking the status and amount of use of system tables (base tables)

To check the status and amount of use of system tables (base tables), execute the adbdbstatus command to output the table summary information.

For details about the adbdbstatus command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

Example of the command to be executed

In this example, summary information of system table (base table) STATUS_CHUNKS is output.

```
adbdbstatus -d summary -c table -n "HADB"."STATUS_CHUNKS" -S M --shared-lock
```

If you execute the command, summary information of system table HADB.STATUS_CHUNKS is output to the standard output. Check the following items in the output result of the adbdbstatus command.

When checking whether data before reorganization remains

Check the value output to Pending_delete_chunks. If the value output to Pending_delete_chunks is 1, the data before reorganization (deletion-pending chunk) remains.

Re-executing the adbreorgsystemdata command deletes the data before reorganization.

When checking the amount of use of system tables (base tables)

Check the value output to `MB_Used_pages`. You can check the size (in megabytes) of the area used by the system tables (base tables) that are stored in the system-table DB area.

Note that, if the `--shared-lock` option is specified, the value output under `MB_Used_pages` does not contain the area used by the deletion-pending chunk. If, in the value output under `MB_Used_pages`, you want to include the area used by the deletion-pending chunk, do not specify the `--shared-lock` option.

(2) Checking the storage efficiency of a system table (base table)

To check the storage efficiency of a system table (base table), you need to use SQL tracing.

Procedure:

1. Specify the settings so that SQL trace information is output.

   Use SQL tracing so that the following SQL trace information is output:

   - Access path information
   - Access path statistical information

   For details about output of the SQL trace information, see 10.11.5 Preparations for outputting SQL trace information.

   > **Note**
   >
   > If `Y` is specified for the following operands in the server definition, access path information and access path statistical information are output:
   >
   > - `adb_sql_trc_out` operand
   > - `adb_sql_trc_accesspath` operand
   >
   > For details about the operands in the server definition, see 7.2.5 Operands related to SQL statements (set format).
   >
   > If the HADB server has started, use the `adbchgsqltrc` command. For details about the `adbchgsqltrc` command, see *adbchgsqltrc (Start or Stop Output of SQL Trace Information)* in the manual *HADB Command Reference*.

2. Use an SQL statement to check the number of rows in a system table.

   On the system table for which you check the storage efficiency, execute an SQL statement used for outputting the number of rows by an HADB user with the `DBA` privilege. The number of rows that is output as the result of execution of the SQL statement is the number of rows in the system table.

   The following shows SQL statements corresponding to individual system tables.

   ▪ SQL statement used to output the number of rows in the STATUS_TABLES table

   ```
   SELECT COUNT(COLLECT_TIME)
     FROM MASTER.STATUS_TABLES
   ```

   ▪ SQL statement used to output the number of rows in the STATUS_COLUMNS table

   ```
   SELECT COUNT(DATA_TYPE_CODE)
     FROM MASTER.STATUS_COLUMNS
   ```

   ▪ SQL statement used to output the number of rows in the STATUS_INDEXES table

```
SELECT COUNT(COLLECT_TIME)
  FROM MASTER.STATUS_INDEXES
```

▪ SQL statement used to output the number of rows in the STATUS_CHUNKS table

```
SELECT COUNT(NVL(CREATE_TIME,CURRENT_TIMESTAMP))
  FROM MASTER.STATUS_CHUNKS
```

▪ SQL statement used to output the number of rows in the STATUS_SYNONYM_DICTIONARIES table

```
SELECT COUNT(CREATE_TIME)
  FROM MASTER.STATUS_SYNONYM_DICTIONARIES
```

3. Check the number of rows for the invalid row data.

   From the access path information in the SQL trace information output by execution of the SQL statement in step 2, check the number of rows for the invalid row data in the system table. Check the value for `Data_deleted_rows_cnt` in the access path information that satisfies the following conditions:

   - The schema name is `HADB`.
   - The table name is the system table name specified in the SQL statement executed in step 2.

   The value for `Data_deleted_rows_cnt` that satisfies the preceding conditions is the number of rows for the invalid row data in the relevant system table.

4. Determine the storage efficiency of the system table (base table).

   Determine the storage efficiency of the relevant system table (base table) based on the results of steps 2 and 3. The following shows the formula:

   Formula

   ```
   Storage efficiency of a system table (base table) =
     result of step 2 / (result of step 2 + result of step 3)
   ```

   If the value obtained from the formula is close to 0, the storage efficiency of the system table (base table) can be determined to be poor. Execute the `adbreorgsystemdata` command to reorganize the system table (base table).

11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables)

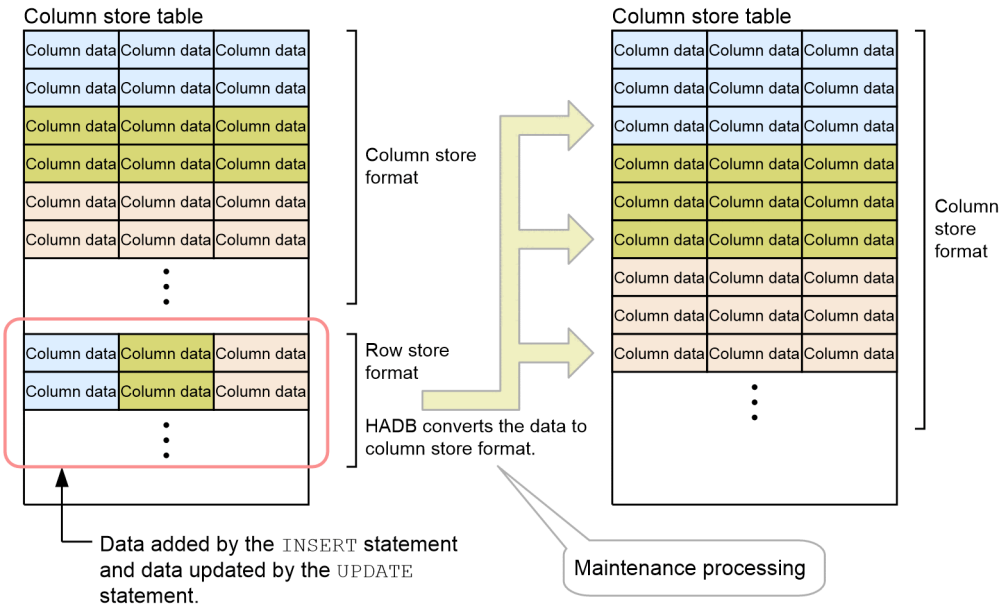
This section explains how to use a facility that maintains the retrieval performance for column store tables (the updated-row columnizing facility).

11.18.1 Overview of the updated-row columnizing facility

If the `INSERT` or `UPDATE` statement is executed for a column store table, the added or updated data is stored in the column store table in row store format. For a column store table whose data is stored in row store format, the retrieval performance might be degraded in processes that exploit the characteristics of column store tables. These processes include data retrieval in a specific column and summation of values in a specific column. The data compression rate might also be degraded. If the *updated-row columnizing facility* is used and data is stored in row store format in a column store table, HADB automatically converts the data into column store format. This prevents the retrieval performance for column store tables and the data compression rate from being degraded. In situations where the `INSERT` or `UPDATE` statement might be executed for a column store table, use the updated-row columnizing facility.

The following figure provides an overview of the updated-row columnizing facility.

Figure 11-46: Overview of the updated-row columnizing facility



■ Maintenance processing

The processing that converts data stored in a column store table from row store format to column store format is called *maintenance processing*. The maintenance processing consists of three processing stages: analysis, re-allocation, and reuse. For details about the maintenance processing, see 11.18.7 Details of the maintenance processing.

Important

- The maintenance processing is not performed while a transaction or command is running, thus preventing the processing performance of the transaction or command from being affected. If a transaction or command starts while maintenance processing is in progress, the maintenance

processing is interrupted. In this case, the maintenance processing is restarted after the processing of the transaction or command is completed.

Note that "command" here refers to a command that connects to the HADB server. For details about the commands that connect to the HADB server, see *List of commands* in *List of Commands and Common Rules* in the manual *HADB Command Reference*.

- In a system in which transactions or commands are continuously running, time for executing the maintenance processing cannot be secured. In such a system, do not use the updated-row columnizing facility.

■ **Relationship with reorganization of tables or chunks**

If the updated-row columnizing facility is used, data in row store format is converted to column store format. However, the following types of row data that has become invalid are not deleted:

- Update-source row data that was updated by using the `UPDATE` statement
- Row data that was deleted by using the `DELETE` statement

To delete row data that has become invalid, you must perform table or chunk reorganization. If you perform table or chunk reorganization, the pages that store the row data that has become invalid become reusable.

If you use the `INSERT` or `UPDATE` statement to add or update a large amount of data at one time, many row-data segments will be created. In such a case, you cannot prevent degradation of retrieval performance or data increase in the database even by using the updated-row columnizing facility. Therefore, if you have added or updated a large amount of data by using the `INSERT` or `UPDATE` statement at one time, consider executing table or chunk reorganization.

Note

We recommend that you use both the updated-row columnizing facility and table or chunk reorganization. For an example of this type of operation, see 11.4.23 Operation taking background import and chunks into consideration (Example 1: Adding and deleting data on a regular basis).

■ **Restrictions**

The updated-row columnizing facility is not applied to column store tables for which B-tree indexes are defined.

11.18.2 Preparation tasks

This subsection explains the preparation tasks required for using the updated-row columnizing facility.

(1) Checking the column store tables to which the facility is applied

The updated-row columnizing facility is not applied to column store tables for which B-tree indexes are defined. Before you enable the updated-row columnizing facility, see (38) Checking the tables subject to the updated-row columnizing facility (checking the column store tables for which no B-tree indexes are defined) in B.22 Searching a dictionary table to check the column store tables to which the updated-row columnizing facility is applied.

In situations where the `INSERT` or `UPDATE` statement might be executed for column store tables to which the updated-row columnizing facility is applied, use the updated-row columnizing facility.

(2) Estimating the memory requirements for the HADB server

If you use the updated-row columnizing facility, re-estimate the memory requirements for the HADB server by referring to the following subsections:

- (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.
- (2) Determining the real thread private memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

(3) Adding messages to be monitored

You can use the following messages to check whether maintenance processing has stopped for any reason. Add the following messages to the monitoring targets:

- KFAA41220-W

  This message is output if the system fails in the processing that starts I/O control of files used by the updated-row columnizing facility.
- KFAA51277-I

  This message is output if the maintenance processing has not been performed for 24 hours.
- KFAA51280-W

  This message is output if an error occurs during the maintenance processing.

■ Message output example

When error messages related to the updated-row columnizing facility are output to a message log file, the following messages are output in pairs: a message indicating the cause of the error and a message indicating that an error occurred in the updated-row columnizing facility.

Example:

```
2019/06/24 15:13:28 ... 00000000009999000001 KFAA40007-E message-text    ...1
2019/06/24 15:13:28 ... 00000000009999000001 KFAA51280-W message-text    ...2
```

Explanation:

1. This message indicates the cause of the error. This message indicates that memory shortage occurred. Take measures as shown in this message.
2. This message indicates that an error occurred in the maintenance processing.

In the preceding example, 9999000001 (underscored portions) is output as a connection sequence number for messages related to the updated-row columnizing facility.

(4) Enabling the updated-row columnizing facility

To enable the updated-row columnizing facility, execute the adbcolumnize command. If the updated-row columnizing facility is enabled, the maintenance processing is performed.

Command execution example

```
adbcolumnize --start
```

Confirm that the updated-row columnizing facility is enabled as explained in 11.18.3 How to check the status of the updated-row columnizing facility (whether the facility is enabled or disabled).

Note

- The status of the updated-row columnizing facility (whether the facility is enabled or disabled) is maintained even when the HADB server is restarted.
- The status of the updated-row columnizing facility (whether the facility is enabled or disabled) is also maintained when a master node switchover is performed by using the multi-node function or when a host switchover occurs in a cold standby configuration.

11.18.3 How to check the status of the updated-row columnizing facility (whether the facility is enabled or disabled)

To check the status of the updated-row columnizing facility (whether the facility is enabled or disabled), execute the `adbcolumnize` command.

Command execution example

```
adbcolumnize -d
```

Example of the command execution result

```
STATUS
ACTIVE
```

The status of the updated-row columnizing facility is displayed in the `STATUS` column. If `ACTIVE` is displayed, the updated-row columnizing facility is enabled. If `INACTIVE` is displayed, the updated-row columnizing facility is disabled.

11.18.4 Cases where the updated-row columnizing facility must be disabled temporarily

Temporarily disable the updated-row columnizing facility in the following cases:

- When creating a backup

  When data in row store format is converted to column store format by the updated-row columnizing facility, a database update occurs. If you do not want the database to be updated while you are creating a backup, temporarily disable the updated-row columnizing facility.
- When you want to have no impact on the execution time of an SQL statement or command

  If the updated-row columnizing facility is enabled, the execution time of an SQL statement or command might be adversely affected slightly (by about 0.5 seconds). To have no impact on the execution time, temporarily disable the updated-row columnizing facility.

  For example, in the following cases, consider whether the updated-row columnizing facility is to be disabled:

  - When you execute an SQL statement or command that ends in a short time (a few seconds to a few minutes) and you think that the facility greatly affects the execution time)
  - When you execute an SQL statement or command whose execution time must not be affected
  - When you perform batch processing or other processing that must be completed in a predefined time period

Note

- A command in this context refers to a command that connects to the HADB server.
- A 0.5-second slowdown is indicated only as a guide. The actual slowdown time differs depending on the condition under which the processing runs. If an SQL statement or command is executed while the maintenance processing is in progress, the maintenance processing is interrupted, and execution of the SQL statement or command also stops until the maintenance processing starts again. This takes about 0.5 seconds.
- If a hard disk rather than a flush storage device is used for the DB area or a regular file rather than a block special file is used for the DB area, it might take a few seconds or more to interrupt the maintenance processing.

To disable the updated-row columnizing facility, execute the `adbcolumnize` command.

Command execution example

```
adbcolumnize --stop
```

Confirm that the updated-row columnizing facility is disabled as explained in 11.18.3 How to check the status of the updated-row columnizing facility (whether the facility is enabled or disabled).

11.18.5 Action to be taken when an error related to the updated-row columnizing facility occurs

If an error related to the updated-row columnizing facility occurs, either of the following messages is output. How to take action if these messages are output is explained later.

- `KFAA51280-W`

  This message is output if an error occurs during the maintenance processing.
- `KFAA41220-W`

  This message is output if an error occurs in the processing that starts I/O control of files used by the updated-row columnizing facility.

(1) If an error occurs during the maintenance processing (KFAA51280-W)

If an error occurs during the maintenance processing, the `KFAA51280-W` message is output to a message log file. In this case, handle the error by referring to the action indicated in the `KFAA51280-W` message.

If you leave the updated-row columnizing facility enabled without handling the error, the maintenance processing is re-executed every six hours, and therefore the same error occurs again (the `KFAA51280-W` message is output again).

(2) If an error occurs in the processing that starts I/O control of files used by the updated-row columnizing facility (KFAA41220-W)

When the HADB server starts, the system performs the processing that starts I/O control of files used by the updated-row columnizing facility. If an error occurs while the system is performing the processing that starts I/O control of files used by the updated-row columnizing facility, the `KFAA41220-W` message is output to a message log file. In this case, handle the error by referring to the action indicated in the `KFAA41220-W` message. If you leave the updated-row

columnizing facility enabled without handling the error, the maintenance processing is re-executed every six hours. At this time, the system again attempts to perform the processing that starts I/O control of files used by the updated-row columnizing facility. Therefore, the same error occurs again (the KFAA41220-W message is output again).

11.18.6 Action to be taken if the maintenance processing is not performed

The maintenance processing is not performed when any of the following conditions are met:

- A transaction or a command that connects to the HADB server is running.
- The HADB server is operating in quiescence mode.
- A node is returning to a multi-node configuration (if the multi-node function is used).

If the maintenance processing is not performed for 24 hours, the KFAA51277-I message is output. The KFAA51277-I message will be output every 24 hours unless the maintenance processing is performed.

The following explains the action to be taken for each of the preceding conditions.

(1) If a transaction or a command that connects to the HADB server is running

If a transaction or a command that connects to the HADB server is running, the maintenance processing is not performed to prevent the processing performance of the transaction or command from being affected. If a transaction or command is started while the maintenance processing is in progress, the maintenance processing is interrupted.

If the maintenance processing is not performed for a long time or if you expect that the maintenance processing will not be performed for a long time, periodically check whether reorganization of the column store table is necessary. If reorganization is necessary, perform reorganization. For details about how to check whether reorganization of the column store table is necessary, see the following subsections:

- (3) If the single-chunk table is a column store table in 11.1.9 Checking whether a single-chunk table needs to be reorganized.
- (b) In a case where the multi-chunk table is a column store table in (2) How to check whether reorganization is necessary in 11.4.13 Checking whether a multi-chunk table needs to be reorganized

(2) If the HADB server is operating in quiescence mode

If the HADB server is operating in quiescence mode, the maintenance processing is not performed, because a database update is suppressed. The maintenance processing will be performed if you change the HADB server operation mode from quiescence mode to another mode.

For details about how to check the operation mode and how to change it, see 10.2.3 HADB server operation modes.

(3) If a node is returning to a multi-node configuration (if the multi-node function is used)

The maintenance processing is not performed while a node is returning to a multi-node configuration. The maintenance processing is performed when the node has returned to a multi-node configuration. For details about the process in which a node returns to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

11.18.7 Details of the maintenance processing

This subsection explains the maintenance processing of the updated-row columnizing facility.

(1) Overview of the maintenance processing

The maintenance processing consists of the following three processing stages:

- Analysis

  How the data in row store format is stored in the column store table is analyzed. If HADB judges that re-allocation is necessary, the data is re-allocated so that it becomes reusable.

  HADB controls the time when analysis processing starts.

- Re-allocation

  Data that is stored in a column store table in row store format is converted to column store format. The converted data is stored in a segment that is dedicated to storing data in column store format (a column-data segment).

- Reuse

  The row-store-format data that was used as the source data for conversion to column store format is deleted so that the pages that store the row-store-format data become reusable. The pages that became reusable are reused when data added or updated by using the `INSERT` or `UPDATE` statement is stored in row store format.

> **Note**
>
> - Non-updatable column store tables are not subject to the maintenance processing.
> - Column store tables whose range indexes are in unfinished status are not subject to the maintenance processing.

(2) Cases where the maintenance processing is interrupted

The maintenance processing is interrupted in the following cases:

- When a transaction is started or when a command that connects to the HADB server is executed

  If a transaction is started or a command that connects to the HADB server is executed while the maintenance processing is in progress, the maintenance processing is interrupted. The maintenance processing is restarted after the transaction or command ends.

- When a connection with the HADB server is established

  When a connection with the HADB server is established, a transaction is temporarily started to authorize the user and check the privilege. Therefore, the maintenance processing is interrupted if it is in progress. This transaction ends when the connection with the HADB server is established. After the transaction ends, the maintenance processing is restarted.

- When the `adbcolumnize --stop` command is executed

  If the updated-row columnizing facility is disabled by executing the `adbcolumnize --stop` command while the maintenance processing is in progress, the maintenance processing is interrupted. The maintenance processing is restarted when the updated-row columnizing facility is enabled by executing the `adbcolumnize --start` command.

- When the `adbstop` command is executed

  If the HADB server is terminated by executing the `adbstop` command while the maintenance processing is in progress, the maintenance processing is interrupted. The maintenance processing is restarted when the HADB server is started by executing the `adbstart` command.

- When the `adbchgsrvmode --quiescence` command is executed

  If the HADB server operation mode is changed to quiescence mode by executing the `adbchgsrvmode --quiescence` command while the maintenance processing is in progress, the maintenance processing is interrupted. The maintenance processing is restarted when the HADB server operation mode is changed to a mode other than quiescence mode by executing the `adbchgsrvmode` command.

- When a process that returns a node to a multi-node configuration is started (if the multi-node function is used)

  If a process that returns a node to a multi-node configuration is started while the maintenance processing is in progress, the maintenance processing is interrupted. The maintenance processing is restarted when the process that returns the node to the multi-node configuration is completed.

12 Audit Trail Facility Operations

This chapter explains the operation of the audit trail facility.

12.1 Matters to consider when using the audit trail facility

This section describes the matters you need to consider before using the audit trail facility.

12.1.1 Considering audit target definitions (selecting events for which to output audit trails)

The audit trail facility can output audit trails for any audit target event. However, outputting an audit trail for every event generates an extremely large volume of audit trail data, which limits how efficiently the auditors can work. For this reason, you need to decide which events will be of interest to an auditor based on the intention of the audit process. The following table shows examples of the events that might be of interest to an auditor conducting the audit process with a certain intention.

Table 12-1: Intention of auditing and examples of events subject to auditing

Intention of auditing	Examples of events subject to auditing	Type of audit target event
Finding out whether the configuration of the HADB server has been modified without authorization	Starting and terminating the HADB server	Mandatory audit event
Finding out whether an unauthorized operation has been performed with respect to the audit trail facility	Performing operations in relation to the audit trail facility	Mandatory audit event
Finding out whether any suspicious connections have been made to the HADB server	Connecting to the HADB server	Optional audit event
Finding out whether data has been tampered with or removed	Creating, searching, updating, and deleting tables	Optional audit event
Finding out whether use of the system complies with the security policy (such as changing passwords regularly)	Changing the password of an HADB user	Optional audit event

A mandatory audit event is an event for which an audit trail will always be output as long as the audit trail facility is enabled. Optional audit events differ in that the auditor can select whether an audit trail is output. If you want to output audit trails for optional audit events, you need to define those events as audit targets by using the `CREATE AUDIT` statement.

Based on the intention of the audit process, decide whether to audit only mandatory audit events, or to also include optional audit events.

For lists of audit target events (mandatory audit events and optional audit events), see 12.9.1 List of audit target events and output items.

12.1.2 Designing the disks used by the audit trail facility

This section explains how to design the disks (storage areas) used by the audit trail facility.

The audit trail facility uses the following directories:

- Audit trail directory
- Audit trail storage directory

- Audit trail long-term storage directory

We recommend that you create each directory on a dedicated disk (storage area). If you create them all on the same disk (storage area), I/O operations are concentrated on that disk (storage area), potentially degrading the processing performance of the HADB server.

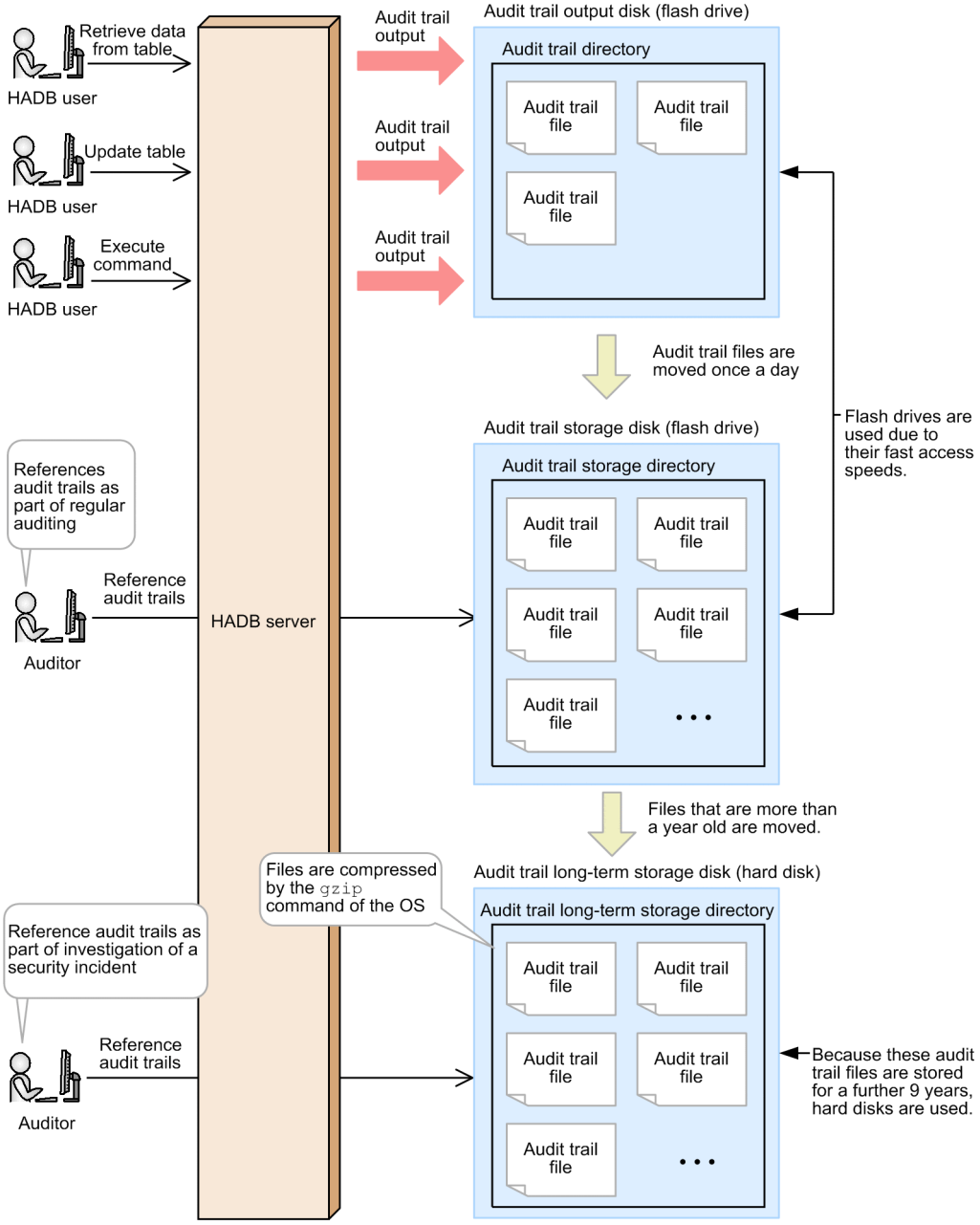
Note that the process of referencing audit trail information does not involve writing to the audit trail file. Therefore, we recommend that you create the following two directories on write-once disks (storage areas). An advantage of this approach is that the audit trail files on write-once disks (storage areas) cannot be tampered with.

- Audit trail storage directory
- Audit trail long-term storage directory

In selecting the type of disk (storage area) on which to store each directory, we recommend that you consider factors including how often the data is referenced and how long the data is retained. For example, the audit trail files in the audit trail storage directory are likely to be referenced quite frequently. For this purpose, you might consider using a flash drive which offers faster access. Because the audit trail files on the audit trail long-term storage directory are accessed infrequently and stored for a long time, a hard disk or magnetic tape might be more suitable than a flash drive.

The following figure shows an example configuration of the disks (storage areas) used by the audit trail facility. In this example, audit trail files are kept in the audit trail storage directory for one year. Those audit trail files are then stored in the audit trail long-term storage directory for a further nine years.

Figure 12-1: Example disk (storage area) configuration used by audit trail facility



For details about how to determine the space requirements of the disks (storage areas) used by the audit trail facility, see the following subsections:

- 12.1.3 Estimating the size of audit trail data
- 12.1.4 Estimating the compressibility of audit trail files
- 12.1.5 Estimating the size of directories used by the audit trail facility

12.1.3 Estimating the size of audit trail data

To estimate the disk space required by the audit trail facility, you first need to determine how much audit trail data the HADB server will output.

Use one of the following methods to determine the size of the audit trail data:

1. Check how much audit trail data is output in a small-scale test environment with the audit trail facility enabled
2. Estimate the amount of audit trail data based on the expected use of the HADB server and the database operations that are likely to be performed

The amount of audit trail data that is generated varies widely depending on the HADB server environment and on the SQL statements, commands, and other operations executed by users. For this reason, we recommend that you use the first of these methods. If you are unable to prepare a test environment, use the second method. When using the second method, determine the size of audit trail data according to the following table:

Table 12-2: Estimated size of audit trail data

No.	Example of audit target event	Estimated size of audit trail data (KB)
1	HADB server startup	4
2	HADB server termination	1
3	Connection to HADB server#	1
4	Disconnection from HADB server#	1
5	SQL statement execution	2 + size-of-SQL-statement
6	Command execution	1

#

Generated when the corresponding command is executed.

The following is an example of using the second method to determine the size of audit trail data:

■ **Example of using method 2 to determine the size of audit trail data**

Suppose that the HADB server is being used under the following conditions:

- The HADB server is restarted once per day (a process that involves stopping and then starting the HADB server)
- A 2 KB SQL statement is executed twice per connection (with one connection occurring each second)
- Audit trails are output for mandatory audit events and optional audit events

The estimated size of audit trail data for one day (86,400 seconds) of activity under these conditions is as follows:

Formula

```
estimated-size-of-audit-trail-data =
    4 KB + 1 KB + (1 KB + (2 KB + 2 KB) × 2 + 1 KB) × 86,400 seconds
  = 864,005 KB/day
```

The estimated size of the audit trail data output per day under these conditions is 864,005 KB (or approximately 850 MB).

12.1.4 Estimating the compressibility of audit trail files

You can use the `gzip` command of the OS to compress audit trail files. This allows you to reduce the file size of the audit trail files stored in the following directories:

- Audit trail storage directory
- Audit trail long-term storage directory

To estimate the disk space required by the audit trail facility when the `gzip` command will be used to compress audit trail files, you first need to determine the compressibility of the audit trail files.

Use one of the following methods to determine the compressibility of the audit trail files:

1. Use the `gzip` command to compress an audit trail file output in a small-scale test environment with the audit trail facility enabled, and check the compression ratio of the resulting file
2. Assume a compressibility of 0.25 (that is, audit trail files are compressed to one quarter of their original size)

The compressibility of audit trail files varies widely depending on the HADB server environment and on the SQL statements, commands, and other operations executed by users. For this reason, we recommend that you use the first of these methods. If you are unable to prepare a test environment, use the second method.

12.1.5 Estimating the size of directories used by the audit trail facility

Estimate the size of the directories used by the audit trail facility based on the results of the estimations in 12.1.3 Estimating the size of audit trail data and 12.1.4 Estimating the compressibility of audit trail files. You also need to prepare the disks where the directories used by the audit trail facility will be created. You need to estimate the size of the following three directories:

- Audit trail directory
- Audit trail storage directory
- Audit trail long-term storage directory

The size of the output audit trail data and compressibility of audit trail files vary widely depending on the HADB server environment, and on the SQL statements, commands, and other operations executed by users. For this reason, you need to ensure the disks you prepare offer sufficient leeway over and above the values you estimate. Even if you use a disk with plenty of space, you must always keep an eye on the amount of free space on the disk while using the audit trail facility.

Take particular care to ensure that the disk containing the audit trail directory does not run out of space. The behavior of the HADB server when the disk that contains the audit trail directory runs out of space depends on which of the approaches you select in 12.1.7 Considering the approach to take when attempts to write to the audit trail file fail. You must therefore ensure that enough disk space is available to accommodate not only the data generated during normal operation, but also any data generated in relation to sudden and unexpected activity.

The following table shows an example of estimating the size of the directories used by the audit trail facility.

■ Conditions for estimation examples

- The audit trail facility outputs 864,005 KB (approximately 850 MB) of audit trail data per day to the audit trail file
- The audit trail files in the audit trail directory are moved to the audit trail storage directory once a day
- Audit trail files are kept for a total of 10 years

  The audit trail files are kept in the audit trail storage directory for one year.

  Those audit trail files are then stored in the audit trail long-term storage directory for a further nine years.
- The audit trail files in the audit trail long-term storage directory are compressed using the `gzip` OS command

  The compressibility of the audit trail files is 0.25.
- The size estimated for each directory is the total of the minimum space required for the directory (required space) plus a certain amount of leeway (reserve space)

Table 12-3: Example of estimating size of directories used by audit trail facility

No.	Directory type	Storage period for audit trail files	Required space (GB)	Reserve space (GB)	Total space (GB)
1	Audit trail directory	1 day	0.9	1.8 (equivalent to 2 days)	2.7
2	Audit trail storage directory	1 year (365 days)	329	54 (equivalent to 2 months)	383
3	Audit trail long-term storage directory	9 year (3,285 days)	740[#1, #2]	83 (equivalent to 1 year)[#1]	823

#1

This is the total size of the audit trail files after compression.

#2

The total size of the uncompressed audit trail files is approximately 2,957 GB.

In this example, the size of the disk you provide for each directory must be at least the total space estimated in the preceding table.

12.1.6 Appointing auditors

When using the audit trail facility, HADB requires users (auditors) to be appointed to the following positions of responsibility:

- Audit trail facility administrator

  A person responsible for operating the audit trail facility. To execute the `adbaudittrail` command that is used to operate the audit trail facility, this role requires the OS account of an OS user who belongs to the HADB administrators group.

  This role also requires an HADB user account with the audit admin privilege.

- Person responsible for auditing

  A person who is responsible for auditing database usage by referencing audit trail information.

  Because audit trails can be referenced from non-server machines, the person responsible for auditing is not required to have an OS account on a server machine. The only time this role requires the OS account of an OS user who belongs to the HADB administrators group is when the person responsible for auditing references audit trails on a server machine.

  This role also requires an HADB user account with the audit viewer privilege.

  If you intend to also audit the activity of the person responsible for auditing, you can appoint several persons responsible for auditing and have them audit each other.

  Note

  When incorporating external auditing by an external person responsible for auditing, we recommend that the external person responsible for auditing have an HADB user account with the audit viewer privilege.

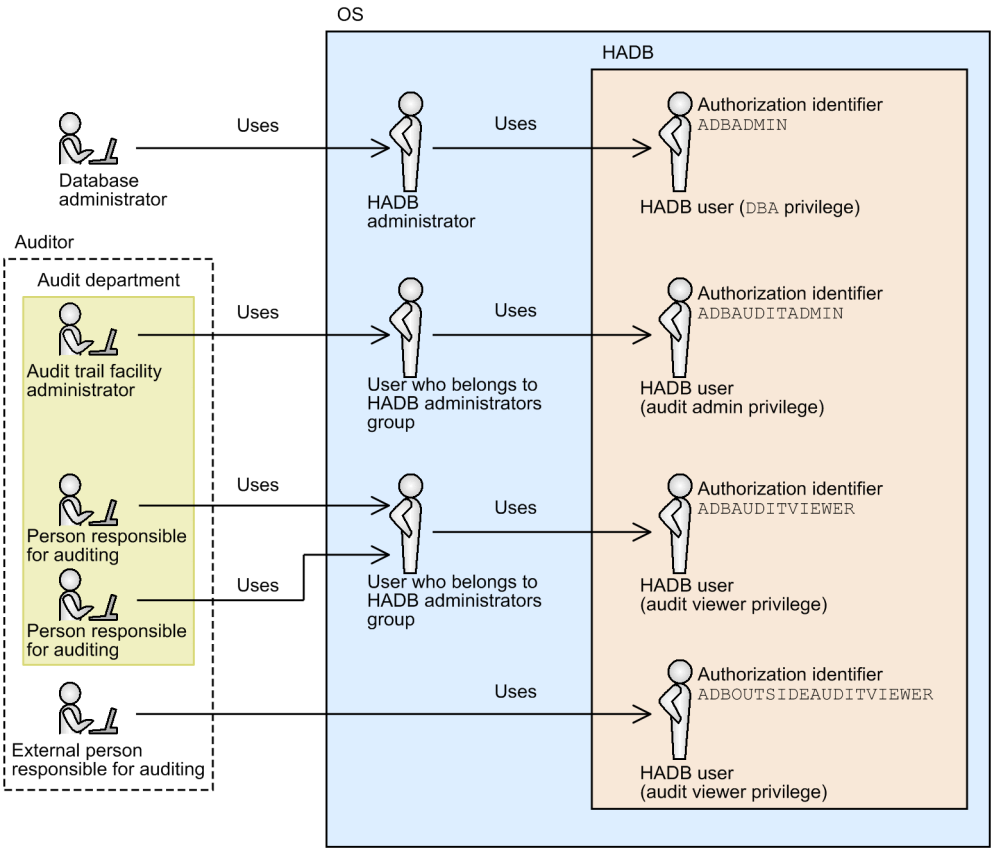
Although you can have one user serve as both audit trail facility administrator and person responsible for auditing, we recommend that the roles are filled by different people.

Important

- Do not appoint as an auditor a user who also serves as the database administrator (an HADB user who has the `DBA` privilege).
- It is preferable that auditors are responsible only for work related to auditing. We also recommend that you keep the HADB user accounts used for auditing work separate from those used for other work, by creating an HADB user used exclusively for auditing.

The following figure shows an example of the user accounts used with the audit trail facility.

Figure 12-2: Example of user accounts used with audit trail facility



Explanation

- In this example, the audit trail facility administrator associated with the audit department has OS and HADB (`ADBAUDITADMIN`) user accounts to use only when operating the audit trail facility.

  In this example, the persons responsible for auditing associated with the audit department share OS and HADB (`ADBAUDITVIEWER` user accounts used only for auditing.
- An external person responsible for auditing can only view audit trail data. Therefore, external persons responsible for auditing do not need to have an OS account on a server machine, and will carry out their work from a machine other than a server. An HADB user account with the audit viewer privilege (`ADBOUTSIDEAUDITVIEWER`) is created for use only by the external person responsible for auditing.

12.1.7 Considering the approach to take when attempts to write to the audit trail file fail

You need to consider whether to stop the HADB server if an attempt to write to the audit trail file fails for either of the following reasons:

- The disk containing the audit trail directory is full
- A failure has occurred on the disk containing the audit trail directory

Select either of the following as the processing method for situations where an attempt to write to the audit trail file fails. Specify the method you select in the `--write-error` option of the `adbauditttrail --start` command.

■ Terminate the HADB server

The HADB server is terminated if audit trail data cannot be written to the audit trail file. The termination mode of the HADB server in this scenario is abnormal termination.

Select this approach if you want to prioritize the recording of HADB user activity as audit trail data over the continued operation of the HADB server. This approach minimizes gaps in the acquisition of audit trail data.

To terminate the HADB server, specify `DOWN` in the `--write-error` option. Alternatively, you can omit the `--write-error` option altogether.

■ Continue HADB server operation

The HADB server does not terminate if audit trail data cannot be written to the audit trail file. The audit trail data that could not be written to the audit trail file is discarded, and the HADB server continues operating.

Select this approach if you want to prioritize the continued operation of the HADB server over the recording of HADB user activity as audit trail data.

To continue HADB server operation, specify `FAILSOFT` in the `--write-error` option. When `FAILSOFT` is specified, if the issue that prevented the audit trail data from being written to the audit trail file is later resolved, output to the audit trail file automatically resumes the next time audit trail data is scheduled to be written.

You can use either of the following methods to confirm that output to the audit trail file has resumed:

- Wait until the next time audit trail data is scheduled to be written, and then execute the `adbauditttrail -d` command
- Execute the `adbauditttrail -d` command after using the `adbauditttrail --swap` command to trigger the writing of audit trail data

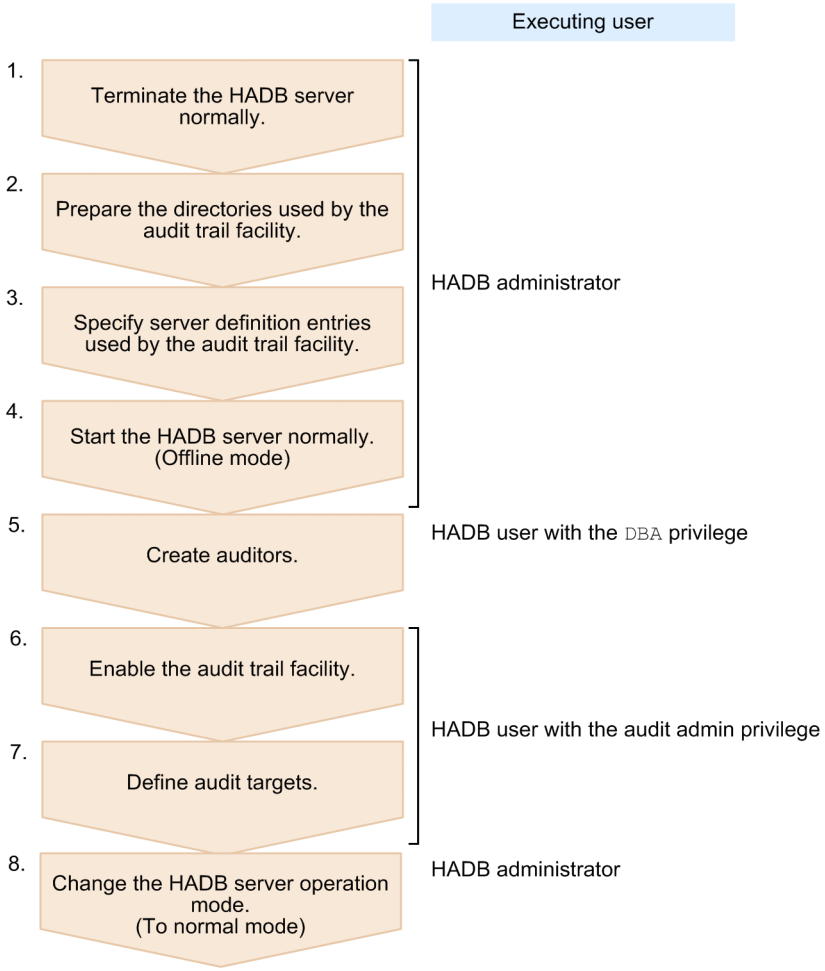
Confirm that `ACTIVE` appears for `audit` in the output of the `adbauditttrail -d` command. If `ACTIVE` appears, output to the audit trail file has resumed.

> **Note**
>
> For information about the `adbauditttrail` command, see *adbauditttrail (Manage the Audit Trail Facility)* in the manual *HADB Command Reference*.

12.2 Setting up the audit trail facility environment

This section describes how to set up the environment for the audit trail facility. Use the procedure shown in the following figure to set up the environment for the audit trail facility.

Figure 12-3: Procedure for setting up audit trail facility environment



Each step in this procedure is described in detail in the following subsections.

12.2.1 Normally terminating the HADB server

The HADB server must have terminated normally before you can set up the environment for the audit trail facility.

To terminate the HADB server normally, the HADB administrator executes the `adbstop` command.

If the HADB server has already terminated, use the following procedure to confirm that it terminated normally.

1. Execute the `adbls -d srv` command.
   Confirm that the HADB server status is `STOP`. This status is output in the output item `STATUS`.
   If it is `STOP`, proceed to step 2 or 3.
2. Check the server message log file.

Open the server message log file (`$ADBDIR/spool/adbmessage`*XX*`.log`[#]), and confirm that the termination mode output in the `KFAA91154-I` message is `normally`.

#

XX is a sequential number between `01` and `04`.

3. Check syslog.
   Open syslog, and confirm that the termination mode output in the `KFAA91154-I` message is `normally`.

If the result in step 1 is `STOP` and the result in step 2 or 3 is `normally`, the HADB server has terminated normally.

If the HADB server did not terminate normally, use the `adbstart` command to start it, and then use the `adbstop` command to terminate it normally.

12.2.2 Preparing the directories used by the audit trail facility

Prepare the directories used by the audit trail facility.

The HADB administrator prepares the following directories used by the audit trail facility based on the results of the assessments and estimates made in 12.1.2 Designing the disks used by the audit trail facility to 12.1.5 Estimating the size of directories used by the audit trail facility:

- Audit trail directory
- Audit trail storage directory
- Audit trail long-term storage directory

The HADB administrator uses the following procedure to prepare the directories used by the audit trail facility. For details about how to perform each step, see the documentation for the operating system you are using.

Procedure:

1. Prepare the disks on which to create the directories used by the audit trail facility.
   Estimate the size of the directories used by the audit trail facility. Make sure that the prepared disks have more than enough space to accommodate directories of the estimated size.
2. Set up the file systems.
   Set up the file systems on the disks prepared in step 1.
3. Create the mount directories.
   Create the directories to which to mount the file systems set up in step 2. Then, mount the file systems.
4. Create the directories used by the audit trail facility.
   Create the directories for storing audit trail data in the file systems set up for each directory.
5. Assign the appropriate permissions in relation to the directories used by the audit trail facility.
   Assign write, read, and execution permissions to the HADB administrator for the directories used by the audit trail facility.

12.2.3 Specifying server definition entries used by the audit trail facility

This subsection describes how to specify the server definition entries used by the audit trail facility.

To use the audit trail facility, the HADB administrator must specify the following operands in the server definition. For details about how to specify operands in the server definition, see 8.5 Creating and modifying a server definition.

■ **Operands related to audit trail facility**

For details about each operand, see 7.2.9 Operands related to audit trail facility (set format).

- `adb_audit_log_path` operand

  Specify the absolute path name of the audit trail directory. You must specify this operand when using the audit trail facility.

- `adb_audit_log_max_size` operand

  Normally, there is no need to specify this operand. Specify this operand if you want to increase the maximum size of audit trail files.

- `adb_audit_log_max_num` operand

  Normally, there is no need to specify this operand. Specify this operand if you want to limit the number of audit trail files that can be created.

12.2.4 Starting the HADB server normally (offline mode)

Start the HADB server in offline mode. The HADB administrator executes the `adbstart` command with the `--offline` option specified.

By starting the HADB server in offline mode, you can prevent unauthorized access while setting up the environment for the audit trail facility.

For details about offline mode, see 10.2.3 HADB server operation modes.

12.2.5 Creating auditors

An HADB user with the `DBA` privilege can create an auditor (an HADB user with the audit privilege). The procedure for creating auditors is as follows:

Example:

The following auditors will be created:

- An HADB user with the audit admin privilege (authorization identifier `ADBAUDITADMIN`)
- An HADB user with the audit viewer privilege (authorization identifier `ADBAUDITOR`)

Procedure:

1. Add the HADB users.

```
CREATE USER "ADBAUDITADMIN" IDENTIFIED BY '#HelloHADB_AUD01'
CREATE USER "ADBAUDITOR" IDENTIFIED BY '#HelloHADB_AUD02'
```

   HADB users are added with the authorization identifiers `ADBAUDITADMIN` and `ADBAUDITOR`.

2. Grant the `CONNECT` privilege and the appropriate audit privilege to the HADB users you created.

```
GRANT CONNECT,AUDIT ADMIN TO "ADBAUDITADMIN"
GRANT CONNECT,AUDIT VIEWER TO "ADBAUDITOR"
```

To the user ADBAUDITADMIN, grant the CONNECT privilege and the audit admin privilege. To the user ADBAUDITOR, grant the CONNECT privilege and the audit viewer privilege.

This completes the process of creating the auditors.

Important

- The audit trail facility administrator and the person responsible for auditing must immediately change the passwords of their accounts from the defaults (#HelloHADB_AUD01 and #HelloHADB_AUD02). For details about how to change passwords, see 11.6.2 Changing an HADB user's password.

  A database administrator who creates an auditor will know the password of that auditor. If the password is not changed from the default, the database administrator could potentially use the account of the auditor to alter the outcome of the audit.
- To enable the audit trail facility, there must be at least one HADB user who has the audit admin privilege.

To make an existing HADB user an auditor, perform only step 2 of the preceding procedure. Note that you cannot grant the audit admin privilege to an HADB user who has the DBA privilege.

12.2.6 Enabling the audit trail facility

The following explains how to enable the audit trail facility.

To enable the audit trail facility, you must execute the adbauditttrail command. As an HADB user with the audit admin privilege, execute the adbauditttrail command with the --start option specified.

When enabling the audit trail facility, you can specify in the --write-error option the approach to take when the HADB server cannot write audit trail data to an audit trail file. Specify a value for the --write-error option based on the approach you selected in 12.1.7 Considering the approach to take when attempts to write to the audit trail file fail.

■ When specifying DOWN in the --write-error option

If you specify DOWN in the --write-error option, the HADB server does not terminate if audit trail data cannot be written to the audit trail file.

A specification example is as follows:

```
adbauditttrail -u ADBAUDITADMIN
               -p '#HelloHADB_ADMIN'
               --start --write-error DOWN
```

■ When specifying FAILSOFT in the --write-error option

If you specify FAILSOFT in the --write-error option, the HADB server does not terminate even if audit trail data cannot be written to the audit trail file. The audit trail data that could not be written to the audit trail file is discarded, and the HADB server continues operating.

A specification example is as follows:

```
adbauditttrail -u ADBAUDITADMIN
               -p '#HelloHADB_ADMIN'
               --start --write-error FAILSOFT
```

Note

For details about the `adbaudittrail` command, see *adbaudittrail (Manage the Audit Trail Facility)* in the manual *HADB Command Reference*.

12.2.7 Defining audit targets

To define audit targets, you execute the `CREATE AUDIT` statement.

Important

- Only an HADB user with audit admin privilege can define audit targets.
- You can only define an audit target when the audit trail facility is enabled.

The following shows an example of defining an audit target:

Example:

This example defines an audit target so that audit trails are also output for optional audit events.

```
CREATE AUDIT AUDITTYPE EVENT FOR ANY OPERATION
```

Note that you cannot define the same audit target definition more than once.

12.2.8 Changing the HADB server operation mode (To normal mode)

After you finish setting up the environment for the audit trail facility, change the operation mode of the HADB server to normal mode.

As the HADB administrator, execute the `adbchgsrvmode` command with the `--normal` option specified. The HADB server operation mode changes from offline mode to normal mode.

Note

For details about the `adbchgsrvmode` command, see *adbchgsrvmode (Change the HADB Server Operation Mode)* in the manual *HADB Command Reference*.

12.3 Scheduled operations for audit trail facility

This section describes the operational items that you must perform regularly in relation to the audit trail facility.

■ **Moving audit trail files**

Use the following methods to move the audit trail files to the appropriate storage directories:

- 12.3.1 Moving audit trail files (to audit trail storage directory)
- 12.3.2 Moving audit trail files (to audit trail long-term storage directory)

■ **Referencing audit trail files**

Use the following methods to check the audit trail data output to audit trail files:

- 12.3.3 Referencing audit trails (when using SELECT statements to reference audit trails)
- 12.3.4 Referencing audit trails (when referencing audit trail data converted to CSV format)

12.3.1 Moving audit trail files (to audit trail storage directory)

This subsection describes how to move audit trail files from the *audit trail directory* to the *audit trail storage directory*.

The HADB administrator must regularly move audit trail files from the audit trail directory to the audit trail storage directory. By moving these files regularly, you can prevent issues like the following:

- The disk containing the audit trail directory becoming full

  If 0 is specified for the `adb_audit_log_max_num` operand in the server definition, there is no limit to the number of audit trail files that can be created. If you do not move the audit trail files, the disk will eventually become full.
- Old audit trail files being deleted

  If 4 or greater is specified for the `adb_audit_log_max_num` operand in the server definition, a limit is imposed on the number of audit trail files that can be created. If the number of audit trail file exceeds the maximum, the oldest audit trail file is deleted. Therefore, older audit trail files will be deleted unless they are moved.

The timing with which to move audit trail files is as follows. Ensure that you move audit trail files before the disk containing the audit trail directory becomes full, and before old audit trail files are deleted.

■ **Timing of moving audit trail files**

- Move audit trail files at predetermined times.

  Move the audit trail files at times determined in advance. For example, you might move audit trail files once a day.
- Move audit trail files when the files are swapped.

  Move audit trail files immediately after the current audit trail file is swapped. Monitor the messages `KFAA81401-I` and `KFAA81402-I` output when the audit trail file is swapped.

> **Important**
>
> If 4 or greater is specified for the `adb_audit_log_max_num` operand in the server definition, older audit trail files might be deleted before the audit trail files can be moved at the predetermined timing. You must therefore monitor these messages and move the audit trail files as soon as the audit trail file is swapped.

Use the following procedure to move audit trail files:

■ Procedure for moving audit trail files

1. Copy the audit trail files from the audit trail directory to the audit trail storage directory.
   Do not copy the current audit trail file.
2. When the copy process has completed, delete the audit trail files at the source.
   Do not delete the following files:
   - The current audit trail file
   - Audit trail files being read by an `ADB_AUDITREAD` function

> **Important**
>
> If you delete the current audit trail file, any audit trail data that was output to the current audit trail file is lost. For this reason, take care not to delete the current audit trail file.

The following is an example of moving audit trail files.

■ Example of moving audit trail files

In this example, audit trail files are moved from the audit trail directory (`/mnt/audittrail/outputarea/audit`) to the audit trail storage directory (`/mnt/audittrail/shorttimesavearea/audit_bak`).

1. Create a list of audit trail files that need to be moved.
   Using the `find` command of the OS, search for audit trail files in the audit trail directory and create a list of audit trail files (`/home/adbmanager/tmp/auditfilelist.txt`) that need to be moved. Note that the following command excludes the current audit trail file from its results.

```
find /mnt/audittrail/outputarea/audit \
    -name "adbaud-????????-??????-???*.aud" \
    > /home/adbmanager/tmp/auditfilelist.txt
```

2. Copy the audit trail files that need to be moved.
   Use the `cp` command of the OS to copy all of the files in the list (`/home/adbmanager/tmp/auditfilelist.txt`) that need to be moved from the audit trail directory to the audit trail storage directory.

```
while read filename ; do
  cp ${filename} /mnt/audittrail/shorttimesavearea/audit_bak
done < /home/adbmanager/tmp/auditfilelist.txt
```

3. Delete the audit trail files for which the copy process has completed.
   When the copy process has completed, use the `rm` command of the OS to delete the source files from the audit trail directory.

```
while read filename ; do
  rm ${filename}
done < /home/adbmanager/tmp/auditfilelist.txt
```

4. Delete the list of audit trail files that need to be moved.
   Using the `rm` command of the OS, delete the list you created in step 1 (`/home/adbmanager/tmp/auditfilelist.txt`) of audit trail files that need to be moved.

```
rm /home/adbmanager/tmp/auditfilelist.txt
```

12.3.2 Moving audit trail files (to audit trail long-term storage directory)

This subsection describes how to move audit trail files from the *audit trail storage directory* to the *audit trail long-term storage directory*.

The HADB administrator moves files that are older than the auditing period from the audit trail storage directory to the audit trail long-term storage directory.

Use the following procedure to move audit trail files:

■ Procedure for moving audit trail files

1. Copy the audit trail files from the audit trail storage directory to the audit trail long-term storage directory.
2. When the copy process has completed, delete the audit trail files at the source.

As needed, you can also use the `gzip` command of the OS to compress the audit trail files in the audit trail long-term storage directory.

The following is an example of moving audit trail files.

■ Example of moving audit trail files

In this example, audit trail files that have been stored for one year are moved from the audit trail storage directory (`/mnt/audittrail/shorttimesavearea/audit_bak`) to the audit trail long-term storage directory (`/mnt/audittrail/longtimesavearea/audit_bak`).

1. Create a list of audit trail files that need to be moved.

   Using the `find` command of the OS, search for audit trail files that have been stored in the audit trail storage directory for a year, and create a list of audit trail files (`/home/adbmanager/tmp/auditfilelist.txt`) that need to be moved.

   ```
   find /mnt/audittrail/shorttimesavearea/audit_bak \
       -name "adbaud-????????-??????-???*.aud" -and -mtime 365 \
       > /home/adbmanager/tmp/auditfilelist.txt
   ```

2. Copy the audit trail files that need to be moved.

   Use the `cp` command of the OS to copy all of the files in the list (`/home/adbmanager/tmp/auditfilelist.txt`) from the audit trail storage directory to the audit trail long-term storage directory.

   ```
   while read filename ; do
     cp ${filename} /mnt/audittrail/longtimesavearea/audit_bak
   done < /home/adbmanager/tmp/auditfilelist.txt
   ```

3. Delete the audit trail files for which the copy process has completed.

   When the copy process has completed, use the `rm` command of the OS to delete the source files from the audit trail storage directory.

   ```
   while read filename ; do
     rm ${filename}
   done < /home/adbmanager/tmp/auditfilelist.txt
   ```

4. Compress the audit trail files that you moved.

   Use the `gzip` command of the OS to compress the audit trail files you moved to the audit trail long-term storage directory.

   ```
   gzip /mnt/audittrail/longtimesavearea/audit_bak/*.aud
   ```

5. Delete the list of audit trail files that need to be moved.

Using the rm command of the OS, delete the list you created in step 1 (/home/adbmanager/tmp/auditfilelist.txt) of audit trail files that need to be moved.

```
rm /home/adbmanager/tmp/auditfilelist.txt
```

12.3.3 Referencing audit trails (when using SELECT statements to reference audit trails)

To reference audit trails, you execute a SELECT statement with the ADB_AUDITREAD function specified.

> **Important**
>
> Only an HADB user with the audit viewer privilege can reference audit trails by executing a SELECT statement that specifies the ADB_AUDITREAD function.

The following are examples of SELECT statements that reference audit trails:

Example 1

This statement references the audit trails in all audit trail files in the audit trail storage directory (/mnt/adittrail/savearea).

Example of SELECT statement execution

```
SELECT * FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/adittrail/savearea/*.aud'])) "DT
"
```

Explanation

In the underlined portion, specify the path of the audit trail files that contain the audit trails you want to reference. In this example, because the name of the audit trail file is specified using the special character * (as *.aud), the statement will reference audit trails stored in all audit trail files (files with the extension aud) in the /mnt/adittrail/savearea directory.

Example 2

In this example, the statement references a list of HADB users who accessed the HADB server between April 1st and April 30th, 2017. The audit trail files to be referenced are stored in the audit trail storage directory (/mnt/adittrail/savearea).

Example of SELECT statement execution

```
SELECT DISTINCT "USER_NAME"
    FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/adittrail/savearea/*.aud'])) "DT"
       WHERE "EXEC_TIME" BETWEEN TIMESTAMP'2017/04/01 00:00:00.000000'
                             AND TIMESTAMP'2017/04/30 23:59:59.999999'
```

Explanation

The USER_NAME column stores the authorization identifier of the HADB user. The EXEC_TIME column contains the date and time at which the HADB user performed the operation on the HADB server (the completion date and time of the event). The USER_NAME and EXEC_TIME columns are the column names in the table function derived table created by the ADB_AUDITREAD function. For details about the column names of table function derived tables, see 12.9.2 Column structure of table function derived table when retrieving audit trails.

For further examples of SELECT statements used when referencing audit trails, see 12.7.5 Performing regular auditing.

For details about the specification rules for the ADB_AUDITREAD function, see *ADB_AUDITREAD function* in the manual *HADB SQL Reference*.

▪ **Notes**

- The audit trail files that can serve as input information for the ADB_AUDITREAD function are the audit trail files output by the HADB server, and these same files after compression by the gzip command of the OS. If you are operating multiple HADB servers, the ADB_AUDITREAD function can also use as input information the audit trail files output by other servers.
- In the ADB_AUDITREAD function, you can specify audit trail files that have been compressed by the gzip command. You cannot specify gzip files that were archived using the tar command.
- Make sure that the appropriate read and execution permissions are assigned and the HADB administrator can access the directories included in the audit trail file path specified in the ADB_AUDITREAD function. For example, if the audit trail files are stored in the /mnt/auditttrail/savearea directory, make sure that the read and execution permissions assigned to the /, /mnt, /mnt/auditttrail, and /mnt/auditttrail/savearea directories all allow access by the HADB administrator. Also assign the appropriate read permission for the audit trail files under the /mnt/auditttrail/savearea directory so that the HADB administrator can read them. If the correct permissions are not assigned, the SELECT statement with the ADB_AUDITREAD function specified will cause an error.
- We recommend that you do not operate the audit trail facility in a way that involves referencing audit trails in the audit trail files in the audit trail directory. If an audit trail file is deleted while an SQL statement is referencing the audit trails it contains, the SQL statement might yield unexpected results. Audit trail files are deleted at the following times:
  - When the audit trail files in the audit trail directory are moved to the audit trail storage directory or audit trail long-term storage directory
  - When both of the following conditions are met and the current audit trail file is swapped:
    - A maximum number of generations of audit trail file has been specified
    - The number of audit trail files in the audit trail directory has reached the maximum

  The second of these scenarios is operationally unavoidable because the HADB server deletes the audit trail files automatically. For this reason, we recommend that you reference the audit trails in audit trail files stored in the audit trail storage directory or the audit trail long-term storage directory.

  If you need to reference audit trails that are in an audit trail file in the audit trail directory, move the audit trail file to the audit trail storage directory before referencing the trails.
- You cannot reference the audit trails in the current audit trail file. If you want to reference the audit trails in this file, swap the current audit trail file and then reference the file you swapped out. For details about how to swap the current audit trail file, see 12.4.2 Swapping the current audit trail file.

12.3.4 Referencing audit trails (when referencing audit trail data converted to CSV format)

You can use spreadsheet software such as Microsoft(R) Excel to view audit trails. To do so, first execute the adbexport command to convert the audit trails to data in CSV format, and then use the resulting CSV data as input information for the spreadsheet software. The procedure for converting audit trails to data in CSV format is as follows:

Example:

This example converts the audit trails in all audit trail files in the audit trail storage directory (/mnt/auditttrail/savearea) to data in CSV format.

Procedure:

1. Create the output data path file for the adbexport command.

   In the output data path file, specify the path name of the output data file (the path name of the file to which audit trails are to be output in CSV format). A specification example of an output data path file is as follows:

   ```
   /home/auditviewer/audit/audit1.csv     ← Audit trail data is output to this file
    in CSV format.
   ```

   Save the output data path file as /home/auditviewer/export/expf01.txt.

2. Create an SQL statement file for the adbexport command.

   In the SQL statement file, specify a SELECT statement that specifies the ADB_AUDITREAD function. A specification example of the SQL statement file is as follows:

   ```
   SELECT * FROM TABLE
       (ADB_AUDITREAD(MULTISET['/mnt/audittrail/savearea/*.aud'])) "T1"
   ```

   Save the SQL statement file as /home/auditviewer/export/sel01.txt.

   Tip

   In this example, because no search conditions are specified in the SELECT statement, the audit trails to be converted to CSV format are all audit trails in the audit trail file targeted by the ADB_AUDITREAD function. If you want to output only the required audit trail information in CSV format, use one of the following methods:

   - Specify search conditions in the SELECT statement in the SQL statement file.
   - In the SQL statement file, specify a SELECT statement that retrieves data from a viewed table.

     In this case, you need to define a viewed table that specifies search conditions in a query specification that specifies, in a table reference, a table function derived table that specifies the ADB_AUDITREAD function.

3. Execute the adbexport command.

   ```
   adbexport -u ADBAUDITOR -p '#HelloHADB_Aud01'
             -q /home/auditviewer/export/sel01.txt
             /home/auditviewer/export/expf01.txt
   ```

   In the -u option and the -p option, specify the authorization identifier and password of an HADB user who has the audit viewer privilege.

   In the -q option, specify the SQL statement file you created in step 2.

   /home/auditviewer/export/expf01.txt is the path name of the output data path file you created in step 1.

Important

The task of converting audit trails to data in CSV format can only be performed by an HADB user who has the audit viewer privilege.

For details about output data path files and the specification rules for SQL statement files, see *Explanation of the specification format and options* in *Specification format for the adbexport command* in the manual *HADB Command Reference*.

Important

- Take care to ensure that the file containing the audit trails converted to CSV data is handled appropriately. You must ensure that only persons who perform auditing can reference the file. You must also make sure that the file cannot be modified.
- We recommend that only persons who perform auditing (HADB users who have the audit viewer privilege) reference audit trails. Therefore, the method in 12.3.3 Referencing audit trails (when using SELECT statements to reference audit trails) is to be used to reference audit trails in most circumstances.

12.4 Non-scheduled operations for the audit trail facility

This section describes the operational items that you must perform from time to time in relation to the audit trail facility.

12.4.1 Adding, deleting, and changing auditors (granting or revoking audit privileges)

This subsection uses examples to describe how to add, delete, and change auditors (how to grant or revoke audit privileges).

(1) Adding auditors (granting audit privileges)

An HADB user who has the DBA privilege can add an auditor. The procedure for adding auditors is as follows:

Example:

In this example, an auditor who has the audit admin privilege (with the authorization identifier ADBAUDITADMIN01) and an auditor who has the audit viewer privilege (with the authorization identifier ADBAUDITOR01) are added. This process will involve creating new HADB users and granting audit privileges to those users.

Procedure:

1. Add the HADB users.

```
CREATE USER "ADBAUDITADMIN01" IDENTIFIED BY '#HelloHADB_AUD01'
CREATE USER "ADBAUDITOR01" IDENTIFIED BY '#HelloHADB_AUD02'
```

   HADB users are added with the authorization identifiers ADBAUDITADMIN01 and ADBAUDITOR01.

2. Grant the CONNECT privilege and the appropriate audit privilege to the HADB users you added.

```
GRANT CONNECT,AUDIT ADMIN TO "ADBAUDITADMIN01"
GRANT CONNECT,AUDIT VIEWER TO "ADBAUDITOR01"
```

   Grant the CONNECT privilege and audit admin privilege to ADBAUDITADMIN01, and the CONNECT privilege and audit viewer privilege to ADBAUDITOR01.

> **Important**
>
> The added auditors must immediately change their passwords from the defaults (#HelloHADB_AUD01 and #HelloHADB_AUD02). For details about how to change passwords, see 11.6.2 Changing an HADB user's password.

You can omit step 1 if you want to make an existing HADB user an auditor. If the existing HADB user already has the CONNECT privilege, you only need to grant the audit privilege in step 2.

Note that you cannot grant the audit admin privilege to an HADB user who has the DBA privilege.

(2) Deleting auditors (revoking audit privileges)

The procedure for deleting auditors is as follows:

Example 1

In this example, an auditor with the audit admin privilege (authorization identifier ADBAUDITADMIN) is deleted.

Procedure:

1. Revoke the audit admin privilege of the auditor (ADBAUDITADMIN).

```
REVOKE AUDIT ADMIN FROM "ADBAUDITADMIN"
```

   This operation is performed by an HADB user who has the audit admin privilege.

2. Delete the HADB user (ADBAUDITADMIN) whose audit admin privilege was revoked.

```
DROP USER "ADBAUDITADMIN"
```

   This operation is performed by an HADB user who has the DBA privilege.

Example 2

In this example, an auditor with the audit viewer privilege (authorization identifier ADBAUDITOR) is deleted.

Procedure:

1. Revoke the audit viewer privilege of the auditor (ADBAUDITOR).

```
REVOKE AUDIT VIEWER FROM "ADBAUDITOR"
```

   This operation is performed by an HADB user who has the audit admin privilege.

2. Delete the HADB user (ADBAUDITOR) whose audit viewer privilege was revoked.

```
DROP USER "ADBAUDITOR"
```

   This operation is performed by an HADB user who has the DBA privilege.

(3) Changing auditors

The procedure for changing auditors is as follows:

Example:

In this example, the auditor with the audit admin privilege is changed. The auditor will be changed from the HADB user with the authorization identifier ADBAUDITADMIN01 to the HADB user with the authorization identifier ADBAUDITADMIN02.

A new HADB user ADBAUDITADMIN02 is created, and this user is then nominated as an auditor.

Procedure:

1. Add the HADB user (ADBAUDITADMIN02).

```
CREATE USER "ADBAUDITADMIN02" IDENTIFIED BY '#HelloHADB_AUD02'
```

   Add an HADB user with the authorization identifier ADBAUDITADMIN02.

   This operation is performed by an HADB user who has the DBA privilege.

2. Grant the CONNECT privilege and audit admin privilege to the HADB user you added (ADBAUDITADMIN02).

```
GRANT CONNECT,AUDIT ADMIN TO "ADBAUDITADMIN02"
```

   This operation is performed by an HADB user who has the DBA privilege.

3. Revoke the audit admin privilege of the auditor (ADBAUDITADMIN01).

```
REVOKE AUDIT ADMIN FROM "ADBAUDITADMIN01"
```

This operation is performed by an HADB user who has the audit admin privilege.

4. Delete the HADB user (ADBAUDITADMIN01) whose audit admin privilege was revoked.

```
DROP USER "ADBAUDITADMIN01"
```

This operation is performed by an HADB user who has the DBA privilege.

Only perform this operation if the HADB user is no longer required.

Important

The new auditor (ADBAUDITADMIN02) must immediately change his or her password from the default (#HelloHADB_AUD02). For details about how to change passwords, see 11.6.2 Changing an HADB user's password.

You can omit step 1 if you want to appoint an existing HADB user as a new auditor. If the existing HADB user already has the CONNECT privilege, you only need to grant the audit admin privilege in step 2.

Note that you cannot grant the audit admin privilege to an HADB user who has the DBA privilege.

(4) Notes

- An HADB user who has the audit admin privilege can revoke their own audit privileges and the audit privileges of other HADB users.
- When the audit trail facility is enabled, if an HADB user is the only user who has the audit admin privilege and the CONNECT privilege, that user cannot revoke his or her own audit admin privilege.
- Audit privileges cannot be revoked while the audit trail facility is disabled. However, if both of the following conditions are met, a user can revoke his or her own audit admin privilege:
  - He or she is the only HADB user who has the audit admin privilege.
  - There are no HADB users who have the audit viewer privilege.

12.4.2 Swapping the current audit trail file

This subsection explains how to swap the current audit trail file.

You cannot reference the audit trail data in the current audit trail file. If you want to reference the audit trail data in the current audit trail file, you must first swap the file. To swap audit trail files, you use the adbauditttrail command.

As an HADB user with the audit admin privilege, execute the adbauditttrail command with the --swap option specified.

Note

For information about the adbauditttrail command, see *adbauditttrail (Manage the Audit Trail Facility)* in the manual *HADB Command Reference*.

When you execute the adbaudittrail command with the --swap option specified, swapping of the audit trail files takes place and the current audit trail file is renamed. You can then use the following methods to reference the audit trails output to the renamed audit trail file:

- 12.3.3 Referencing audit trails (when using SELECT statements to reference audit trails)
- 12.3.4 Referencing audit trails (when referencing audit trail data converted to CSV format)

12.4.3 Checking audit target definitions

To check audit target definitions, you retrieve data from the SQL_AUDITS dictionary table.

Important

Only an HADB user with the audit admin privilege can retrieve data from the SQL_AUDITS dictionary table.

The following shows an example of retrieving data from the SQL_AUDITS dictionary table:

Example of SQL statement execution

```
SELECT * FROM "MASTER"."SQL_AUDITS"
```

Example search results

```
OPERATION_TYPE   EVENT_TYPE  ...  AUDIT_TYPE  IS_EXCLUSION_AUDIT
--------------   ----------      ----------  ------------------
ANY              **********  ...  EVENT                       N
```

A row of search results means that an audit target definition has been defined. In this example, the presence of this definition means that audit trails will also be output for optional audit events.

A lack of search results indicates that there is no audit target definition. In this case, audit trails will not be output for optional audit events.

Note that the search results in this example are those output if the SQL statement was executed by the adbsql command. ********** represents a null value.

12.4.4 Changing audit target definitions

The following describes how to change an audit target definition (that is, how to define an audit target or delete an audit target definition).

Important

- Only an HADB user with audit admin privilege can change an audit target definition.
- You can only change an audit target definition when the audit trail facility is enabled.

■ Defining audit targets

To define an audit target, you execute the CREATE AUDIT statement. The following shows an example of defining an audit target:

Example:

This example defines an audit target so that audit trails are also output for optional audit events.

```
CREATE AUDIT AUDITTYPE EVENT FOR ANY OPERATION
```

Note that you cannot define the same audit target more than once.

■ **Deleting audit target definitions**

To delete an audit target definition, you execute the `DROP AUDIT` statement. The following shows an example of deleting an audit target definition:

Example:

This example deletes the audit target definition so that audit trails are no longer output for optional audit events.

```
DROP AUDIT AUDITTYPE EVENT FOR ANY OPERATION
```

12.4.5 Checking the operational status of the audit trail facility

The following explains how to check the operational status of the audit trail facility.

To check the operational status of the audit trail facility, you must execute the `adbauditttrail` command.

As either of the following HADB users, execute the `adbauditttrail` command with the `-d` option specified:

- An HADB user who has the audit admin privilege
- An HADB user who has the audit viewer privilege

> **Note**
>
> For information about the `adbauditttrail` command, see *adbauditttrail (Manage the Audit Trail Facility)* in the manual *HADB Command Reference*.

When you execute the `adbauditttrail` command with the `-d` option specified, the command outputs the operational status of the audit trail facility.

■ **Checking whether the audit trail facility is enabled**

To find out whether the audit trail facility is enabled, check the value output for `audit`.

If the value output for `audit` is `ACTIVE` or `ACTIVE(NO OUTPUT)`, the audit trail facility is enabled. A value of `ACTIVE(NO OUTPUT)` means that output of audit trails to the audit trail file is suspended.

If the value output for `audit` is `INACTIVE`, the audit trail facility is disabled.

■ **Checking the processing method used when audit trail data cannot be written to the audit trail file**

To find out the approach used when audit trails cannot be written to the audit trail file due to the disk being full or a disk failure, check the value output for `write-error`.

If the value output for `write-error` is `DOWN`, the HADB server is configured to terminate abnormally when an attempt to write to the audit trail file fails.

If the value output for `write-error` is `FAILSOFT`, the HADB server will continue to operate even if an attempt to write to the audit trail file fails.

■ **Checking values specified in server definition entries related to the audit trail facility**

To find out the values specified for server definition entries that relate to the audit trail facility, check the values output for the following items:

- The value output for `audit-directory-path` is the directory specified for the `adb_audit_log_path` operand in the server definition. You can use this value to check the location of the audit trail directory.
- The value output for `audit-file-max-size` is the value specified for the `adb_audit_log_max_size` operand in the server definition. This value represents the maximum size of an audit trail file.
- The value output for `audit-file-number` is the value specified for the `adb_audit_log_max_num` operand in the server definition. This value represents the maximum number of generations of audit trail files.

12.5 Troubleshooting the audit trail facility

This section describes the steps to take if a problem occurs in relation to the audit trail facility.

If an error occurs when outputting audit trails, the KFAA51404-E message is output. The action you need to take depends on the value of the variable *aa....aa* in the KFAA51404-E message.

- If the variable *aa....aa* is `The audit trail file disk is full`
  See 12.5.1 Steps to take when the disk containing the audit trail directory is full.
- If the variable *aa....aa* is `errno` *cc....cc*
  See 12.5.2 Steps to take when a failure occurs in the disk containing the audit trail directory.
- If the variable *aa....aa* is `The audit trail file could not be opened, because file descriptors were insufficient`
  See 12.5.3 Steps to take when there are insufficient file descriptors.

For details about the KFAA51404-E message, see the manual *HADB Messages*.

12.5.1 Steps to take when the disk containing the audit trail directory is full

The following explains the steps to take when the disk that contains the audit trail directory becomes full.

If the disk that contains the audit trail directory becomes full, `The audit trail file disk is full` is displayed for the variable *aa....aa* in the KFAA51404-E message. The steps to take in this situation depends on whether the HADB server is running.

■ If the HADB server terminated abnormally

Create more free space on the disk by using one of the following three methods. After doing so, execute the `adbstart` command to start the HADB server.

■ If the HADB server is running

Create more free space on the disk by using the first or second of the following three methods. You can do so while the HADB server is running.

If you wish to use the third method, you must first use the `adbstop` command to terminate the HADB server. You can then use the third method to create free space on the disk. After doing so, execute the `adbstart` command to start the HADB server.

Action to take

1. Move the audit trail files in the audit trail directory to the audit trail storage directory, with the exception of the current audit trail file.
2. Delete any unnecessary files on the disk containing the audit trail directory.
3. Increase the amount of free disk space by making the capacity of the disk that contains the audit trail directory larger.

12.5.2 Steps to take when a failure occurs in the disk containing the audit trail directory

The following explains the steps to take when a failure occurs in the disk containing the audit trail directory.

When a failure occurs in the disk containing the audit trail directory, `errno` *cc....cc* is displayed for the variable *aa....aa* in the `KFAA51404-E` message. The corrective action to take is as follows.

Action to take

1. Terminate the HADB server.
   If the HADB server is running, use the `adbstop` command to terminate the HADB server.
2. Eliminate the cause of the error.
   Look up the error number displayed in `errno` *cc....cc* in the documentation for the operating system, and eliminate the cause of the error.
   If you are unable to eliminate the cause of the error, create a new audit trail directory. After creating the new audit trail directory, specify the path of the new directory in the `adb_audit_log_path` operand in the server definition.
3. Start the HADB server.
   Use the `adbstart` command to start the HADB server.

12.5.3 Steps to take when there are insufficient file descriptors

The following explains the steps to take when there are insufficient file descriptors.

When there are insufficient file descriptors, `The audit trail file could not be opened, because file descriptors were insufficient` is displayed for the variable *aa....aa* in the `KFAA51404-E` message. The corrective action to take is as follows.

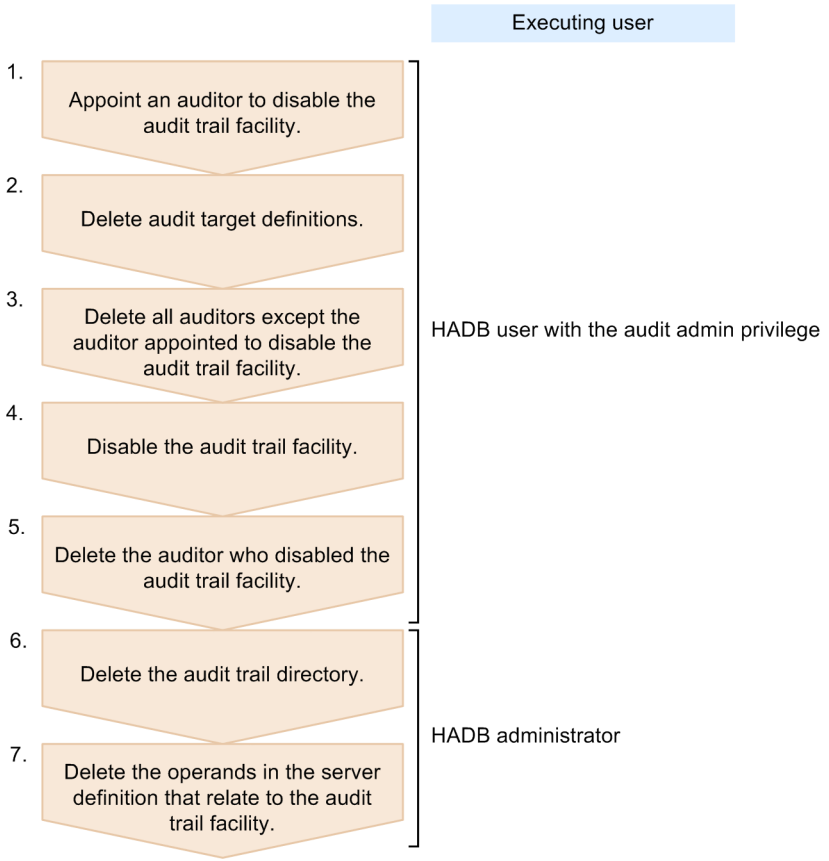
Action to take

1. If there are files associated with processes other than the HADB server process that are able to be closed, first close those files.
   If closing these files does not resolve the problem, continue with the next step.
2. Terminate the HADB server.
   Use the `adbstop` command to terminate the HADB server.
3. Increase the maximum number of file descriptors by changing the value specified for the kernel parameters.
   Increase the value specified for the `nofile` and `file-max` kernel parameters. For details about kernel parameters, see 6.2 Estimating the kernel parameters.
4. Start the HADB server.
   Use the `adbstart` command to start the HADB server.

12.6 Stopping use of the audit trail facility

The following figure shows the procedure for stopping use of the audit trail facility.

Figure 12-4: Procedure for stopping use of audit trail facility



Procedure:

1. Appoint an auditor to disable the audit trail facility.

   Appoint one auditor (an HADB user with the audit admin privilege) whose responsibility it will be to disable the audit trail facility. The auditor you appoint will perform steps 2 to 5.

2. Delete audit target definitions.

   The auditor appointed in step 1 uses a `DROP AUDIT` statement to delete all defined audit target definitions. For details about how to delete audit target definitions, see 12.4.4 Changing audit target definitions.

3. Delete all auditors except the auditor appointed to disable the audit trail facility.

   Before disabling the audit trail facility, you need to delete all auditors other than the auditor who was appointed in step 1. Use the following procedure to delete these auditors:

   - First, the auditor appointed in step 1 identifies the authorization identifiers of all auditors. For details about how to identify the authorization identifier of an auditor, see (36) Checking the authorization identifiers and audit privileges of auditors in B.22 Searching a dictionary table.
   - Next, using these authorization identifiers, the auditor appointed in step 1 uses a `REVOKE` statement to revoke the audit viewer privilege of all auditors, including themselves. There should now be no HADB users who have the audit viewer privilege. For an example of an SQL statement that revokes the audit viewer privilege, see (2) Deleting auditors (revoking audit privileges) in 12.4.1 Adding, deleting, and changing auditors (granting or revoking audit privileges).

- Finally, using the authorization identifiers, the auditor appointed in step 1 uses a `REVOKE` statement to revoke the audit admin privileges of all auditors other than itself. The auditor appointed in step 1 should now be the only HADB user who has the audit admin privilege. For an example of an SQL statement that revokes the audit admin privilege, see (2) Deleting auditors (revoking audit privileges) in 12.4.1 Adding, deleting, and changing auditors (granting or revoking audit privileges).

4. Disable the audit trail facility.

   The auditor appointed in step 1 uses the `adbauditttrail` command to disable the audit trail facility. Executing the `adbauditttrail` command with the `--stop` option specified disables the audit trail facility. Audit trails will no longer be output after the audit trail facility is disabled.

   Command execution example

```
adbauditttrail -u ADBAUDITADMIN
               -p '#HelloHADB_ADMIN'
               --stop
```

   Note

   For information about the `adbauditttrail` command, see *adbauditttrail (Manage the Audit Trail Facility)* in the manual *HADB Command Reference*.

5. Delete the auditor who disabled the audit trail facility.

   The auditor appointed in step 1 uses a `REVOKE` statement to revoke his or her own audit admin privilege. For details about revoking the audit admin privilege, see (2) Deleting auditors (revoking audit privileges) in 12.4.1 Adding, deleting, and changing auditors (granting or revoking audit privileges).

6. Delete the audit trail directory.

   Audit trails will no longer be output after the audit trail facility is disabled. This means that the audit trail directory is no longer required and can be deleted.

   - First, the HADB administrator moves any audit trail files remaining in the audit trail directory to the audit trail storage directory. For details about how to move audit trail files, see 12.3.1 Moving audit trail files (to audit trail storage directory).
   - The HADB administrator then deletes the audit trail directory.

7. Delete the operands in the server definition that relate to the audit trail facility.

   Because you are stopping use of the audit trail facility, the following operands specified in the server definition are no longer required. The HADB administrator can delete these operands as needed.

   - `adb_audit_log_path` operand
   - `adb_audit_log_max_size` operand
   - `adb_audit_log_max_num` operand

   For details about how to modify the server definition, see 8.5.2 Modifying the server definition.

12.7 Audit trail facility operation example

This section provides an operation example that illustrates a real world application of the audit trail facility.

12.7.1 Configuration of information analysis system operated by Company A

The following is an operation example of the audit trail facility, using as an example a fictional Company A that operates an eCommerce website.

Motivation for use of audit trail facility

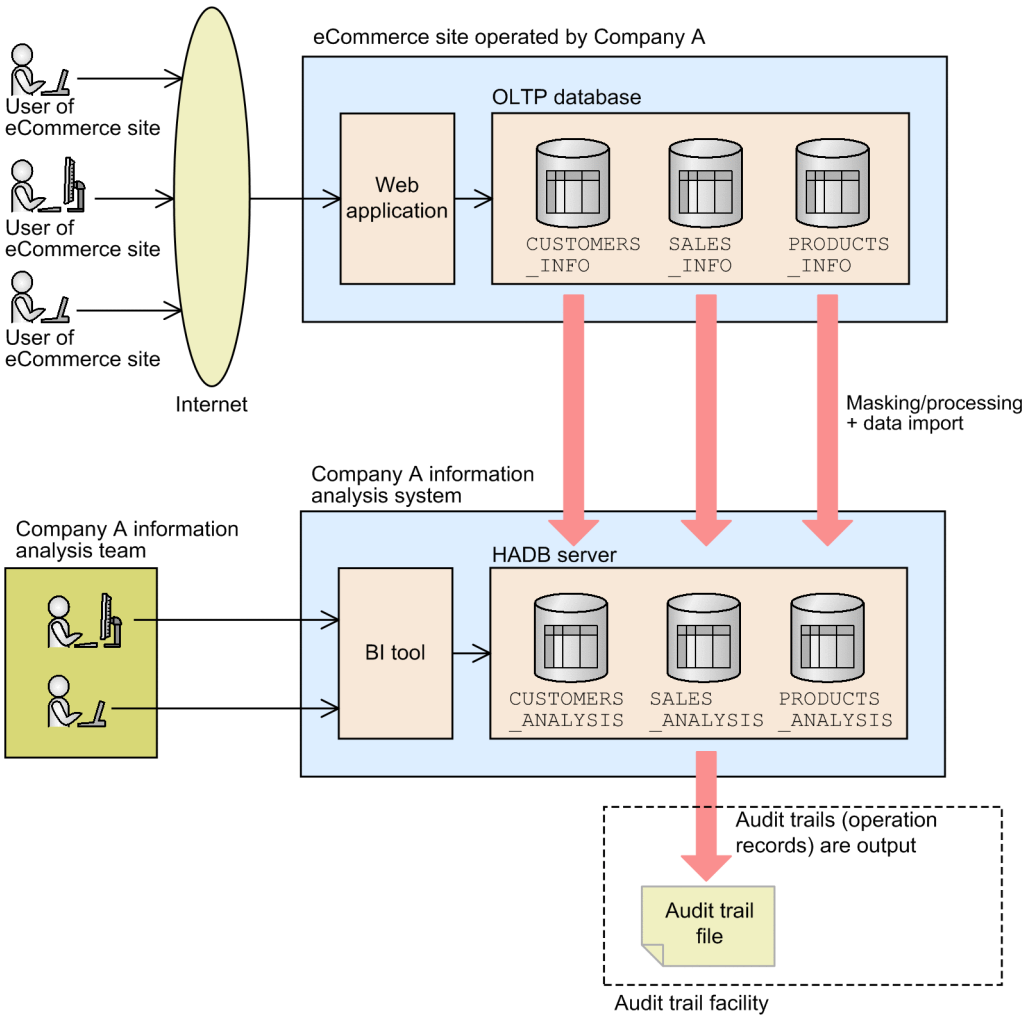
Company A operates an eCommerce site that sells products and services over the Internet. Company A uses an HADB database to manage information such as customer details, sales history, and stock levels. To allow the real-time analysis of trends in product demand and customer preferences, this information (data) is imported into the database every 30 minutes. Because the data imported into the database includes personal information, Company A has chosen to use the audit trail facility provided by HADB so that use of the database can be audited.

Operation of Company A's information analysis system

- Information about customers who log in to the eCommerce website is imported into the database at 30 minute intervals, along with sales and inventory information for the website.
- Company A uses BI tools to analyze the information.
- The HADB server processes approximately 2,000 to 2,400 queries per day.
- The length of any one SQL statement is less than 16 KB.

The following figure shows the configuration of the information analysis system and eCommerce website operated by Company A:

Figure 12-5: Configuration of information analysis system and eCommerce website operated by Company A



An operation example of the audit trail facility is explained in the following subsections, based on the preceding conditions and system configuration.

12.7.2 Design

This subsection explains the design-related tasks that needed to be performed before using the audit trail facility in the manner intended.

(1) Determining the auditing policy

First, the entity using the audit trail facility needs to determine their auditing policy. This covers aspects such as the operations to be audited and the interval at which auditing takes place.

The information analysis system operated by Company A handles personal information. Company A needs to be sure that employees are not accessing personal information inappropriately. Because the database also contains commercially sensitive information such as sales histories and revenue information, they also need to make sure that employees are not accessing this information without the proper authorization.

Because sales information and visitor counts are calculated monthly, Company A has chosen to conduct auditing using the same schedule.

Based on the preceding factors, the operations that are subject to auditing and the interval at which auditing is conducted are determined as follows:

Operations subject to auditing

- Access to the database that stores personal information
- Access to the database that stores commercially sensitive information such as sales histories and revenue information

Frequency of auditing

- Auditing takes place monthly

(2) Determining the events for which to output audit trails

Company A determines the events for which to output audit trails based on the audit policy formulated in (1) Determining the auditing policy.

The auditing process requires that a record is kept of access to the database that stores the data subject to auditing. To allow the auditor to identify the pathway the user took to connect to the HADB server, a record must also be kept of connections to and disconnections from the HADB server.

To output a record of database access, you need to designate *data manipulation SQL events* as subject to audit trail output. To output a record of HADB server connections and disconnections, you need to designate *session events* as subject to audit trail output. If you want the audit trail facility output audit trails when data manipulation SQL events and session events occur, you must use a `CREATE AUDIT` statement to define *optional audit events* as subject to auditing.

> **Note**
>
> For details about event types and the relationship between event types and operations, see 12.9.1 List of audit target events and output items.

(3) Designing the directories used by the audit trail facility

Because personal information is included in the data handled by the information analysis system operated by Company A, the rules for storing audit trail files were determined as follows:

- The past three months of audit trail files are stored in the audit trail directory
- Audit trail files that were output more than three months ago are moved to the audit trail storage directory
- Audit trail files are kept in the audit trail storage directory until one year has elapsed since they were output
- Audit trail files that were output more than a year ago are deleted from the audit trail storage directory

In the information analysis system operated by Company A, data is imported into the database every 30 minutes. The BI tools used by Company A generate approximately 2,400 search queries per day. The directories used by the audit trail facility are designed with these conditions in mind.

■ Designing the audit trail directory

The size of the audit trail directory must be sufficient to store three months of audit trail files. Estimate the size of the audit trail directory as follows:

Formula

```
{2,400 queries × {2 KB + 16 KB (SQL statement length)} + 24 × 2 × (1 KB + 1 KB + 1
 KB)} × 31 days × 3 months
={2,400 × (2 + 16) + 24 × 2 × (1 + 1 + 1)} × 31 × 3 = 4,030,992 KB
≈ 4 GB
```

Based on the result of this formula, after allowing for a certain amount of leeway, a size of 10 GB is selected for the disk area where the audit trail directory is created.

This disk area is mounted to `/mnt/auditttrail/outputarea`, giving the audit trail directory the following path:

- `/mnt/auditttrail/outputarea/audit`

■ **Designing the audit trail storage directory**

The size of the audit trail storage directory must be sufficient to store nine months of audit trail files. Estimate the size of the audit trail storage directory as follows:

Formula

```
{2,400 queries × {2 KB + 16 KB (SQL statement length)} + 24 × 2 × (1 KB + 1 KB + 1
 KB)} × 31 days × 9 months
= {2,400 × (2 + 16) + 24 × 2 × (1 + 1 + 1)} × 31 × 9 = 12,092,976 KB
≈ 12 GB
```

Based on the result of this formula, after allowing for a certain amount of leeway, a size of 20 GB is selected for the disk area where the audit trail storage directory is created.

This disk area is mounted to `/mnt/auditttrail/savearea`, giving the audit trail storage directory the following path:

- `/mnt/auditttrail/savearea/audit_bak`

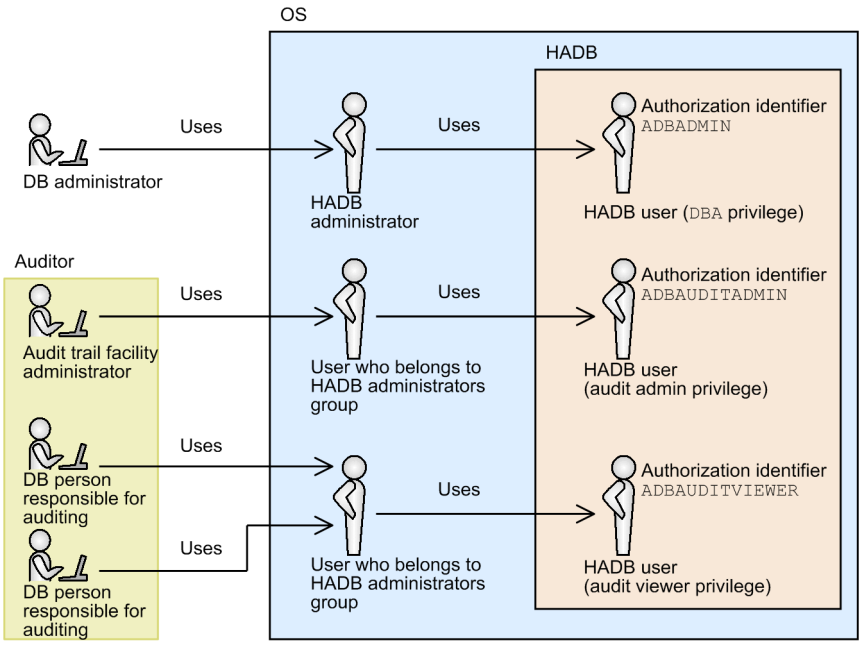
> **Note**
>
> After audit trail files older than three months are moved to the audit trail storage directory, those that are more than a year old are deleted. This means that the audit trail storage directory must have enough space to store nine months of audit trail files.

(4) Appointing auditors

You cannot use the audit trail facility without appointing auditors. The responsibilities of an auditor can include managing and operating the audit trail facility, and conducting the audit process itself.

The database operator cannot be the same person as the auditor who reviews the activity of the database operator. In Company A, the division of roles in relation to the audit trail facility among those responsible for the information analysis system is as follows:

Figure 12-6: Division of roles among those responsible for information analysis system in Company A



- DB administrator

  A person who manages and operates the HADB database. The DB administrator is primarily responsible for database maintenance and other tasks. When logging in to the OS, the DB administrator uses the OS account of the HADB administrator. To connect to the HADB server, the DB administrator uses the authorization identifier `ADBADMIN` of an HADB user who has the `DBA` privilege.

- Audit trail facility administrator

  A person who manages and operates the audit trail facility. The responsibilities of the audit trail facility administrator include enabling and disabling the audit trail facility, and defining audit targets. When logging in to the OS, the audit trail facility administrator uses the account of an OS user who belongs to the HADB administrators group. To connect to the HADB server, the audit trail facility administrator uses the authorization identifier `ADBAUDITADMIN` of an HADB user who has the audit admin privilege.

- DB person responsible for auditing

  A person responsible for the auditing process itself. The DB person responsible for auditing reviews records of database operations and other activity by referencing the audit trail data. When logging in to the OS, the DB person responsible for auditing uses the account of an OS user who belongs to the HADB administrators group. To connect to the HADB server, the DB person responsible for auditing uses the authorization identifier `ADBAUDITVIEWER` of an HADB user who has the audit viewer privilege.

  Two DB persons responsible for auditing are appointed who share use of an OS user account and HADB user authorization identifier.

(5) Selecting an approach for when attempts to write to the audit trail file fail

Company A must select how the system behaves when the HADB server cannot write to the audit trail file, such as when the disk is full or affected by a failure. The following approaches are available:

1. When an attempt to write to the audit trail file fails, the HADB server is stopped (terminated abnormally).
2. When an attempt to write to the audit trail file fails, the data that could not be written is discarded. The HADB server continues to operate.

When the second of these methods is selected, audit trail data might be lost. This can make it impossible to fully investigate the cause of any security incidents that occurred. It might also give the impression that security control measures are not being followed, leading to a loss of reputation for the company and potentially causing considerable losses.

The information analysis system is fully isolated from the eCommerce site that Company A runs 24 hours a day. As such, interruptions to the operation of the information analysis system have no impact on the operation of the eCommerce site.

For these reasons, the first approach was selected in which the HADB server is terminated when an attempt to write to the audit trail file fails.

Company A also decided to monitor the `KFAA51404-E` message output when writing to the audit trail file fails, and have the HADB server administrator and audit trail facility administrator notified by email when a failure to write to the file is detected.

12.7.3 Environment setup

The following explains how to set up the environment for the audit trail facility based on the policies and operation methods determined in 12.7.2 Design.

(1) Creating the directories used by the audit trail facility

Create the following directories for use by the audit trail facility:

- Audit trail directory (`/mnt/auditttrail/outputarea/audit`)
- Audit trail storage directory (`/mnt/auditttrail/savearea/audit_bak`)

For details about the permissions required for each directory, see 12.2.2 Preparing the directories used by the audit trail facility.

> **Important**
>
> The DB administrator creates these directories after logging in to the OS using the account of the HADB administrator.

(2) Adding operands related to the audit trail facility to the server definition

Add the `adb_audit_log_path` operand to the server definition. As the value of the `adb_audit_log_path` operand, specify the audit trail directory created in (1) Creating the directories used by the audit trail facility.

Specification example of operand added to server definition

```
set adb_audit_log_path = /mnt/auditttrail/outputarea/audit
```

Because the default values can be applied, the following operands related to the audit trail facility are not specified:

- `adb_audit_log_max_size`
- `adb_audit_log_max_num`

Important

- The DB administrator performs this operation after logging in to the OS using the account of the HADB administrator.
- If the HADB server is running, terminate the HADB server normally before amending the server definition.

(3) Starting the HADB server in offline mode

Execute the `adbstart` command to start the HADB server in offline mode. By starting the HADB server in offline mode, you can prevent unauthorized access while setting up the environment for the audit trail facility.

Command execution example

```
adbstart --offline
```

Important

The DB administrator performs this operation after logging in to the OS using the account of the HADB administrator.

(4) Creating auditors

The following describes how to create the auditor accounts (HADB users with audit privileges) used by the audit trail facility administrator and DB person responsible for auditing.

Creating auditors

Create a new HADB user (`ADBAUDITADMIN`) to be used by the audit trail facility administrator, and grant the audit admin privilege to this HADB user.

- Example of creating an HADB user with the audit admin privilege

```
CREATE USER "ADBAUDITADMIN" IDENTIFIED BY '@udi+@dmin'
GRANT AUDIT ADMIN,CONNECT TO "ADBAUDITADMIN"
```

Create a new HADB user (`ADBAUDITVIEWER`) to be used by the DB person responsible for auditing, and grant the audit viewer privilege to this HADB user.

- Example of creating an HADB user with the audit viewer privilege

```
CREATE USER "ADBAUDITVIEWER" IDENTIFIED BY '@udi+viewer'
GRANT AUDIT VIEWER,CONNECT TO "ADBAUDITVIEWER"
```

Important

The DB administrator performs this operation after connecting to the HADB server using the authorization identifier (`ADBADMIN`) of an HADB user who has the `DBA` privilege.

Changing the passwords of auditors

After creating the auditor account, the DB administrator notifies the audit trail facility administrator of the authorization identifier and password of the HADB user who was granted the audit admin privilege. The DB administrator also notifies the DB person responsible for auditing of the authorization identifier and password of

the HADB user who was granted the audit viewer privilege. Each auditor must now connect to the HADB server using the authorization identifier of the associated HADB user, and change their password from the default.

- Example of changing password of HADB user with audit admin privilege

```
ALTER USER "ADBAUDITADMIN" IDENTIFIED BY '@uditDDD'
```

  Execute the preceding SQL statement after connecting to the HADB server as the HADB user with the authorization identifier ADBAUDITADMIN.

- Example of changing password of HADB user with audit viewer privilege

```
ALTER USER "ADBAUDITVIEWER" IDENTIFIED BY '@uditVVV'
```

  Execute the preceding SQL statement after connecting to the HADB server as the HADB user with the authorization identifier ADBAUDITVIEWER.

> **Note**
>
> The password of an auditor (an HADB user with audit privileges) can only be changed by that auditor.

(5) Enabling the audit trail facility

Enable the audit trail facility by executing the adbauditttrail --start command.

Command execution example

```
adbauditttrail -u "ADBAUDITADMIN" -p '@uditDDD' --start --write-error DOWN
```

Explanation

- In the -u option and the -p option, specify the authorization identifier and password of an HADB user who has the audit admin privilege.
- The --start option specifies that the audit trail facility is to be enabled.
- In the --write-error option, specify the processing method selected in (5) Selecting an approach for when attempts to write to the audit trail file fail under 12.7.2 Design. In this example, DOWN is specified (the processing method whereby the HADB server is terminated abnormally when an attempt to write to the audit trail file fails).

> **Important**
>
> The audit trail facility administrator performs this operation after logging in to the OS using the account of a user who belongs to the HADB administrators group.

After executing the adbauditttrail --start command, execute the adbauditttrail -d command to confirm that the audit trail facility is enabled.

Command execution example

```
adbauditttrail -u "ADBAUDITADMIN" -p '@uditDDD' -d
```

Command execution result

```
audit  write-error audit-directory-path             audit-file-max-size audit-file-nu
mber
ACTIVE DOWN        /mnt/audittrail/outputarea/audit 256                 0
```

Explanation

- If the audit trail facility is enabled, `ACTIVE` is displayed in the `audit` column.
- The `write-error` column displays the value specified for the `--write-error` option (the behavior when an attempt to write to an audit trail file fails).

(6) Defining audit targets

Based on the design policy established in (2) Determining the events for which to output audit trails under 12.7.2 Design, the audit trail facility is configured to also output audit trails for optional audit events. Define the audit targets by executing the `CREATE AUDIT` statement.

Execution example of `CREATE AUDIT` statement

```
CREATE AUDIT AUDITTYPE EVENT FOR ANY OPERATION
```

> **Important**
>
> The audit trail facility administrator performs this operation after connecting to the HADB server using the authorization identifier (`ADBAUDITADMIN`) of an HADB user who has the audit admin privilege.

(7) Changing the HADB server operation mode

Execute the `adbchgsrvmode` command to change the HADB server operation mode to normal mode. When the HADB server operation mode has changed to normal mode, the HADB server will begin accepting connection requests from clients.

Command execution example

```
adbchgsrvmode --normal
```

> **Important**
>
> The DB administrator performs this operation after logging in to the OS using the account of the HADB administrator.

12.7.4 Scheduled operations

The following operations need to be performed on a regular basis.

- Moving the audit trail files in the audit trail directory to the audit trail storage directory
- Deleting any audit trail files in the audit trail storage directory that are older than the retention period

The following explains each of these operations in detail.

(1) Moving audit trail files

The audit trail files in the audit trail directory need to be regularly moved to the audit trail storage directory. Based on the rules governing audit trail file retention determined in (3) Designing the directories used by the audit trail facility under 12.7.2 Design, audit trail files are moved to the audit trail storage directory when they are more than three months old.

Command execution example

```
find /mnt/audittrail/outputarea/audit -name "adbaud-*.aud" -and -mtime 93 -exec mv {}
 /mnt/audittrail/savearea/audit_bak \;
```

Important

The DB administrator performs this operation after logging in to the OS using the account of the HADB administrator.

(2) Deleting audit trail files

Audit trail files in the audit trail storage directory need to be deleted regularly. Based on the rules governing audit trail file retention determined in (3) Designing the directories used by the audit trail facility under 12.7.2 Design, audit trail files are deleted when they are more than a year old.

Command execution example

```
find /mnt/audittrail/savearea/audit_bak -name "adbaud-*.aud" -and -mtime 365 -exec rm
 -f {} \;
```

Important

The DB administrator performs this operation after logging in to the OS using the account of the HADB administrator.

You can automate the process of moving and deleting files by registering the following command in cron:

```
crontab -e
# The following are crontab settings.
# Every day at 00:05, audit trail files older than three months are moved
5 0 * * * /bin/find /mnt/audittrail/outputarea/audit -name "adbaud-*.aud" -and -mtime
 93 -exec mv {} /mnt/audittrail/savearea/audit_bak \;
# Every day at 00:05, audit trail files older than one year are deleted
5 0 * * * /bin/find /mnt/audittrail/savearea/audit_bak -name "adbaud-*.aud" -and -mti
me 365 -exec rm -f {} \;
```

12.7.5 Performing regular auditing

Based on the auditing policy determined in (1) Determining the auditing policy under 12.7.2 Design, the auditors of Company A check the following as part of regular auditing:

- Whether any suspicious connections have been made to the HADB server

Most connections to the HADB server in Company A come from BI tools. If the DB administrator has reason to connect to the HADB server, such as database maintenance, he or she submits an application to that effect in advance. Therefore, this check identifies whether any connections have been made to the HADB server by other means. The auditors also check whether connections to the HADB server are made in a way that matches the submitted applications.

- Whether any unauthorized database operations have been performed

  Database operations are performed using SQL statements generated automatically by the BI tools. This means that the SQL statements executed on the HADB server are generally typical ones, and tend to conform to a limited number of patterns. The auditors check whether any complex SQL statements have been executed that do not fit the usual patterns. The presence of SQL statements of this nature might indicate that unauthorized operations have been performed.

- Whether the configuration of the audit trail facility has been modified without authorization

  Changes to the configuration of the audit trail facility are subject to a predetermined internal application process, and are implemented only after permission is granted. The auditors must therefore check whether any changes have been made to the configuration without permission.

The following subsections provide examples of how to check these audit items.

(1) Checking for suspicious connections to the HADB server

In the information analysis system operated by Company A, the authorization identifier and source IP address for connections from BI tools to the HADB server are fixed as follows:

- Authorization identifier: `BI_ANALYSIS`
- Source IP address: `11.168.xx.32`

If a different authorization identifier or IP address is used for a connection to the HADB server, this might indicate that that connection lacks the proper authorization.

It is also Company A policy for the DB administrator to submit an application before performing database maintenance or other work. This application includes information like the reason for maintenance, the nature of the work, the machine the DB administrator will use, when the work will begin, and the authorization identifier of the HADB user. If the DB administrator performs any work that was not mentioned in the application, he or she follows up with the application recipient by reporting the additional work and the reason for it. Because access to the HADB server is governed by these rules, when a connection is made to the HADB server that involves operations that are not mentioned in an application or later report, that connection potentially involves unauthorized activity.

Accordingly, the following are checked as part of regular auditing:

- Whether any HADB users have made failed attempts to connect to the HADB server
- Whether any connections have been made to the HADB server whose attributes (authorization identifier, IP address, and working time period) do not appear in an application

The following explains how to look for each of these irregularities.

■ Investigating whether any HADB users have made failed attempts to connect to the HADB server

Connections from BI tools to the HADB server are unlikely to fail. This means that the auditor can focus on whether any HADB users have made failed attempts to connect to the HADB server. An auditor can check for HADB users who have made failed attempts to connect to the HADB server by executing the following SQL statement:

Example of SQL statement execution

```
SELECT "USER_NAME","CLIENT_IP_ADDRESS","EXEC_TIME"
   ...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/audittrail/outputarea/audit/*.aud'])) "D
T" ...2
    WHERE "EVENT_SUBTYPE"='CONNECT' AND "EVENT_RESULT"='FAILURE'
   ...3
```

Explanation

1. This SQL statement outputs a list of failed connections to the HADB server. This list contains the authorization identifiers (USER_NAME) of HADB users who made failed attempts, the IP address of the connection source machine (CLIENT_IP_ADDRESS), and the date and time of the connection attempt (EXEC_TIME).
2. Because regular auditing targets the past three months of audit trail data, the input information is all audit trail files under the audit trail directory (/mnt/audittrail/outputarea/audit).
3. The SQL statement specifies conditions that extract records of operations that involve failed connections to the HADB server.

The auditor checks the results of executing the preceding SQL statement. Specifically, the auditor makes sure that the authorization identifiers (USER_NAME), IP addresses of the connection source machines (CLIENT_IP_ADDRESS), and the dates and times of connection attempts (EXEC_TIME) match those in submitted applications or later reports. If the auditor identifies an operation that is not mentioned in an application or later report, due to potentially signifying unauthorized activity, it is reported as an audit observation.

■ Investigating whether any connections have been made to the HADB server whose attributes (authorization identifier, IP address, and working time period) do not appear in an application

The auditor investigates whether any connections have been made to the HADB server that do not involve work submitted in an application. The auditor can check for connections to the HADB server that do not relate to such work by executing the following SQL statement:

Example of SQL statement execution

```
SELECT "USER_NAME","CLIENT_IP_ADDRESS","EXEC_TIME"
   ...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/audittrail/outputarea/audit/*.aud'])) "D
T" ...2
    WHERE "EVENT_SUBTYPE"='CONNECT'
   ...3
     AND ("CLIENT_IP_ADDRESS"!='10.196.xx.122'
   ...4
       OR "USER_NAME"!='MAINTENANCE_USER25239'
   ...4
       OR "EXEC_TIME" NOT BETWEEN TIMESTAMP'2017/06/08 00:00:00.000000'
   ...4
                              AND TIMESTAMP'2017/06/08 00:30:00.000000')
   ...4
```

Explanation

1. This SQL statement outputs a list that contains the authorization identifiers (USER_NAME) of HADB users who attempted to connect to the HADB server, the IP address of the connection source machine (CLIENT_IP_ADDRESS), and the date and time of the connection (EXEC_TIME).
2. Because regular auditing targets the past three months of audit trail data, the input information is all audit trail files under the audit trail directory (/mnt/audittrail/outputarea/audit).
3. The SQL statement specifies conditions that extract records of connections to the HADB server.
4. The SQL statement specifies conditions that do not match the contents of applications. The IP address, authorization identifier, and work time in the application are specified as negated conditions.

The preceding SQL statement is for an example where an application has only been submitted in relation to one task. If applications were submitted for multiple tasks, add the conditions of those applications as search conditions.

The execution results of the preceding SQL statement contain information such as the authorization identifiers of HADB users who attempted to connect to the HADB server outside the parameters of any submitted application. These connections to the HADB server, due to potentially signifying unauthorized activity, are reported as audit observations.

(2) Checking whether any unauthorized database operations have been performed

In the information analysis system operated by Company A, the following database operations take place:

- Importing data into tables (every 30 minutes)
- Using BI tools to retrieve data from the database
- Performing database operations in relation to work for which an application has been submitted

Because any database operations other than the preceding potentially involve unauthorized activity, the following aspects are checked during regular auditing:

- Whether data has been imported in a way that appears suspicious
- Whether data has been retrieved from the database by a method other than a BI tool
- Whether database operations have been performed for which there is no corresponding application

The following explains how to look for each of these irregularities.

■ Investigating whether data has been imported in a way that appears suspicious

By executing the following SQL statement, the auditor can review records of operations that imported data into tables.

Example of SQL statement execution

```
SELECT "USER_NAME","EXEC_TIME"
   ...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/audittrail/outputarea/audit/*.aud'])) "D
T" ...2
    WHERE "EVENT_SUBTYPE"='ADBIMPORT'
   ...3
```

Explanation

1. This SQL statement outputs a list that contains the authorization identifiers (`USER_NAME`) of HADB users who imported data, and the date and time (`EXEC_TIME`) when the data was imported.
2. Because regular auditing targets the past three months of audit trail data, the input information is all audit trail files under the audit trail directory (`/mnt/audittrail/outputarea/audit`).
3. The SQL statement specifies conditions that extract records of operations that involve executing the `adbimport` command.

In the result of the preceding SQL statement, the auditor checks the `EXEC_TIME` column (the date and time when data was imported). At Company A, data is imported on the hour and half past every hour. If data is imported at a time that clearly does not match this pattern, it is possible that this activity is not part of the regular import of data. Because data import taking place without a corresponding application being submitted might also signify unauthorized activity, this is also reported as an audit observation.

■ **Investigating whether data has been retrieved from the database other than by a BI tool**

By executing the following SQL statement, the auditor can review records of data retrieval from the database that exclude those associated with BI tools.

Example of SQL statement execution

```
SELECT "USER_NAME","EXEC_TIME"
   ...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/audittrail/outputarea/audit/*.aud'])) "D
T" ...2
    WHERE "EVENT_SUBTYPE"='SELECT'
   ...3
     AND ("CLIENT_IP_ADDRESS"!='11.168.xx.32' OR "USER_NAME"!='BI_ANALYSIS')
   ...3
```

Explanation

1. This SQL statement outputs a list that contains of the authorization identifiers (`USER_NAME`) of HADB users who retrieved data (excluding retrieval using BI tools), and the date and time (`EXEC_TIME`) when the data was retrieved.
2. Because regular auditing targets the past three months of audit trail data, the input information is all audit trail files under the audit trail directory (`/mnt/audittrail/outputarea/audit`).
3. The SQL statement specifies conditions that extract records of data retrieval from the database (excluding retrieval using BI tools).

When a BI tool searches the database, it always uses the authorization identifier `BI_ANALYSIS` when connecting to the HADB server. The IP address of the client that connects to the HADB server is always `11.168.xx.32`. Accordingly, a connection to the HADB server using a different authorization identifier or IP address means that data was retrieved without using a BI tool. Such data retrieval is reported as an audit observation due to potentially signifying unauthorized activity.

The auditor also checks whether the data retrieval operation is associated with an application submitted in advance.

■ **Investigating whether database operations have been performed without a corresponding application**

One approach an auditor might take to identify this kind of activity is to output a list of database operations and compare it to the submitted applications.

(3) Checking whether the audit trail facility configuration has been modified without authorization

Changes to the configuration of the audit trail facility are subject to an internal application process. The requested changes are only made after permission is granted at the end of this process. Accordingly, changes to the configuration of the audit trail facility made without undergoing this application process might signify unauthorized activity.

Even when a user has gone through the application process, if changes are made that differ from those in the application, this could also signify unauthorized activity. This potentially unauthorized activity is reported as audit observations.

By executing the following SQL statement, the auditor can review records of operations that change the configuration of the audit trail facility.

Example of SQL statement execution

```
SELECT "USER_NAME","CLIENT_IP_ADDRESS","EXEC_TIME"
...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/audittrail/outputarea/audit/*.aud'])) "DT"
...2
```

```
    WHERE "EVENT_TYPE"='AUDIT'
...3
```

Explanation

1. This SQL statement outputs a list that contains the authorization identifiers (USER_NAME) of HADB users who changed the configuration of the audit trail facility, the IP address of the machine from which the user requested the configuration change (CLIENT_IP_ADDRESS), and the date and time of the configuration change (EXEC_TIME).
2. Because regular auditing targets the past three months of audit trail data, the input information is all audit trail files under the audit trail directory (/mnt/audittrail/outputarea/audit).
3. The SQL statement specifies conditions that extract records of operations that change the configuration of the audit trail facility.

The auditor checks the results of executing the preceding SQL statement. If configuration changes have been made to the audit trail facility without a corresponding application, due to potentially signifying unauthorized activity, it is reported as an audit observation.

12.7.6 Investigating security incidents

This subsection provides an example of the investigation that might take place when the following security incident occurs:

Circumstances of security incident

Company A is notified that the personal information of users of the eCommerce site operated by Company A might have leaked onto the Internet. It quickly becomes clear that the information is customer information managed by Company A's information analysis system. Company A begins an investigation led by the DB administrator and audit trail facility administrator, who investigate the channels through which the leak occurred, when it occurred, and the extent of the leak. An example of this investigation is explained in the following subsections.

(1) Identifying the source of the leak

The investigation shows that the information leaked onto the Internet includes information such as the gender and date of birth of customers. Because this information is stored in the CUSTOMERS_ANALYSIS table of the database of the information analysis system, the auditors were able to identify this table as the source of the leaked data.

(2) Investigating which HADB users accessed the data which was leaked

The auditors investigate which HADB users accessed the CUSTOMERS_ANALYSIS table that stored the data that was leaked, the IP address from which the table was accessed, the date and time it was accessed, and the number of accessed rows.

Executing the following SQL statement shows the CUSTOMERS_ANALYSIS table access information, such as HADB users who accessed the table:

Example of SQL statement execution

```
SELECT "USER_NAME","CLIENT_IP_ADDRESS","EXEC_TIME","ACCESS_COUNT"
 ...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/audittrail/savearea/audit_bak/*.aud',
 ...2
                                   '/mnt/audittrail/outputarea/audit/*.aud'])) "DT"
 ...2
```

```
   WHERE "OBJECT_SCHEMA_NAME"='ANALYSIS_SYSTEM'
...3
     AND "OBJECT_NAME"='CUSTOMERS_ANALYSIS'
...3
     AND ("EVENT_SUBTYPE"='SELECT' OR "EVENT_SUBTYPE"='ADBEXPORT')
...4
     AND "EVENT_RESULT"='SUCCESS'
...5
```

Explanation

1. This SQL statement outputs the access information of the `CUSTOMERS_ANALYSIS` table in a list format. This list consists of the following types of information: Authorization identifiers of the HADB users who accessed the table (`USER_NAME`), the IP address of the machine that requested access to the table (`CLIENT_IP_ADDRESS`), the access date and time (`EXEC_TIME`), and the number of accessed rows (`ACCESS_COUNT`).
2. Because it is still unknown when the unauthorized access took place, the scope of the investigation includes all audit trail data. Therefore, all audit trail files in the following directories are used as input information:
   - Audit trail storage directory (`/mnt/auditttrail/savearea/audit_bak`)
   - Audit trail directory (`/mnt/auditttrail/outputarea/audit`)
3. The SQL statement specifies the schema name and table identifier of the leaked `CUSTOMERS_ANALYSIS` table as retrieval conditions.
4. The SQL statement specifies the operations in question (data retrieval using a `SELECT` statement and exporting table data using an `adbexport` command) as retrieval conditions.
5. The SQL statement specifies the success of the operation (event) in step 4 as a retrieval condition.

Important

- The DB person responsible for auditing performs this operation after connecting to the HADB server using the authorization identifier (`ADBAUDITVIEWER`) of an HADB user who has the audit viewer privilege.
- The auditor can include the audit trail data in the current audit trail file in the investigation by using the `adbauditttrail --swap` command to swap the current audit trail file before executing the SQL statement.

(3) Investigating whether an HADB user responsible for unauthorized access has accessed other tables

After identifying the HADB user who accessed data without authorization in (2) Investigating which HADB users accessed the data which was leaked, the next step is to investigate whether there is any evidence of that HADB user accessing other tables. If the HADB user has accessed other tables, the information stored in those tables might also have leaked.

Executing the following SQL statement shows which tables have been accessed by the HADB user.

Example of SQL statement execution

This example assumes the authorization identifier of the HADB user responsible for unauthorized access is `ADBUSER05`.

```
SELECT "EXEC_TIME","OBJECT_OWNER_NAME","OBJECT_SCHEMA_NAME",
 ...1
                "OBJECT_NAME","ACCESS_COUNT"
 ...1
  FROM TABLE(ADB_AUDITREAD(MULTISET['/mnt/audittrail/savearea/audit_bak/*.aud',
 ...2
                                    '/mnt/audittrail/outputarea/audit/*.aud'])) "DT"
 ...2
    WHERE "USER_NAME"='ADBUSER05'
 ...3
     AND ("OBJECT_TYPE"='TABLE' OR "OBJECT_TYPE"='VIEW')
 ...4
     AND ("EVENT_SUBTYPE"='SELECT' OR "EVENT_SUBTYPE"='ADBEXPORT')
 ...5
     AND "EVENT_RESULT"='SUCCESS'
 ...6
     AND "OBJECT_NAME"!='CUSTOMERS_ANALYSIS'
 ...7
```

Explanation

1. This SQL statement outputs a list of the following items:
   - The date and time (`EXEC_TIME`) at which the HADB user responsible for unauthorized access accessed tables (including viewed tables)
   - The following information about the tables accessed by the HADB user responsible for unauthorized access:
     - The authorization identifier of the table owner (`OBJECT_OWNER_NAME`)
     - The schema name of the table (`OBJECT_SCHEMA_NAME`)
     - The table identifier (`OBJECT_NAME`)
   - The number of accessed rows (`ACCESS_COUNT`)
2. Because it is still unknown when the unauthorized access took place, the scope of the investigation includes all audit trail data. Therefore, all audit trail files in the following directories are used as input information:
   - Audit trail storage directory (`/mnt/audittrail/savearea/audit_bak`)
   - Audit trail directory (`/mnt/audittrail/outputarea/audit`)
3. The SQL statement specifies the authorization identifier of the HADB user responsible for unauthorized access (`ADBUSER05`) as a retrieval condition.
4. The SQL statement specifies the object types (table and viewed table) as retrieval conditions.
5. The SQL statement specifies the operations in question (data retrieval using a `SELECT` statement and exporting table data using an `adbexport` command) as retrieval conditions.
6. The SQL statement specifies the success of the operation (event) in step 5 as a retrieval condition.
7. The SQL statement specifies a retrieval condition that excludes the `CUSTOMERS_ANALYSIS` table from the result. This condition is specified because the auditor already knows that the `CUSTOMERS_ANALYSIS` table was accessed without authorization.

Important

- The DB person responsible for auditing performs this operation after connecting to the HADB server using the authorization identifier (`ADBAUDITVIEWER`) of an HADB user who has the audit viewer privilege.

- The auditor can include the audit trail data in the current audit trail file in the investigation by using the `adbaudittrail --swap` command to swap the current audit trail file before executing the SQL statement.

(4) Reporting findings

The investigators at Company A summarize and report their findings as follows:

- HADB user who leaked the information

  The HADB user identified in (2) Investigating which HADB users accessed the data which was leaked might be the HADB user who leaked the information.

- Date and time when leak occurred

  The date and time identified in (2) Investigating which HADB users accessed the data which was leaked and (3) Investigating whether an HADB user responsible for unauthorized access has accessed other tables might be the date and time when the information was leaked.

- Leaked information

  The information in the tables identified in (1) Identifying the source of the leak and (3) Investigating whether an HADB user responsible for unauthorized access has accessed other tables might have been leaked.

- Number of items of leaked information (extent of harm)

  The access counts identified in (2) Investigating which HADB users accessed the data which was leaked and (3) Investigating whether an HADB user responsible for unauthorized access has accessed other tables might reflect the number of pieces of information that were leaked.

12.8 Linkage between the audit trail facility and JP1/Audit Management - Manager

This section describes how to link between the audit trail facility and JP1/Audit Management - Manager (JP1/Audit).

The description of linkage between the audit trail facility and JP1/Audit here assumes that the reader has knowledge of JP1/Audit and JP1/Base. For details about JP1/Base, see the *JP1/Base User's Guide*.

For an overview of conversion of audit trails, see 2.18.9 Conversion of audit trail information (linkage with JP1/Audit Management - Manager).

> **Important**
>
> In the JP1/Audit manuals, audit trails are called *audit logs*. In the HADB manuals, therefore, audit trails are sometimes called audit logs.

12.8.1 Overview of linkage between the audit trail facility and JP1/Audit

This subsection provides information about the following three points as an overview of linkage between the audit trail facility and JP1/Audit:

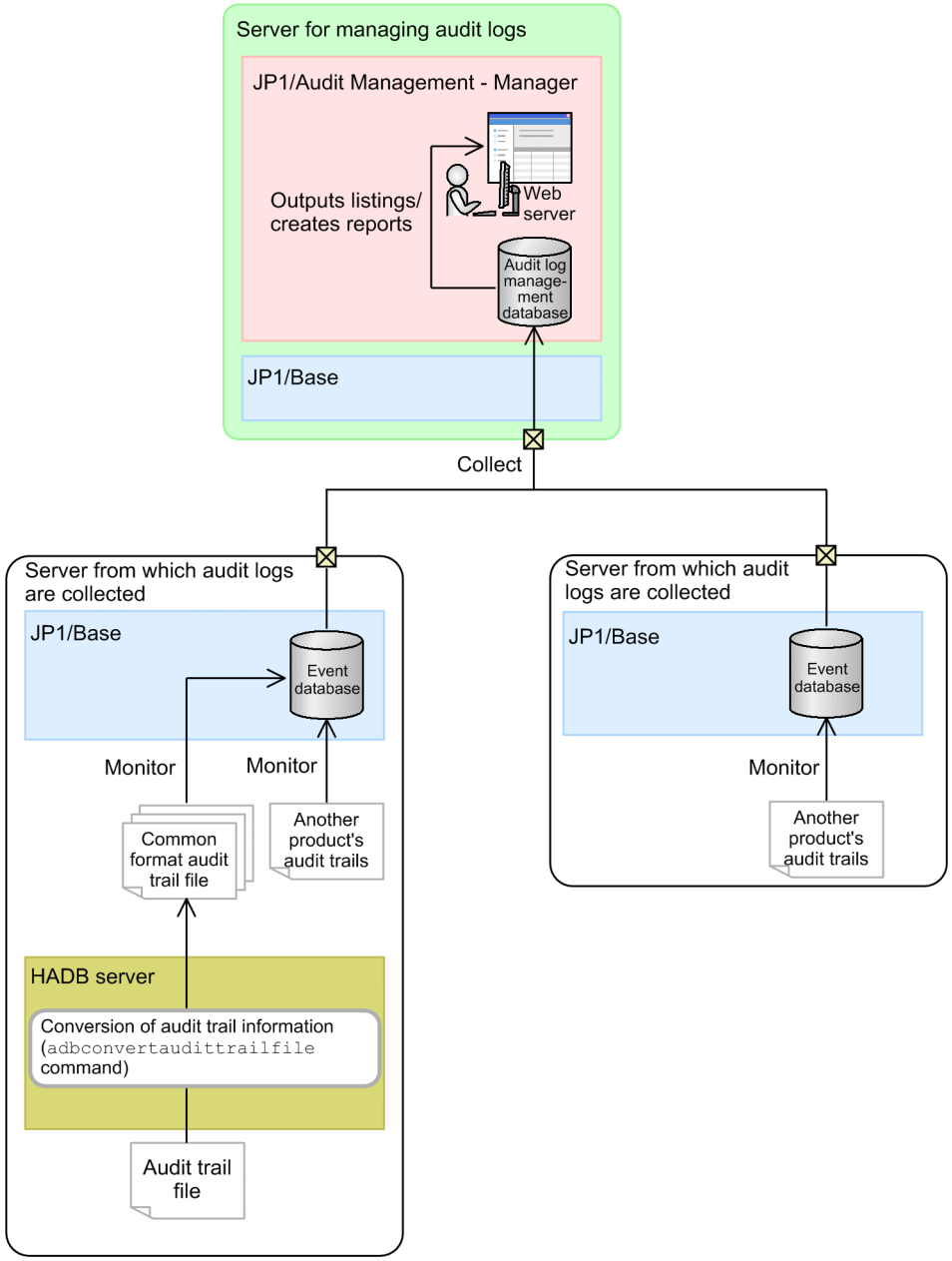
- System configuration example
- Server configuration and prerequisite software programs
- Notes on executing the `adbconvertaudittrailfile` command

(1) System configuration example

This subsection explains a system configuration for linking the audit trail facility with JP1/Audit.

The following figure shows an example of a system configuration for linking the audit trail facility with JP1/Audit.

Figure 12-7: Example of a system configuration for linking the audit trail facility with JP1/Audit



Explanation

The system configuration example in this figure consists of two types of servers (*a server for managing audit logs* and *servers from which audit logs are collected*).

The server for managing audit logs collects audit trail information from the event databases of JP1/Base on the servers from which audit logs are collected. The server for managing audit logs then can centrally manage the collected information.

On one server from which audit logs are collected, the HADB server outputs audit trail information and converts the information by using the `adbconvertaudittrailfile` command. The converted audit trail information is collected into the event database of JP1/Base on the server from which audit logs are collected.

Note

The user who is appointed as an auditor can perform auditing by accessing the server for managing audit logs from a web browser.

(2) Server configuration and prerequisite software programs

This subsection explains a server configuration and prerequisite software programs for linking the audit trail facility with JP1/Audit.

■ Server configuration

As explained in (1) System configuration example, the following two types of servers are required to link the audit trail facility with JP1/Audit:

- Server for managing audit logs

  A server machine on which JP1/Audit and JP1/Base operate. This server collects audit trail information from servers from which audit logs are collected, and centrally manages the collected information.

- Server from which audit logs are collected

  A server machine on which the HADB server and JP1/Base operate. This server collects the HADB server's audit trail information converted by the `adbconvertaudittrailfile` command into the event database of JP1/Base.

> **Important**
>
> The HADB server and JP1/Audit support different OSs. Therefore, it is impossible to install JP1/Audit on a server machine on which the HADB server operates.

■ Prerequisite software programs

The following table lists the software programs that are prerequisite for linkage between the audit trail facility and JP1/Audit.

Table 12-4: Software programs prerequisite for linkage between the audit trail facility and JP1/Audit

No.	Installation-target server	Prerequisite software program
1	Server for managing audit logs	JP1/Audit Management - Manager 11-00 or later
2		JP1/Base 11-10 or later
3	Server from which audit logs are collected	JP1/Base 11-10 or later

(3) Notes on executing the adbconvertaudittrailfile command

Note the following points when executing the `adbconvertaudittrailfile` commands:

- Execute the `adbconvertaudittrailfile` command while audit logs are being monitored by JP1/Audit. If the `adbconvertaudittrailfile` command is executed while the audit logs are not being monitored, JP1/Audit does not collect the converted audit trail information.
- The `adbconvertaudittrailfile` command can be executed only while the HADB server is operating. To prevent other users from using the HADB server while the `adbconvertaudittrailfile` command is being executed, change the HADB server operation mode to maintenance mode. After that, execute the `adbconvertaudittrailfile` command. For details about maintenance mode, see 10.2.3 HADB server operation modes.
- The `adbconvertaudittrailfile` command cannot be executed for the *current audit trail file*. To convert the audit trail information in a file that is the current audit trail file, use the `adbaudittrail --swap` command to swap the current audit trail file. Then, execute the `adbconvertaudittrailfile` command for the renamed audit trail file. For details about swapping the current audit trail file, see 12.4.2 Swapping the current audit trail file.

Note

- For details about the `adbconvertaudittrailfile` command, see *adbconvertaudittrailfile (Convert the Audit Trail File)* in the manual *HADB Command Reference*.
- For details about the `adbaudittrail` command, see *adbaudittrail (Manage the Audit Trail Facility)* in the manual *HADB Command Reference*.

12.8.2 Format and output items of common format audit trail files

This subsection explains the format of common format audit trail files. Audit trail information converted by the `adbconvertaudittrailfile` command is output to these files.

When the `adbconvertaudittrailfile` command is executed, audit trail information is output to a common format audit trail file in the format in which a single audit trail is output per line. In a common format audit trail file, a blank line is output before the first audit trail is output.

The following shows the format and output items of audit trail information that is converted by the `adbconvertaudittrailfile` command and is output to a common format audit trail file.

■ Output format of audit trail information converted by the `adbconvertaudittrailfile` command

```
CALFHM 1.0,item1=value1,item2=value2,item3=value3,...itemn=valuen
```

■ Output items of audit trail information converted by the `adbconvertaudittrailfile` command

The following table lists the items that are output to audit trail information converted by the `adbconvertaudittrailfile` command.

Note that the `ADB_AUDITREAD` function can also convert the audit trail information in an audit trail file. Most of the items output to the conversion results of the `adbconvertaudittrailfile` command are the same as the items output when the function converts the information to a dataset in tabular format (as a table function derived table). For details about the column configuration of a table function derived table generated by the `ADB_AUDITREAD` function, see 12.9.2 Column structure of table function derived table when retrieving audit trails.

If the data of an output item is longer than the maximum length of that output item, the heading or trailing bytes are omitted. The omitted part of data is indicated as an ellipsis (...). For the output items in the following table, unless otherwise specified, the trailing bytes are omitted if data is too long.

If the data to be output to an output item is `NULL`, the output item itself is omitted.

Table 12-5: List of items output to the audit trail information converted by the adbconvertaudittrailfile command

No.	Output item			Information that is output	Corresponding column name in the table function derived table converted by the ADB_AUDITREAD function
	Type	Item name	Attribute name		
1	Header information	Common specification identifier	--	The string `CALFHM` is output. This string is output before the first item of all audit trails in common format audit trail files.	None

No.	Output item			Information that is output	Corresponding column name in the table function derived table converted by the ADB_AUDITREAD function
	Type	Item name	Attribute name		
2		Common specification revision number	--	The string `1.0` is output. This string is output before the first item of all audit trails in common format audit trail files.	None
3	Common information	Sequence number	`seqnum`	A sequence number in the range from 1 to 2,147,483,647 is output on each output line. If the value exceeds the maximum, the value is reset to 1. The value is also reset to 1 each time the `adbconvertaudittrailfile` command is executed.	None
4		Message ID	`msgid`	The value of the `EXIT_STATUS` column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output.	`EXIT_STATUS`
5		Date and time	`date`	The value of the `EXEC_TIME` column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output in the following format: *YYYY-MM-DD*`T`*hh*`:`*mm*`:`*ss*`.`*nnn*`TZD` *YYYY-MM-DD*: Year, month, and day T: Separator between the date and time *hh*:*mm*:*ss*.*nnn*: Hour, minute, second, and millisecond *TZD*: Time zone of the time at which the audit trail is output to the audit trail file[#1]	`EXEC_TIME`
6		Name of the relevant program	`progid`	The string `HADB` is output.	None
7		Name of the relevant component	`compid`	The value of the `ADBDIR` column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output within 64 bytes. If the data to be output is longer than 64 bytes, the heading bytes are omitted. A value is output including escape characters.[#2]	`ADBDIR`
8		ID of the relevant process	`pid`	The value of the `HADB_PROCESS_ID` column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output.	`HADB_PROCESS_ID`
9		Location	`ocp:host`	The value of the `HADB_HOST_NAME` column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output within 64 bytes.	`HADB_HOST_NAME`

No.	Output item			Information that is output	Corresponding column name in the table function derived table converted by the ADB_AUDITREAD function
	Type	Item name	Attribute name		
10		Event type	ctgry	The value of the EVENT_SUBTYPE column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output after being converted into the type as defined by the common format. For details about conversion, see Table 12-6: Correspondence between the event subtype values and audit event type values.	EVENT_SUBTYPE
11		Event result	result	The value of the EVENT_RESULT column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output.	EVENT_RESULT
12		Subject identification information	subj:uid	The value of the USER_NAME column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output. Note that this item is not output if the value of the USER_NAME column is NULL.	USER_NAME
13			subj:euid	The value of the OS_USER_NAME column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output within 100 bytes. Note that this item is output only if the value of the USER_NAME column is NULL.	OS_USER_NAME
14				For this item, an asterisk (*) is output if the values of both the OS_USER_NAME and USER_NAME columns in Table 12-10: Column structure of table function derived table when retrieving audit trails are NULL.	None
15	Specific information	Object information	obj	The value into which the values of the OBJECT_SCHEMA_NAME and OBJECT_NAME columns in Table 12-10: Column structure of table function derived table when retrieving audit trails are concatenated is output. For details about the output format, see Table 12-7: Output format for object information. A value is output including escape characters.#2	• OBJECT_SCHEMA_NAME • OBJECT_NAME
16		Operating information	op	The value of the EVENT_SUBTYPE column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output within 32 bytes. A value is output including escape characters.#2	EVENT_SUBTYPE

No.	Output item			Information that is output	Corresponding column name in the table function derived table converted by the ADB_AUDITREAD function
	Type	Item name	Attribute name		
17		Request-originating host	from:ipv4	The value of the CLIENT_IP_ADDRESS column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output.	CLIENT_IP_ADDRESS
18		Request-originating port number	from:port	The value of the CLIENT_PORT_NUMBER column in Table 12-10: Column structure of table function derived table when retrieving audit trails is output.	CLIENT_PORT_NUMBER
19		Message	msg	The name of the conversion-source audit trail file is output. The directory names are not included. A value is output including escape characters.#2	None

Legend:

--: Not applicable.

#1

The offset from UTC (Coordinated Universal Time) is indicated as the time zone. One of the following values is output:

- *+hh:mm*

  Indicates that the time is ahead of UTC (Coordinated Universal Time) by *hh*:*mm*.

- *-hh:mm*

  Indicates that the time is behind of UTC (Coordinated Universal Time) by *hh*:*mm*.

- *Z*

  Indicates that the time is the same as UTC (Coordinated Universal Time).

For example, JST (Japan Standard Time) is output as +09:00.

#2

A value that includes escape characters is enclosed by double quotation marks ("). A double quotation mark (") included as an ordinary character in an output value is replaced by two double quotation marks ("").

The double quotation marks (") enclosing a value that includes escape characters are included in the data length. Therefore, if the total length of data to be output and the double quotation marks (") added as a result of escaping is longer than the maximum, the heading or trailing bytes of the data are omitted.

The following table shows the correspondence between the event subtype values of HADB audit trails and the audit event type values in common format.

Table 12-6: Correspondence between the event subtype values and audit event type values

No.	Event category	Event type	Event subtype value output in audit trails	Audit event type value in common format
1	Mandatory audit event	System event	ADBSTART	StartStop
2			ADBSTOP	

No.	Event category	Event type	Event subtype value output in audit trails	Audit event type value in common format
3			ADBCHGSRVMODE	ConfigurationAccess
4			ADBCHGNODETYPE	
5			ADBCHGSQLTRC	
6			ADBCLIENTDEFMANG	
7			ADBMODAREA	
8			ADBMODBUFF	
9			ADBCOLUMNIZE	
10		Audit event	GRANT	AccessControl
11			REVOKE	
12			CREATE AUDIT	ConfigurationAccess
13			DROP AUDIT	
14			ALTER USER	AccessControl
15			ADBAUDITTRAIL START	ConfigurationAccess
16			ADBAUDITTRAIL STOP	
17			ADBAUDITTRAIL SWAP	
18			ADBAUDITTRAIL DISPLAY	
19			SELECT	ContentAccess
20			ADBEXPORT	
21			ADBCONVERTAUDITTRAILFILE	
22	Optional audit event	Session event	CONNECT	Authentication
23			DISCONNECT	
24		Privilege management event	GRANT	AccessControl
25			REVOKE	
26			CREATE USER	
27			DROP USER	
28			ALTER USER	
29		Definition SQL event	CREATE INDEX	ContentAccess
30			CREATE SCHEMA	
31			CREATE TABLE	
32			CREATE VIEW	
33			DROP INDEX	
34			DROP SCHEMA	
35			DROP TABLE	
36			DROP VIEW	
37			ALTER TABLE	

No.	Event category	Event type	Event subtype value output in audit trails	Audit event type value in common format
38			ALTER VIEW	
39		Data manipulation SQL event	SELECT	
40			INSERT	
41			UPDATE	
42			DELETE	
43			TRUNCATE TABLE	
44			PURGE CHUNK	
45			UNKNOWN	
46			GETDATA	
47			GETCOUNT	
48			TABLES	
49			COLUMNS	
50			INDEXES	
51			CHUNKS	
52			GETUSER	
53		Command operation event	ADBIMPORT	
54			ADBIDXREBUILD	
55			ADBGETCST	
56			ADBDBSTATUS	
57			ADBEXPORT	
58			ADBMERGECHUNK	
59			ADBCHGCHUNKCOMMENT	
60			ADBCHGCHUNKSTATUS	
61			ADBARCHIVECHUNK	
62			ADBUNARCHIVECHUNK	
63			ADBREORGSYSTEMDATA	
64			ADBSYNDICT	

The following table shows the output formats that can be used for object information. Object information is output in one of these formats according to the values of the OBJECT_SCHEMA_NAME and OBJECT_NAME columns.

Table 12-7: Output format for object information

No.	OBJECT_SCHEMA_NAME column	OBJECT_NAME column	Format	Output example
1	Non-null value	Non-null value	The values of the OBJECT_SCHEMA_NAME and OBJECT_NAME columns are	obj="ADBUSER01.T1"

No.	OBJECT_SCHEMA_NAME column	OBJECT_NAME column	Format	Output example
			concatenated with an intervening period (.).	
2	Null value	Non-null value	Only the value of the OBJECT_NAME column is output.	obj="ADBUSER01"#
3	Null value	Null value	No object information is output.	--

Legend:

--: Not applicable.

#

This is an output example when the CREATE SCHEMA statement is used. In this example, a schema name is output to the OBJECT_NAME column.

12.8.3 Environment settings for linking the audit trail facility with JP1/Audit

This subsection explains the environment settings for linking the audit trail facility with JP1/Audit.

To link the audit trail facility with JP1/Audit, you must specify the environment settings for both the HADB server and JP1/Audit.

(1) Environment settings for the HADB server

As an environment setting for the HADB server, you must prepare the *output-directory for common format audit trails*.

The following provides notes on creating the output-directory for common format audit trails:

- Create the output-directory for common format audit trails on a disk that has at least 4 gigabytes of free space. The following files are stored in the output-directory for common format audit trails:
  - Common format audit trail files (adbcommonaudit*XX*.log)

    Audit trail information converted by the adbconvertaudittrailfile command is output to these files. A maximum of four files are stored. The *XX* part in the file name is a sequential number in the range from 01 to 04.

    The maximum total size of four common format audit trail files is 2 gigabytes. However, when old *common format audit trail files* are switched to new ones, the data in the old and new files might temporarily coexist. For this reason, you must prepare a disk that has at least 4 gigabytes of free space.
  - Management file for common format audit trails (.adbcommonaudit)

    This file is used to manage the common format audit trail files. Only one file is stored.

> **Note**
>
> For details about switching the common format audit trail file, see *adbconvertaudittrailfile (Convert the Audit Trail File)* in the manual *HADB Command Reference*.

- The length of the full path of the output-directory for common format audit trails must be within 239 bytes including the slash (/) at the end of the path. If the length of the full path is 240 bytes or longer, JP1/Audit cannot set the path correctly.
- Assign read, write, and execution permissions to the output-directory for common format audit trails so that the HADB administrator can access the directory.
- You might want multiple HADB servers to operate on one server machine when, for example, you test operation. In such a case, prepare a separate output-directory for common format audit trails for each server environment. Multiple HADB servers must not share one output-directory for common format audit trails.

(2) Environment settings for JP1/Audit

JP1/Audit does not handle the HADB server as a program supported by default. Therefore, you must specify the following environment settings:

- Set up the event service of JP1/Base.
  For details, see (a) Setting up the event service of JP1/Base.
- Prepare the definition file for normalization rules.
  For details, see (b) Preparing the definition file for normalization rules.
- Create the definition file for operational behavior.
  For details, see (c) Creating the definition file for operational behavior.
- Create the definition file for product behavior.
  For details, see (d) Creating the definition file for product behavior.
- Edit the definition file for audit-log standard reports.
  For details, see (e) Editing the definition file for audit-log standard reports.
- Specify the audit log collection target settings.
  For details, see (f) Specifying the audit log collection target settings.

For details about how to specify these environment settings, see the *JP1/Base User's Guide*.

(a) Setting up the event service of JP1/Base

You must set up the event service of JP1/Base. Specify the size of the audit log event database with a proper value according to the number of audit trails output by the HADB server.

If you cannot estimate the number of audit trails converted by a single execution of the `adbconvertaudittrailfile` command, specify the maximum size of the audit log event database.

(b) Preparing the definition file for normalization rules

You do not need to newly define the contents of the definition file for normalization rules.

The audit trails output by the `adbconvertaudittrailfile` command to common format audit trail files conform to the common format for Hitachi open middleware products. Therefore, you can use the `admrglrule_CALFHM.conf` file, which is a standard file provided by JP1/Audit, as the common format definition file for normalization rules.

(c) Creating the definition file for operational behavior

Linkage with JP1/Audit uses the definition file for operational behavior. You must create this file. The creation procedure is as follows.

Procedure:

1. Copy the sample file for the definition file for operational behavior.
   Copy the sample file (`admjevlog_HADB.conf`) that is stored in the `$ADBDIR/sample/jp1audit` directory under the server directory. Then, save the sample file (`admjevlog_HADB.conf`) in the *JP1/Audit-Management-Manager-installation-folder*`\conf\logdef` folder.
2. Confirm the settings in the sample file for the definition file for operational behavior.
   You do not need to change the settings in the sample file. Confirm that the following settings are specified.
   Settings:

```
retry-times=60
retry-interval=10
FILETYPE=SEQ2
ACTDEF =<Information>1000 "^CALFHM"
```

You might want multiple HADB servers to operate on one server machine when, for example, you test operation. In such a case, you must create the definition file for operational behavior in each server environment. To do so, rename the definition files for operational behavior in the `admjevlog_HADB_`*`string-of-your-choice`*`.conf` format so that each file name is unique. Note that the values of the *string-of-your-choice* parts in file names must be shared with the definition files for product behavior (described in (d) Creating the definition file for product behavior).

(d) Creating the definition file for product behavior

Linkage with JP1/Audit uses the definition file for product behavior. You must create this file. The creation procedure is as follows.

Procedure:

1. Copy the sample file for the definition file for product behavior.
   Copy the sample file (`HADB.conf`) that is stored in the `$ADBDIR/sample/jp1audit` directory under the server directory. Then, save the sample file (`HADB.conf`) in the *JP1/Audit-Management-Manager-installation-folder*`\conf\product` folder.
2. Confirm the settings in the sample file for the definition file for product behavior.
   You do not need to change the settings in the sample file. Confirm that the following settings are specified.
   Settings:

```
AuditLogNum=4
AuditLogName=adbcommonaudit01.log
AuditLogName=adbcommonaudit02.log
AuditLogName=adbcommonaudit03.log
AuditLogName=adbcommonaudit04.log
RegularPattern=admrglrule_CALFHM.conf
ReadOnly=1
```

You might want multiple HADB servers to operate on one server machine when, for example, you test operation. In such a case, you must create the definition file for product behavior in each server environment. To do so, rename the definition files for product behavior in the `HADB_`*`string-of-your-choice`*`.conf` format so that each file name is unique. Note that the values of the *string-of-your-choice* parts in file names must be shared with the definition files for operational behavior (described in (c) Creating the definition file for operational behavior).

(e) Editing the definition file for audit-log standard reports

You must edit the definition file for audit-log standard reports. The following shows the procedure for editing the file.

Important

Normally, the definition file for audit-log reports is used for the audit trail information that JP1/Audit collected by using the definition file for normalization rules. However, the HADB server adopts the common format for Hitachi open middleware products. Therefore, you need to use the definition file for audit-log standard reports rather than the definition file for audit-log reports.

Procedure:

Add the following entries to the definition file for audit-log standard reports (`admAnalysis.ini`) stored in the *JP1/Audit-Management-Manager-virtual-directory*`\conf` folder.

Entries

```
[HADB]
TYPE=Common
```

Make sure that each line ends with a line break.

(f) Specifying the audit log collection target settings

You must specify the audit log collection target settings. Use the audit log collection manager of JP1/Audit to set the HADB server as the collection target.

The following shows the values that you need to specify in the **Set Collection Target** dialog box when setting the HADB server as the collection target.

Table 12-8: Values to be specified in the audit log collection target settings

No.	Item	Value to be set
1	Server	Specify the host name of the HADB server.
2	Program	Select the item that corresponds to the name of the definition file for product behavior from the drop-down menu. For example, if the name of the definition file for product behavior is `HADB.conf`, select **HADB**. Note that an underscore (_) is replaced by a slash (/).
3	Log folder	Specify the full path name of the output-directory for common format audit trails prepared in (1) Environment settings for the HADB server.
4	Comment	Write a comment freely.
5	Start monitoring audit logs when the OS starts	You do not need to set this item.

Important

Before JP1/Audit can collect the audit trails in common format audit trail files, JP1/Audit must have started monitoring of audit logs. When you finish specifying the environment settings of JP1/Audit, always start monitoring of audit logs. We recommend that you specify the OS settings so that monitoring of audit logs starts when the OS starts.

12.8.4 Operation methods available with linkage between the audit trail facility and JP1/Audit

This subsection describes the operation methods available with linkage with JP1/Audit. Either of the following two operation methods can be used according to when audit trail information is converted by using the `adbconvertaudittrailfile` command:

- Operation method 1

  The `adbconvertaudittrailfile` command is executed to convert audit trail information when the audit trail file is swapped.

  For details, see (1) Operation method 1 (conversion is performed when the audit trail file is swapped).
- Operation method 2

  The `adbconvertaudittrailfile` command is periodically executed to convert audit trail information.

  To perform conversion periodically, the `adbconvertaudittrailfile` command must be executed as many times as the number of audit trail files to be converted.

  For details, see (2) Operation method 2 (conversion is performed periodically).

Note that if multiple audit trail files are converted at the same time, audit trails being collected might be lost if the *common format audit trail file is switched*. In this case, JP1/Audit might fail to collect some audit trails. Therefore, we recommend that you use operation method 1 rather than operation method 2.

(1) Operation method 1 (conversion is performed when the audit trail file is swapped)

The following describes the operation method in which audit trails are converted when the audit trail file is swapped.

In this operation method, the system needs to monitor messages that are output when the current audit trail file is swapped, and to execute the `adbconvertaudittrailfile` command each time the audit trail file is swapped. We recommend that you create a job that automatically executes the `adbconvertaudittrailfile` command when the target messages are output.

Note that the audit trail file swapped when the HADB server is terminated cannot be converted until the next time the HADB server is started. Therefore, it is necessary to confirm that the HADB server is operating before executing the `adbconvertaudittrailfile` command.

Procedure:

1. Start monitoring the messages.

   Monitor the `KFAA81401-I` and `KFAA81402-I` messages that are output to the server message log file (`$ADBDIR/spool/adbmessage`*XX*`.log`[#]). These messages are output when the audit trail file is swapped. When the audit trail file is swapped, obtain the path of the file.

   #

   XX is a sequential number between `01` and `04`.
2. Move the audit trail file from the audit trail directory to the audit trail storage directory.

   Move the target audit trail file based on the audit trail file's path obtained in step 1.
3. Execute the `adbls -d srv` command.

   Confirm that the HADB server is operating. Check whether the HADB server's status that is output for the output item `STATUS` is one of the following statuses:

   - `ACTIVE`: Operating (normal mode)

- QUIESCE: Operating (quiescence mode)
- OFFLINE: Operating (offline mode)
- MAINTNCE: Operating (maintenance mode)

If the HADB server is not operating, wait for it to start (create a job that automatically waits for the HADB server to start). Before you go to the next step, make sure that the HADB server is started by, for example, the HADB administrator or another job.

Important

Do not include the processing that starts the HADB server in the same job that automatically executes the adbconvertaudittrailfile command. If you do so, the HADB server becomes unable to terminate. Therefore, if the HADB server is not operating, it must be started by using a means other than the job that automatically executes the adbconvertaudittrailfile command.

4. Execute the adbconvertaudittrailfile command.

   Use the adbconvertaudittrailfile command to convert the audit trail file that you moved in step 2.

Note

We recommend that you provide a means to swap the current audit trail file periodically in addition to the job that automatically executes the adbconvertaudittrailfile command. Use the adbaudittrail --swap command to swap the current audit trail file. Make sure that the job converts audit trails each time the adbaudittrail --swap command is executed periodically.

(2) Operation method 2 (conversion is performed periodically)

The following describes the operation method in which audit trails are converted periodically.

In this operation method, the adbconvertaudittrailfile command targets all the audit trail files stored in the *audit trail directory*. We recommend that you create and execute a batch program that executes the adbconvertaudittrailfile command periodically (for example, once a day or once a week).

Procedure:

1. Execute the adbls -d srv command.

   Confirm that the HADB server is operating. Check whether the HADB server's status that is output for the output item STATUS is one of the following statuses:

   - ACTIVE: Operating (normal mode)
   - QUIESCE: Operating (quiescence mode)
   - OFFLINE: Operating (offline mode)
   - MAINTNCE: Operating (maintenance mode)

   If the HADB server is not operating, start it. Then, go to the next step.

2. Create a list of audit trail files that need to be moved.

   Using the find command of the OS, search for audit trail files in the audit trail directory and create a list of audit trail files (/home/adbmanager/tmp/auditfilelist.txt) that need to be moved. Note that the following command excludes the current audit trail file from its results.

   Command execution example

```
find /mnt/audittrail/outputarea/audit \
    -name "adbaud-????????-??????-???*.aud" \
    > /home/adbmanager/tmp/auditfilelist.txt
```

3. Copy the audit trail files that need to be moved.

   Use the `cp` command of the OS to copy all of the files in the list (`/home/adbmanager/tmp/auditfilelist.txt`) that need to be moved from the audit trail directory to the audit trail storage directory.

```
while read filename ; do
  cp ${filename} /mnt/audittrail/shorttimesavearea/audit_bak
done < /home/adbmanager/tmp/auditfilelist.txt
```

4. Delete the audit trail files for which the copy process has completed.

   When the copy process has completed, use the `rm` command of the OS to delete the source files from the audit trail directory.

```
while read filename ; do
  rm ${filename}
done < /home/adbmanager/tmp/auditfilelist.txt
```

5. Execute the `adbconvertaudittrailfile` command.

   Use the `adbconvertaudittrailfile` command to convert all the audit trail files that have been moved, based on the list of audit trail files that need to be moved. This list is contained in the `/home/adbmanager/tmp/auditfilelist.txt` file created in step 2.

   Note that if multiple audit trail files are converted at the same time, audit trails being collected might be lost if the *common format audit trail file is switched*. In this case, JP1/Audit might fail to collect some audit trails.

6. Delete the list of audit trail files that need to be moved.

   Using the `rm` command of the OS, delete the list you created in step 2 (`/home/adbmanager/tmp/auditfilelist.txt`) of audit trail files that need to be moved.

```
rm /home/adbmanager/tmp/auditfilelist.txt
```

12.8.5 Performing auditing

After JP1/Audit has collected audit trails, you can search and aggregate them by using the audit log management window of JP1/Audit from a web browser.

Note that the information that the `adbconvertaudittrailfile` command outputs to common format audit trail files is only the audit trail information that can be displayed with JP1/Audit. The information that can be displayed with JP1/Audit is sometimes insufficient for auditing. For example, JP1/Audit cannot be used to check the SQL statements that were executed and the number of times a table was accessed.

You can also view more details about the audit trail information that can be displayed with JP1/Audit. To do so, execute an SQL statement with the `ADB_AUDITREAD` function specified for the audit trail files whose names are recorded as **specific information** of JP1/Audit, and then view the resulting audit trail information. For details about an SQL statement with the `ADB_AUDITREAD` function specified, see 12.3.3 Referencing audit trails (when using SELECT statements to reference audit trails).

12.9 Audit target events and column structure of table function derived tables

This section describes audit target events and the column structure of table function derived tables.

12.9.1 List of audit target events and output items

This section provides a list of audit target events and the associated output items.

The following table lists these audit target events and output items.

Table 12-9: List of audit target events and output items

No.	Event category	Event type	Event Description
1	Mandatory audit event	System event	These events are used to audit activity that involves starting, stopping, and modifying the system. An audit trail is output when any of the following events occur: • Starting the HADB server[#1] (`adbstart` command) • Stopping the HADB server[#2] (`adbstop` command) • Changing the HADB server operation mode (`adbchgsrvmode` command) • Changing the node type (`adbchgnodetype` command) • Starting and stopping output of SQL trace information (`adbchgsqltrc` command) • Centrally managing client definitions (`adbclientdefmang` command) • Adding and modifying DB areas (`adbmodarea` command) • Changing a buffer (`adbmodbuff` command) • Managing the updated-row columnizing facility (`adbcolumnize` command)
2		Audit event	These events are used to audit operations related to the auditing process itself. An audit trail is output when any of the following events occur: • Granting audit admin privilege (`GRANT` statement with `AUDIT ADMIN` specified) • Granting audit viewer privilege (`GRANT` statement with `AUDIT VIEWER` specified) • Revoking audit admin privilege (`REVOKE` statement with `AUDIT ADMIN` specified) • Revoking audit viewer privilege (`REVOKE` statement with `AUDIT VIEWER` specified) • Defining audit targets (`CREATE AUDIT` statement) • Deleting audit target definition information (`DROP AUDIT` statement) • Referencing audit target definition information (retrieval from `SQL_AUDITS` dictionary table) • Retrieving data from viewed tables that depend on the `SQL_AUDITS` dictionary table • Changing auditor passwords (`ALTER USER` statement) • Enabling the audit trail facility (`adbauditttrail` command with `--start` option specified) • Disabling the audit trail facility (`adbauditttrail` command with `--stop` option specified) • Swapping the audit trail file (`adbauditttrail` command with `--swap` option specified) • Referencing information related to the audit trail facility (`adbauditttrail` command with `-d` option specified)

No.	Event category	Event type	Event Description
			• Using a system-defined function for audit trails (ADB_AUDITREAD function) • Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function • Converting audit trail files (adbconvertaudittrailfile command)
3	Optional audit event	Session event	These events are used to audit user authorization based on the authorization identifiers and passwords of HADB users. An audit trail is output when any of the following events occur: • Connection to the HADB server • Disconnection from the HADB server
4		Privilege management event	These events are used to audit the granting or revocation of privileges. An audit trail is output when any of the following events occur: • Granting privileges (GRANT statement) • Revoking privileges (REVOKE statement) • Creating an HADB user (CREATE USER statement) • Deleting an HADB user (DROP USER statement) • Changing user information for an HADB user (ALTER USER statement)

5		Definition SQL event	These events are used to audit the definition, deletion, and modification of objects.
			An audit trail is output when any of the following events occur:
			• Defining an object
			  • CREATE INDEX statement
			  • CREATE SCHEMA statement
			  • CREATE TABLE statement
			  • CREATE VIEW statement
			• Deleting an object
			  • DROP INDEX statement
			  • DROP SCHEMA statement
			  • DROP TABLE statement
			  • DROP VIEW statement
			• Modifying an object
			  • ALTER TABLE statement
			  • ALTER VIEW statement

6		Data manipulation SQL event	These events are used to audit access to objects. An audit trail is output when any of the following events occur: • Retrieving data from tables (SELECT statement) • Inserting rows into tables (INSERT statement) • Updating table row data (UPDATE statement) • Deleting table row data (DELETE statement) • Deleting all row data from a table (TRUNCATE TABLE statement) • Deleting all row data in a chunk (PURGE CHUNK statement) • SQL parsing error • Acquiring data stored in chunks (#GETDATA subcommand of adbsql command) • Acquiring the number of data items stored in a chunk (#GETCOUNT subcommand of adbsql command) • Displaying table information (#TABLES subcommand of adbsql command) • Displaying column information (#COLUMNS subcommand of adbsql command) • Displaying index information (#INDEXES subcommand of adbsql command) • Displaying chunk information (#CHUNKS subcommand of adbsql command) • Displaying authorization identifiers (#GETUSER subcommand of adbsql command)

No.	Event category	Event type	Event Description
7		Command operation event[#3]	These events are used to audit execution of commands that connect to the HADB server. An audit trail is output when any of the following events occur: • Importing data (`adbimport` command) • Rebuilding indexes (`adbidxrebuild` command) • Collecting cost information (`adbgetcst` command) • Analyzing DB status (`adbdbstatus` command) • Exporting data (`adbexport` command) • Merging chunks (`adbmergechunk` command) • Setting, changing, and deleting chunk comments (`adbchgchunkcomment` command) • Change the chunk status (`adbchgchunkstatus` command) • Archiving chunks (`adbarchivechunk` command) • Unarchiving chunks (`adbunarchivechunk` command) • Reorganizing system tables (base tables) (`adbreorgsystemdata` command) • Registering and deleting synonym dictionaries (`adbsyndict` command)

#1

Subject to auditing when the HADB server is able to be started.

#2

Stopping the HADB server is subject to auditing except when the `adbstop` command is executed with the `--force` option.

#3

The `adbsql` command is not audited as a command operation event. Operations that are implemented using the `adbsql` command are recorded as other types of events.

Operations that are not explicitly performed by an HADB user are not subject to auditing. Specifically, the following information is not subject to auditing:

- SQL statements executed internally by a subcommand of the `adbsql` command
- SQL statements executed internally by a command[#]
- Object deletion or privilege revocation performed recursively by `CASCADE` specification when `CASCADE` is specified or omitted in the following SQL statements:
  - `DROP USER` statement
  - `DROP SCHEMA` statement
  - `DROP TABLE` statement
  - `REVOKE` statement

#

SQL statements specified in the `-q` option of the `adbexport` command are subject to auditing.

12.9.2 Column structure of table function derived table when retrieving audit trails

This section explains the column structure of table function derived tables when retrieving audit trails.

The following table shows the column structure of table function derived tables when retrieving audit trails:

Table 12-10: Column structure of table function derived table when retrieving audit trails

No.	Column name	Column data type	Description	NOT NULL constraint
1	HADB_VERSION	CHAR(8)	The version of the HADB server that received the event.	None
2	AUDIT_TRAIL_TYPE	VARCHAR(32)	The event that caused the audit trail to be acquired. • EVENT (event termination)	None
3	EXEC_TIME	TIMESTAMP(6)	The time at which the event finished.	None
4	USER_NAME	VARCHAR(100)	The authorization identifier of the HADB user who executed the event.	None
5	EVENT_TYPE	VARCHAR(32)	The type of the event. • SYSTEM (system event) • AUDIT (audit event) • SESSION (session event) • PRIVILEGE (privilege management event) • DEFINE (definition SQL event) • MANIPULATE (data manipulation SQL event) • COMMAND (command operation event)	None
6	EVENT_SUBTYPE	VARCHAR(64)	The subtype of the event. For details, see Table 12-11: Event subtypes output in audit trails.	None
7	EVENT_RESULT	CHAR(10)	Whether the event was successful. • SUCCESS • FAILURE • OCCURRENCE# # This type might be output as the event result of event type DISCONNECT. For details, see (2) Notes about executing the adbcancel command in 12.10.2 Notes about audit trails output during command execution.	None
8	USED_PRIVILEGE	VARCHAR(32)	If EVENT is output in the AUDIT_TRAIL_TYPE column, NULL is output in this column.	None
9	OS_USER_NAME	VARCHAR(256)	The OS account name of the client that issued the event. If the OS account name cannot be identified, NULL is output.	None
10	AP_NAME	VARCHAR(30)	The application identifier of the event issuer. If a command was executed that connects to the HADB server, the command name is output as the application identifier. If the application identifier of the event issuer cannot be identified, NULL is output.	None
11	CLIENT_IP_ADDRESS#	VARCHAR(46)	The IP address of the client that issued the event. If the IP address of the client that issued the event cannot be identified, NULL is output.	None
12	CLIENT_PORT_NUMBER#	SMALLINT	The port number of the client that issued the event. If the port number of the client that issued the event cannot be identified, NULL is output.	None

No.	Column name	Column data type	Description	NOT NULL constraint
13	CLIENT_PROCESS_NAME	VARCHAR(255)	The process name of the client that issued the event. If either of the following applies, NULL is output: • The client that issued the event is the JDBC driver • The process name of the client that issued the event cannot be identified	None
14	CLIENT_PROCESS_ID	SMALLINT	The process ID of the client that issued the event. If the process ID of the client that issued the event cannot be identified, NULL is output.	None
15	HADB_HOST_NAME	VARCHAR(255)	The host name of the HADB server that received the event. If the host name of the HADB server cannot be identified, NULL is output.	None
16	HADB_IP_ADDRESS	VARCHAR(46)	The IP address of the HADB server that received the event. If the IP address of the HADB server cannot be identified, NULL is output.	None
17	HADB_PORT_NUMBER	SMALLINT	The port number of the HADB server that received the event. If the port number of the HADB server cannot be identified, NULL is output.	None
18	HADB_PROCESS_ID	SMALLINT	The process ID of the HADB server that received the event. If the process ID of the HADB server cannot be identified, NULL is output.	None
19	ADBDIR	VARCHAR(118)	The path name of the server directory specified in the environment variable ADBDIR. If the specified path name is longer than 118 bytes, only the first 118 bytes are output.	None
20	HADB_NODE_NUMBER	SMALLINT	When using the multi-node function, the node number (1 to 4) of the HADB server that received the event is output. If the multi-node function is not being used, NULL is output.	None
21	CONNECTION_ID	SMALLINT	The connection ID (1 to 1,024) of the connection that executed the event. If the connection ID cannot be identified, NULL is output.	None
22	CONNECTION_NUMBER	INTEGER	The connection sequence number (1 to 4,294,967,295) of the connection that executed the event. If the connection sequence number cannot be identified, NULL is output.	None
23	STATEMENT_HANDLE	SMALLINT	The statement handle (1 to 4,095) used when executing the event. If the statement handle cannot be identified, NULL is output.	None
24	SQL_SERIAL_NUMBER	INTEGER	The serial number of the SQL statement (1 to 9,223,372,036,854,775,807). If the serial number of the SQL statement cannot be identified, NULL is output.	None
25	MESSAGE_LOG_INFO	CHAR(20)	The message log information associated with the event. If the message log information cannot be identified, NULL is output.	None
26	EXIT_STATUS	SMALLINT	If EVENT is output in the AUDIT_TRAIL_TYPE column, either of the following is output:	None

No.	Column name	Column data type	Description	NOT NULL constraint
			• If a SQL statement was executed, SQLCODE is output. • If a command was executed, the return code is output.	
27	OBJECT_TYPE	VARCHAR(32)	The type of object specified as the target of the event. • INDEX (index) • SCHEMA (schema) • TABLE (base table) • TABLE FUNCTION (table function derived table) • VIEW (viewed table) If the object type cannot be identified, NULL is output. For details about the circumstances in which audit trails are output for each object type, see Table 12-19: Circumstances in which object types are output in audit trails.	None
28	OBJECT_OWNER_NAME	VARCHAR(100)	The authorization identifier of the owner of the object specified as the target of the event. If the object owner cannot be identified, NULL is output.	None
29	OBJECT_SCHEMA_NAME	VARCHAR(100)	The schema name of the object specified as the target of the event. If the schema name of the object cannot be identified, NULL is output.	None
30	OBJECT_NAME	VARCHAR(100)	The identifier of the object specified as the target of the event. If the object identifier cannot be identified, NULL is output.	None
31	ACCESS_COUNT	INTEGER	The number of rows of the object (base table or viewed table) that were accessed by the event. The following are included in the number of rows: • The number of retrieved rows (including data export) • The number of inserted rows (including data import) • The number of updated rows • The number of deleted rows If the number of accessed rows cannot be acquired, NULL is output. For details about the number of rows output in relation to each operation, see Table 12-20: Number of rows output in ACCESS_COUNT column.	None
32	PRIVILEGE_TYPE	VARCHAR(32)	The name of the privilege specified for granting or revocation by the event. For details about privilege names, see Table 12-12: Privilege names output in audit trails. If the privilege name cannot be identified, NULL is output.	None
33	PRIVILEGE_USER_NAME	VARCHAR(100)	The authorization identifier of an HADB user to whom either of the following applies: • The authorization identifier of an HADB user whose privilege is specified for granting or revocation • The authorization identifier of the HADB user specified when creating, deleting or changing the information of an HADB user If the authorization identifier of the HADB user cannot be identified, NULL is output.	None

No.	Column name	Column data type	Description	NOT NULL constraint
34	SQL_SOURCE	VARCHAR(64000)	The SQL statement executed in the event. If one SQL statement specifies multiple objects, the SQL statement is output in the audit trail for each object. The SQL statement is truncated after the first 64,000 bytes. If no SQL statement was executed, NULL is output.	None
35	PARAM	VARCHAR(64000)	The dynamic parameters of the SQL statement executed in the event. If one SQL statement specifies multiple objects, the dynamic parameters are output in the audit trail for each object. The dynamic parameters are separated by commas (,). For example: `dynamic-parameter,dynamic-parameter,dynamic-parameter,...` The list of dynamic parameters is truncated after the first 64,000 bytes. If no dynamic parameters are specified, NULL is output. For details about the display format of each data type of data bound to dynamic parameters, see Table 12-13: Output format for dynamic parameters.	None
36	BEFORE_SYSTEM_INFO	VARCHAR(2000)	The system information before being changed. This information is output when the following events are successful: • adbstart The HADB server operation mode. For details about the output information, see **■ Values output when an event (adbstart or adbchgsrvmode) is successful**. • adbchgsrvmode The HADB server operation mode. For details about the output information, see **■ Values output when an event (adbstart or adbchgsrvmode) is successful**. • adbchgnodetype The node type. For details about the output information, see **■ Values output when an event (adbchgnodetype) is successful**. • adbchgsqltrc The trace information and trace level that are in effect. For details about the output information, see **■ Values output when an event (adbchgsqltrc) is successful**. • adbclientdefmang The associations between the client definition file and the HADB user. For details about the output information, see **■ Values output when an event (adbclientdefmang) is successful**. • adbmodarea The DB area information before being changed. For details about the output information, see **■ Values output when an event (adbmodarea) is successful**. • adbmodbuff	None

No.	Column name	Column data type	Description	NOT NULL constraint
			The number of pages in the buffer for the local work table. For details about the output information, see ■ **Values output when an event (adbmodbuff) is successful**. • adbcolumnize The information about the status of the updated-row columnizing facility. For details about the output information, see ■ **Values output when an event (adbcolumnize) is successful**. For events other than the preceding, NULL is output. The system information before being changed is truncated after the first 2,000 bytes. If there were no system changes, the information in this column is the system information in effect when the event was executed. This means that the BEFORE_SYSTEM_INFO column and the AFTER_SYSTEM_INFO column will contain the same information.	
37	AFTER_SYSTEM_INFO	VARCHAR(2000)	System information after being changed. This information is output when the following events are successful: • adbstart The HADB server operation mode. For details about the output information, see ■ **Values output when an event (adbstart or adbchgsrvmode) is successful**. • adbchgsrvmode The HADB server operation mode. For details about the output information, see ■ **Values output when an event (adbstart or adbchgsrvmode) is successful**. • adbchgnodetype The node type. For details about the output information, see ■ **Values output when an event (adbchgnodetype) is successful**. • adbchgsqltrc The trace information and trace level that are in effect. For details about the output information, see ■ **Values output when an event (adbchgsqltrc) is successful**. • adbclientdefmang The associations between the client definition file and the HADB user. For details about the output information, see ■ **Values output when an event (adbclientdefmang) is successful**. • adbmodarea The DB area information after the change. For details about the output information, see ■ **Values output when an event (adbmodarea) is successful**. • adbmodbuff The number of pages in the buffer for the local work table. For details about the output information, see ■ **Values output when an event (adbmodbuff) is successful**. • adbcolumnize	None

No.	Column name	Column data type	Description	NOT NULL constraint
			The information about the status of the updated-row columnizing facility. For details about the output information, see **■ Values output when an event (adbcolumnize) is successful**. For events other than the preceding, NULL is output. The system information after being changed is truncated after the first 2,000 bytes. If there were no system changes, the information in this column is the system information in effect when the event was executed. This means that the BEFORE_SYSTEM_INFO column and the AFTER_SYSTEM_INFO column will contain the same information.	
38	SERVER_OPERAND	VARCHAR(4000)	The values of the operands specified in the server definition when starting the HADB server. The operands are separated by slashes (/). The server definition information is truncated after the first 4,000 bytes. NULL is output unless the HADB server is starting (that is unless ADBSTART is in the EVENT_SUBTYPE column). For details about the format of the server definition information output in the SERVER_OPERAND column, see Table 12-14: Output format of server definition.	None
39	USER_INFO_1	VARCHAR(100)	Additional information set by the HADB user. In either of the following cases, NULL is output: • Additional information set by the HADB user cannot be identified • The HADB user has not set any additional information	None
40	USER_INFO_2	VARCHAR(100)		None
41	USER_INFO_3	VARCHAR(100)		None

#

The following information is output as the audit trail for a SQL statement executed by an HADB client on the same machine as the HADB server:

- IP address: 127.0.0.1
- Port number: NULL

When using the multi-node function, the information in the audit trail for a SQL statement executed by an HADB client on the same machine as the HADB server might differ depending on the node that processed the SQL statement.

The following table shows the *event subtypes* output in the EVENT_SUBTYPE column in Table 12-10: Column structure of table function derived table when retrieving audit trails.

Table 12-11: Event subtypes output in audit trails

No.	Event category	Event type	Event subtype	Event subtype value output in audit trail
1	Mandatory audit event	System event	Starting the HADB server (adbstart command)	ADBSTART
2			Stopping the HADB server (adbstop command)	ADBSTOP
3			Changing the HADB server operation mode (adbchgsrvmode command)	ADBCHGSRVMODE
4			Changing the node type (adbchgnodetype command)	ADBCHGNODETYPE

No.	Event category	Event type	Event subtype	Event subtype value output in audit trail
5			Starting and stopping output of SQL trace information (adbchgsqltrc command)	ADBCHGSQLTRC
6			Centrally managing client definitions (adbclientdefmang command)	ADBCLIENTDEFMANG
7			Adding and modifying DB areas (adbmodarea command)	ADBMODAREA
8			Changing a buffer (adbmodbuff command)	ADBMODBUFF
9			Managing the updated-row columnizing facility (adbcolumnize command)	ADBCOLUMNIZE
10		Audit event	Granting audit admin privilege (GRANT statement with AUDIT ADMIN specified)	GRANT
11			Granting audit viewer privilege (GRANT statement with AUDIT VIEWER specified)	
12			Revoking audit admin privilege (REVOKE statement with AUDIT ADMIN specified)	REVOKE
13			Revoking audit viewer privilege (REVOKE statement with AUDIT VIEWER specified)	
14			Defining audit targets (CREATE AUDIT statement)	CREATE AUDIT
15			Deleting audit target definitions (DROP AUDIT statement)	DROP AUDIT
16			Changing auditor passwords (ALTER USER statement)	ALTER USER
17			Enabling the audit trail facility (adbauditttrail command with --start option specified)	ADBAUDITTRAIL START
18			Disabling the audit trail facility (adbauditttrail command with --stop option specified)	ADBAUDITTRAIL STOP
19			Swapping the audit trail file (adbauditttrail command with --swap option specified)	ADBAUDITTRAIL SWAP
20			Referencing information related to the audit trail facility (adbauditttrail command with -d option specified)	ADBAUDITTRAIL DISPLAY
21			• Referencing audit target definition information (retrieval from SQL_AUDITS dictionary table) • Retrieving data from viewed tables that depend on the SQL_AUDITS dictionary table	SELECT
22				ADBEXPORT
23			• Using a system-defined function for audit trails (ADB_AUDITREAD function) • Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function	SELECT
24				ADBEXPORT
25			Converting audit trail files (adbconvertaudittrailfile command)	ADBCONVERTAUDITTRAILFILE
26	Optional audit event	Session event	Connecting to the HADB server	CONNECT
27			Disconnecting from the HADB server	DISCONNECT
28		Privilege management event	Granting the DBA privilege (GRANT statement with DBA specified)	GRANT

No.	Event category	Event type	Event subtype	Event subtype value output in audit trail
29			Granting the CONNECT privilege (GRANT statement with CONNECT specified)	
30			Granting the schema definition privilege (GRANT statement with SCHEMA specified)	
31			Granting the SELECT privilege (GRANT statement with SELECT specified)	
32			Granting the INSERT privilege (GRANT statement with INSERT specified)	
33			Granting the UPDATE privilege (GRANT statement with UPDATE specified)	
34			Granting the DELETE privilege (GRANT statement with DELETE specified)	
35			Granting the TRUNCATE privilege (GRANT statement with TRUNCATE specified)	
36			Granting the REFERENCES privilege (GRANT statement with REFERENCES specified)	
37			Granting the IMPORT TABLE privilege (GRANT statement with IMPORT TABLE specified)	
38			Granting the REBUILD INDEX privilege (GRANT statement with REBUILD INDEX specified)	
39			Granting the GET COSTINFO privilege (GRANT statement with GET COSTINFO specified)	
40			Granting the EXPORT TABLE privilege (GRANT statement with EXPORT TABLE specified)	
41			Granting the MERGE CHUNK privilege (GRANT statement with MERGE CHUNK specified)	
42			Granting the CHANGE CHUNK COMMENT privilege (GRANT statement with CHANGE CHUNK COMMENT specified)	
43			Granting the CHANGE CHUNK STATUS privilege (GRANT statement with CHANGE CHUNK STATUS specified)	
44			Granting the ARCHIVE CHUNK privilege (GRANT statement with ARCHIVE CHUNK specified)	
45			Granting the UNARCHIVE CHUNK privilege (GRANT statement with UNARCHIVE CHUNK specified)	
46			Revoking the DBA privilege (REVOKE statement with DBA specified)	REVOKE
47			Revoking the CONNECT privilege (REVOKE statement with CONNECT specified)	
48			Revoking the schema definition privilege (REVOKE statement with SCHEMA specified)	
49			Revoking the SELECT privilege (REVOKE statement with SELECT specified)	

No.	Event category	Event type	Event subtype	Event subtype value output in audit trail
50			Revoking the INSERT privilege (REVOKE statement with INSERT specified)	
51			Revoking the UPDATE privilege (REVOKE statement with UPDATE specified)	
52			Revoking the DELETE privilege (REVOKE statement with DELETE specified)	
53			Revoking the TRUNCATE privilege (REVOKE statement with TRUNCATE specified)	
54			Revoking the REFERENCES privilege (REVOKE statement with REFERENCES specified)	
55			Revoking the IMPORT TABLE privilege (REVOKE statement with IMPORT TABLE specified)	
56			Revoking the REBUILD INDEX privilege (REVOKE statement with REBUILD INDEX specified)	
57			Revoking the GET COSTINFO privilege (REVOKE statement with GET COSTINFO specified)	
58			Revoking the EXPORT TABLE privilege (REVOKE statement with EXPORT TABLE specified)	
59			Revoking the MERGE CHUNK privilege (REVOKE statement with MERGE CHUNK specified)	
60			Revoking the CHANGE CHUNK COMMENT privilege (REVOKE statement with CHANGE CHUNK COMMENT specified)	
61			Revoking the CHANGE CHUNK STATUS privilege (REVOKE statement with CHANGE CHUNK STATUS specified)	
62			Revoking the ARCHIVE CHUNK privilege (REVOKE statement with ARCHIVE CHUNK specified)	
63			Revoking the UNARCHIVE CHUNK privilege (REVOKE statement with UNARCHIVE CHUNK specified)	
64			Creating an HADB user (CREATE USER statement)	CREATE USER
65			Deleting an HADB user (DROP USER statement)	DROP USER
66			Changing user information for an HADB user (ALTER USER statement)	ALTER USER
67		Definition SQL event	Defining an index (CREATE INDEX statement)	CREATE INDEX
68			Defining a schema (CREATE SCHEMA statement)	CREATE SCHEMA
69			Defining a base table (CREATE TABLE statement)	CREATE TABLE
70			Defining a viewed table (CREATE VIEW statement)	CREATE VIEW
71			Deleting an index (DROP INDEX statement)	DROP INDEX
72			Deleting a schema (DROP SCHEMA statement)	DROP SCHEMA
73			Deleting a base table (DROP TABLE statement)	DROP TABLE
74			Deleting a viewed table (DROP VIEW statement)	DROP VIEW

No.	Event category	Event type	Event subtype	Event subtype value output in audit trail
75			Changing a table definition (ALTER TABLE statement)	ALTER TABLE
76			Changing a viewed table definition (ALTER VIEW statement)	ALTER VIEW
77		Data manipulation SQL event	Retrieving data from tables (SELECT statement)	SELECT
78			Inserting rows into tables (INSERT statement)	INSERT
79			Updating table row data (UPDATE statement)	UPDATE
80			Deleting table row data (DELETE statement)	DELETE
81			Deleting all row data from a table (TRUNCATE TABLE statement)	TRUNCATE TABLE
82			Deleting all row data in a chunk (PURGE CHUNK statement)	PURGE CHUNK
83			SQL parsing error	UNKNOWN
84			Acquiring data stored in chunks (#GETDATA subcommand of adbsql command)	GETDATA
85			Acquiring the number of data items stored in a chunk (#GETCOUNT subcommand of adbsql command)	GETCOUNT
86			Displaying table information (#TABLES subcommand of adbsql command)	TABLES
87			Displaying column information (#COLUMNS subcommand of adbsql command)	COLUMNS
88			Displaying index information (#INDEXES subcommand of adbsql command)	INDEXES
89			Displaying chunk information (#CHUNKS subcommand of adbsql command)	CHUNKS
90			Displaying authorization identifiers (#GETUSER subcommand of adbsql command)	GETUSER
91		Command operation event	Importing data (adbimport command)	ADBIMPORT
92			Rebuilding indexes (adbidxrebuild command)	ADBIDXREBUILD
93			Collecting cost information (adbgetcst command)	ADBGETCST
94			Analyzing DB status (adbdbstatus command)	ADBDBSTATUS
95			Exporting data (adbexport command)	ADBEXPORT
96			Merging chunks (adbmergechunk command)	ADBMERGECHUNK
97			Setting, changing, and deleting chunk comments (adbchgchunkcomment command)	ADBCHGCHUNKCOMMENT
98			Changing the chunk status (adbchgchunkstatus command)	ADBCHGCHUNKSTATUS
99			Archiving chunks (adbarchivechunk command)	ADBARCHIVECHUNK
100			Unarchiving chunks (adbunarchivechunk command)	ADBUNARCHIVECHUNK
101			Reorganizing system tables (base tables) (adbreorgsystemdata command)	ADBREORGSYSTEMDATA

No.	Event category	Event type	Event subtype	Event subtype value output in audit trail
102			Registering and deleting synonym dictionaries (`adbsyndict` command)	`ADBSYNDICT`

The following table shows the *privilege names* output in the `PRIVILEGE_TYPE` column in Table 12-10: Column structure of table function derived table when retrieving audit trails.

Table 12-12: Privilege names output in audit trails

No.	Privilege type	Privilege name output in audit trails
1	`DBA` privilege	`DBA`
2	`CONNECT` privilege	`CONNECT`
3	Schema definition privilege	`SCHEMA`
4	Audit admin privilege	`AUDIT ADMIN`
5	Audit viewer privilege	`AUDIT VIEWER`
6	`SELECT` privilege	`SELECT`
7	`INSERT` privilege	`INSERT`
8	`UPDATE` privilege	`UPDATE`
9	`DELETE` privilege	`DELETE`
10	`TRUNCATE` privilege	`TRUNCATE`
11	`REFERENCES` privilege	`REFERENCES`
12	`IMPORT TABLE` privilege	`IMPORT TABLE`
13	`REBUILD INDEX` privilege	`REBUILD INDEX`
14	`GET COSTINFO` privilege	`GET COSTINFO`
15	`EXPORT TABLE` privilege	`EXPORT TABLE`
16	`MERGE CHUNK` privilege	`MERGE CHUNK`
17	`CHANGE CHUNK COMMENT` privilege	`CHANGE CHUNK COMMENT`
18	`CHANGE CHUNK STATUS` privilege	`CHANGE CHUNK STATUS`
19	`ARCHIVE CHUNK` privilege	`ARCHIVE CHUNK`
20	`UNARCHIVE CHUNK` privilege	`UNARCHIVE CHUNK`

The following table shows the data types of data bound to dynamic parameters output in the `PARAM` column in Table 12-10: Column structure of table function derived table when retrieving audit trails.

Table 12-13: Output format for dynamic parameters

No.	Data type	Output format
1	`INTEGER`	Decimal
2	`DECIMAL`(*m*, *n*)	Decimal
3	`SMALLINT`	Decimal

No.	Data type	Output format
4	DOUBLE PRECISION	Mantissa (decimal) and exponent (decimal)
5	CHARACTER(*n*)	'character-string'
6	VARCHAR(*n*)	'character-string'[#] If the actual length is 0, the value is output as ''.
7	DATE	Predefined output representation for date data
8	TIME(*p*)	Predefined output representation for time data
9	TIMESTAMP(*p*)	Predefined output representation for time stamp data
10	BINARY(*n*)	X'hexadecimal-value'
11	VARBINARY(*n*)	X'hexadecimal-value' If the actual length is 0, the value is output as X''.
12	ROW	X'hexadecimal-value'

Legend:

m, *n*, and *p*: Positive integer

Note:

If the data bound to the dynamic parameter is a null value, NULL is output in the PARAM column.

#

If only a dynamic parameter is specified in the value expression of a NULL predicate and data that is not the null value is specified in the dynamic parameter, '*' is output regardless of the value that is specified.

The following table shows the output format of server definition entries in the SERVER_OPERAND column in Table 12-10: Column structure of table function derived table when retrieving audit trails.

Table 12-14: Output format of server definition

No.	Format	Output format
1	set format	*operand-name* = *specified-value*[#1]
2	Command format	*command-name option-name specified-value*[, *option-name specified-value* ...][#2]

#1

The maximum length of the = *specified-value* portion is 1,024 bytes. Anything longer is truncated from the 1,025th byte.

#2

The maximum length of the *option-name specified-value*[, *option-name specified-value* ...] portion is 1,024 bytes. Anything longer is truncated from the 1,025th byte.

The following shows a specification example of the server definition, and an example of the information output in the SERVER_OPERAND column. A slash (/) delimits each valid pair of a server definition operand and its value in the output information.

■ Server definition specification example

```
set adb_db_path = XXXXX
set adb_rpc_port = YYYYY
set adb_sys_max_users = 10
set adb_sys_rthd_num = 40
```

```
set adb_sys_uthd_num = 128
set adb_sql_exe_max_rthd_num = 4
set adb_sys_memory_limit = 64000
adbbuff -g TBLBUF01 -n ADBUTBL01 -p 1000000 -v 1024
adbbuff -g IDXBUF01 \
        -n ADBUIDX01 \
        -p 2500000
adbcltgrp -g group -m 10 -u 0 -r 40 -e 0 -w 0
```

■ **Output example of `SERVER_OPERAND` column**

```
adb_db_path = XXXXX / adb_rpc_port = YYYYY / adb_sys_max_users = 10 / adb_sys_rthd
_num = 40 / adb_sys_uthd_num = 128 / adb_sql_exe_max_rthd_num = 4 /
adb_sys_memory_limit = 64000 / adbbuff -g TBLBUF01 -n ADBUTBL01 -p 1000000 -v 1024
 / adbbuff -g IDXBUF01 -n ADBUIDX01 -p 2500000 / adbcltgrp -g group -m 10 -u 0 -r
40 -e 0 -w 0
```

The following explains the values output for each event in the `BEFORE_SYSTEM_INFO` and `AFTER_SYSTEM_INFO` columns in Table 12-10: Column structure of table function derived table when retrieving audit trails,

■ **Values output when an event (`adbstart` or `adbchgsrvmode`) is successful**

When an event (`adbstart` or `adbchgsrvmode`) is successful, the values shown in the following table are output to represent the HADB server operation mode:

Table 12-15: HADB server operation mode values

No.	HADB server operation mode	Value
1	Normal mode	`NORMAL`
2	Quiescence mode	`QUIESCENCE`
3	Offline mode	`OFFLINE`
4	Maintenance mode	`MAINTENANCE`

When using the multi-node function, the `BEFORE_SYSTEM_INFO` column for the master node displays the HADB server operation mode for the node that was the master node the last time the HADB server started. The `BEFORE_SYSTEM_INFO` column for the slave node always displays `NULL`.

■ **Values output when an event (`adbchgnodetype`) is successful**

When an event (`adbchgnodetype`) is successful, the values shown in the following table are output to represent the node type:

Table 12-16: Node type values

No.	Node type	Value
1	Master node	`MASTER`
2	Slave node	`SLAVE`

■ **Values output when an event (`adbchgsqltrc`) is successful**

When an event (`adbchgsqltrc`) is successful, the values shown in the following two tables are output to represent the trace information and trace level:

Table 12-17: Trace information values

No.	Trace information	Value
1	Access path information	`accesspath`
2	Dynamic parameter information	`param`

Table 12-18: Trace level value

No.	Trace level	Value
1	Output at SQL statement level	sql
2	Output at call level	call

When an event (adbchgsqltrc) is successful, the trace information and trace level that are in effect are output in that order, separated by a slash (/). If SQL tracing is disabled, a blank character is output.

An example of output when access path is in effect and the trace level is the SQL statement level is as follows:

```
accesspath / sql
```

■ **Values output when an event (adbclientdefmang) is successful**

When an event (adbclientdefmang) is successful, the client-managing definitions are output separated by slashes (/) in the command format explained in Table 12-14: Output format of server definition. If centralized management of client definitions is disabled, a blank character is output.

The following are a specification example and output example of client-managing definitions.

- **Specification example of client-managing definitions**

```
adbclientmang -f client01.def -i USER01, USER02
adbclientmang -f client02.def -i USER03
```

- **Output example**

```
adbclientmang -f client01.def -i USER01, USER02 / adbclientmang -f client02.def
-i USER03
```

■ **Values output when an event (adbmodarea) is successful**

When an event (adbmodarea) is successful, the values are output in the following format:

```
AREA NAME = DB-area-name, FILE NUMBER = number-of-DB-area-files
```

Description

DB-area-name

The DB area name. Either the specified value or a normalized character string is output.

number-of-DB-area-files

The number of DB area files. If the number of files is unknown, ? is output. If the DB area does not exist, 0 is output.

The following are a specification example and output example for a DB area addition and modification option file.

- **Specification example of DB area addition and modification option file**

```
adbaddarea -n ADBUTBL02
```

- **Output example (BEFORE_SYSTEM_INFO column)**

```
AREA NAME = ADBUTBL02, FILE NUMBER = 0
```

- **Output example (AFTER_SYSTEM_INFO)**

```
AREA NAME = ADBUTBL02, FILE NUMBER = 1
```

■ **Values output when an event (adbmodbuff) is successful**

When an event (adbmodbuff) is successful, the number of pages in the buffer for the local work table (5 to 100,000,000) changed by the adbmodbuff command is output.

The following are a specification example and output example for a buffer-modifying option file of the adbmodbuff command.

- **Specification example of buffer-modifying option file**

```
set adb_dbbuff_wrktbl_clt_blk_num = 256
```

- **Output example**

```
256
```

■ Values output when an event (adbcolumnize) is successful

When an event (adbcolumnize) is successful, the information about the status of the updated-row columnizing facility is output.

- If the updated-row columnizing facility is enabled, ACTIVE is output.
- If the updated-row columnizing facility is disabled, INACTIVE is output.

The following table shows the circumstances under which each *object type* is output in the OBJECT_TYPE column in Table 12-10: Column structure of table function derived table when retrieving audit trails:

Table 12-19: Circumstances in which object types are output in audit trails

No.	Object type	Output trigger
1	INDEX#1	• Defining and deleting indexes • Rebuilding indexes (adbidxrebuild command)
2	SCHEMA	Defining and deleting schema
3	TABLE	• Defining a base table • Changing the definition of a base table • Referencing a base table (including data export) • Updating a base table (including data import) • Deleting a base table • Granting or revoking an access privilege • Collecting cost information (adbgetcst command) • Merging chunks (adbmergechunk command) • Setting, changing, and deleting chunk comments (adbchgchunkcomment command) • Change the chunk status (adbchgchunkstatus command) • Archiving chunks (adbarchivechunk command) • Unarchiving chunks (adbunarchivechunk command) • Reorganizing system tables (base tables) (adbreorgsystemdata command)
4	TABLE FUNCTION#2	Referencing a table function derived table (including data export)
5	VIEW	• Defining a viewed table • Re-creating a viewed table • Referencing a viewed table (including data export) • Updating a viewed table • Deleting a viewed table • Granting or revoking an access privilege

#1

Information about the indexes used to retrieve data from a table is not output.

#2

The object identifier is the function name of the table function. Arguments of the table function are not output.

The following table shows the *number of rows output by each operation* in the ACCESS_COUNT column in Table 12-10: Column structure of table function derived table when retrieving audit trails.

Table 12-20: Number of rows output in ACCESS_COUNT column

No.	Event	EVENT_SUBTYPE of audit trail output by operation	Number of output rows
1	SELECT statement	SELECT	Number of rows in search results#1, #2, #3
2	INSERT statement	INSERT	Number of rows#3 inserted into the table output in the OBJECT_NAME column in Table 12-10: Column structure of table function derived table when retrieving audit trails
3		SELECT	NULL#2
4	UPDATE statement	UPDATE	Number of rows#3 updated in the table output in the OBJECT_NAME column in Table 12-10: Column structure of table function derived table when retrieving audit trails
5		SELECT	NULL#2
6	DELETE statement	DELETE	Number of rows#3 deleted from the table output in the OBJECT_NAME column in Table 12-10: Column structure of table function derived table when retrieving audit trails
7		SELECT	NULL#2
8	TRUNCATE TABLE statement	TRUNCATE TABLE	0
9	PURGE CHUNK statement	PURGE CHUNK	0
10		SELECT	NULL#2
11	Definition SQL statements	Subtype of each event	0#4
12	adbimport command	ADBIMPORT	Number of data rows#3 imported into the table output in the OBJECT_NAME column in Table 12-10: Column structure of table function derived table when retrieving audit trails
13	adbexport command	ADBEXPORT	Number of exported data rows#2, #3
14	All other	Subtype of each event	NULL

#1

The output value is the number of rows for which FETCH processing had completed on the HADB server side when the cursor was closed. Due to batch transfer of search results, the output row count might be a number determined by the following formula instead of the number of rows actually undergoing FETCH processing on the client. In this case, the value will be larger than the number of rows undergoing FETCH processing on the HADB client side.

```
number-of-batch-transferred-rows × 2 - 1
```

#2

When executing one SQL statement or one command, if multiple audit trails are output for which the content of the EVENT_SUBTYPE column in Table 12-10: Column structure of table function derived table when retrieving audit trails is the same, the same value will be output in the ACCESS_COUNT column for these audit trails.

#3

If the value of the EVENT_RESULT column in Table 12-10: Column structure of table function derived table when retrieving audit trails is FAILURE, 0 is output.

#4

If multiple audit trails are output when one definition SQL statement is executed, the same value (0) is output in the ACCESS_COUNT column for these audit trails.

For details about the circumstances in which an audit trail is output for each event and a description of the output items, see 12.9.3 Audit trail output triggers and output items.

12.9.3 Audit trail output triggers and output items

This section explains the circumstances in which audit trails are output, and the items output in each audit trail.

The following table lists the output triggers and output items for audit trails. Because the Column structure of table function derived table for audit trail column has a large number of sub-columns, these are split across five tables.

- Table 12-21: Audit trail output triggers and output items (1 of 5)
- Table 12-22: Audit trail output triggers and output items (2 of 5)
- Table 12-23: Audit trail output triggers and output items (3 of 5)
- Table 12-24: Audit trail output triggers and output items (4 of 5)
- Table 12-25: Audit trail output triggers and output items (5 of 5)

For details about the contents of the Audit event column in each of these five tables, see Table 12-9: List of audit target events and output items in 12.9.1 List of audit target events and output items.

For details about the contents of the Column structure of table function derived table for audit trail column in each of these five tables, see Table 12-10: Column structure of table function derived table when retrieving audit trails in 12.9.2 Column structure of table function derived table when retrieving audit trails.

The legend and annotations for all five tables are provided after Table 12-25: Audit trail output triggers and output items (5 of 5).

Table 12-21: Audit trail output triggers and output items (1 of 5)

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
1	Mandatory audit event	System event	Starting the HADB server (adbstart command)		Y	EVENT	Y		SYSTEM	ADBSTART
2			Stopping the HADB server (adbstop command)	Normal termination	Y	EVENT	Y		SYSTEM	ADBSTOP

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
3			Changing the HADB server operation mode (adbchgsrvmode command)	Normal termination	Y	EVENT	Y		SYSTEM	ADBCHGSRVMODE
4			Changing the node type (adbchgnodetype command)	Normal termination	Y	EVENT	Y		SYSTEM	ADBCHGNODETYPE
5			Starting and stopping output of SQL trace information (adbchgsqltrc command)	Normal termination	Y	EVENT	Y		SYSTEM	ADBCHGSQLTRC
6			Centrally managing client definitions (adbclientdefmang command with --update option specified)	Normal termination[#2]	Y	EVENT	Y	Y	SYSTEM	ADBCLIENTDEFMANG
7			Centrally managing client definitions (adbclientdefmang command with -i option specified)	Normal termination[#2]	Y	EVENT	Y	Y	SYSTEM	ADBCLIENTDEFMANG
8			Adding and modifying DB areas (adbmodarea command)	Normal termination[#2]	Y	EVENT	Y		SYSTEM	ADBMODAREA
9			Changing a buffer (adbmodbuff command)	Normal termination[#2]	Y	EVENT	Y		SYSTEM	ADBMODBUFF
10			Managing the updated-row columnizing facility (adbcolumnize command)	Normal termination	Y	EVENT	Y		SYSTEM	ADBCOLUMNIZE
11		Audit event	Granting audit admin privilege (GRANT statement with	Normal termination[#6, #7]	Y	EVENT	Y	Y	AUDIT	GRANT

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			AUDIT ADMIN specified)							
12			Granting audit viewer privilege (GRANT statement with AUDIT VIEWER specified)	Normal termination[#6, #7]	Y	EVENT	Y	Y	AUDIT	GRANT
13			Revoking audit admin privilege (REVOKE statement with AUDIT ADMIN specified)	Normal termination[#6, #7]	Y	EVENT	Y	Y	AUDIT	REVOKE
14			Revoking audit viewer privilege (REVOKE statement with AUDIT VIEWER specified)	Normal termination[#6, #7]	Y	EVENT	Y	Y	AUDIT	REVOKE
15			Defining audit targets (CREATE AUDIT statement)	Normal termination	Y	EVENT	Y	Y	AUDIT	CREATE AUDIT
16			Deleting audit target definition information (DROP AUDIT statement)	Normal termination	Y	EVENT	Y	Y	AUDIT	DROP AUDIT
17			Changing auditor passwords (ALTER USER statement)	Normal termination	Y	EVENT	Y	Y	AUDIT	ALTER USER
18			Enabling the audit trail facility (adbauditrail command with --start option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBAUDITTRAIL START
19			Disabling the audit trail facility (adbauditrail command with --stop option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBAUDITTRAIL STOP
20			Swapping the audit trail file (adbauditra	Normal termination[#2, #9]	Y	EVENT	Y	Y	AUDIT	ADBAUDITTRAIL SWAP

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			`il` command with `--swap` option specified)							
21			Referencing information related to the audit trail facility (`adbaudittrail` command with `-d` option specified)	Normal termination[#2, #9]	Y	EVENT	Y	Y	AUDIT	ADBAUDITTRAIL DISPLAY
22			Executing a system-defined function for audit trails (ADB_AUDITREAD function) (data manipulation SQL)	Normal termination	Y	EVENT	Y	Y	AUDIT	SELECT
23			Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function (data manipulation SQL)	Normal termination	Y	EVENT	Y	Y	AUDIT	SELECT
24			Executing a system-defined function for audit trails (ADB_AUDITREAD function) (`adbexport` command with `-q` option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBEXPORT
25			Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function (`adbexport` command with `-q` option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBEXPORT

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
26			Retrieving data from SQL_AUDITS dictionary table (data manipulation SQL)	Normal termination	Y	EVENT	Y	Y	AUDIT	SELECT
27			Retrieving data from viewed tables that depend on the SQL_AUDITS dictionary table (data manipulation SQL)	Normal termination	Y	EVENT	Y	Y	AUDIT	SELECT
28			Retrieving data from SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBEXPORT
29			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBEXPORT
30			Retrieving data from SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBEXPORT
31			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]	Y	EVENT	Y	Y	AUDIT	ADBEXPORT

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
32			Converting audit trail files (adbconverta udittrailfil e command)	Normal termination[#2, #9]	Y	EVENT	Y	Y	AUDIT	ADBCONVERTAUDITTRAILFILE
33	Optional audit event	Session event	Connecting to the HADB server	Normal termination	Y	EVENT	Y	Y	SESSION	CONNECT
34			Disconnecting from the HADB server	Normal termination[#1]	Y	EVENT	Y	Y	SESSION	DISCONNECT
35		Privilege management event	Granting the DBA privilege (GRANT statement with DBA specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
36			Granting the CONNECT privilege (GRANT statement with CONNECT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
37			Granting the SCHEMA privilege (GRANT statement with SCHEMA specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
38			Granting the SELECT privilege (GRANT statement with SELECT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
39			Granting the INSERT privilege (GRANT statement with INSERT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
40			Granting the UPDATE privilege (GRANT statement with UPDATE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
41			Granting the DELETE privilege (GRANT statement with DELETE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
42			Granting the TRUNCATE privilege (GRANT statement with TRUNCATE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
43			Granting the REFERENCES privilege (GRANT statement with REFERENCES specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
44			Granting the IMPORT TABLE privilege (GRANT statement with IMPORT TABLE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
45			Granting the REBUILD INDEX privilege (GRANT statement with REBUILD INDEX specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
46			Granting the GET COSTINFO privilege (GRANT statement with GET COSTINFO specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
47			Granting the EXPORT TABLE privilege (GRANT statement with EXPORT TABLE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
48			Granting the MERGE CHUNK privilege (GRANT statement with	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			MERGE CHUNK specified)							
49			Granting the CHANGE CHUNK COMMENT privilege (GRANT statement with CHANGE CHUNK COMMENT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
50			Granting the CHANGE CHUNK STATUS privilege (GRANT statement with CHANGE CHUNK STATUS specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
51			Granting the ARCHIVE CHUNK privilege (GRANT statement with ARCHIVE CHUNK specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
52			Granting the UNARCHIVE CHUNK privilege (GRANT statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	GRANT
53			Revoking the DBA privilege (REVOKE statement with DBA specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
54			Revoking the CONNECT privilege (REVOKE statement with CONNECT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
55			Revoking the SCHEMA privilege (REVOKE	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			statement with SCHEMA specified)							
56			Revoking the SELECT privilege (REVOKE statement with SELECT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
57			Revoking the INSERT privilege (REVOKE statement with INSERT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
58			Revoking the UPDATE privilege (REVOKE statement with UPDATE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
59			Revoking the DELETE privilege (REVOKE statement with DELETE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
60			Revoking the TRUNCATE privilege (REVOKE statement with TRUNCATE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
61			Revoking the REFERENCES privilege (REVOKE statement with REFERENCES specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
62			Revoking the IMPORT TABLE privilege (REVOKE	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			statement with IMPORT TABLE specified)							
63			Revoking the REBUILD INDEX privilege (REVOKE statement with REBUILD INDEX specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
64			Revoking the GET COSTINFO privilege (REVOKE statement with GET COSTINFO specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
65			Revoking the EXPORT TABLE privilege (REVOKE statement with EXPORT TABLE specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
66			Revoking the MERGE CHUNK privilege (REVOKE statement with MERGE CHUNK specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
67			Revoking the CHANGE CHUNK COMMENT privilege (REVOKE statement with CHANGE CHUNK COMMENT specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
68			Revoking the CHANGE CHUNK STATUS privilege (REVOKE statement with CHANGE CHUNK STATUS specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
69			Revoking the ARCHIVE CHUNK privilege (REVOKE statement with ARCHIVE CHUNK specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
70			Revoking the UNARCHIVE CHUNK privilege (REVOKE statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	Y	EVENT	Y	Y	PRIVILEGE	REVOKE
71			Creating an HADB user (CREATE USER statement)	Normal termination	Y	EVENT	Y	Y	PRIVILEGE	CREATE USER
72			Deleting an HADB user (DROP USER statement)	Normal termination	Y	EVENT	Y	Y	PRIVILEGE	DROP USER
73			Changing user information for an HADB user (ALTER USER statement)	Normal termination	Y	EVENT	Y	Y	PRIVILEGE	ALTER USER
74		Definition SQL event	Defining an index (CREATE INDEX statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	CREATE INDEX
75			Defining a schema (CREATE SCHEMA statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	CREATE SCHEMA
76			Defining a base table (CREATE TABLE statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	CREATE TABLE
77			Defining a viewed table (CREATE VIEW statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	CREATE VIEW
78			Deleting an index (DROP INDEX statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	DROP INDEX
79			Deleting a schema (DROP	Normal termination	Y	EVENT	Y	Y	DEFINE	DROP SCHEMA

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			SCHEMA statement)							
80			Deleting a base table (DROP TABLE statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	DROP TABLE
81			Deleting a viewed table (DROP VIEW statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	DROP VIEW
82			Changing the definition of a base table (ALTER TABLE statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	ALTER TABLE
83			Changing the definition of a viewed table (ALTER VIEW statement)	Normal termination	Y	EVENT	Y	Y	DEFINE	ALTER VIEW
84		Data manipulation SQL event	Retrieving data from tables (SELECT statement)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	SELECT
85			Inserting rows into tables (INSERT statement[#3])	Normal termination	Y	EVENT	Y	Y	MANIPULATE	INSERT
86			Updating table row data (UPDATE statement[#3])	Normal termination	Y	EVENT	Y	Y	MANIPULATE	UPDATE
87			Deleting table row data (DELETE statement[#3])	Normal termination	Y	EVENT	Y	Y	MANIPULATE	DELETE
88			Deleting all row data from a table (TRUNCATE TABLE statement)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	TRUNCATE TABLE
89			Deleting all row data in a chunk (PURGE CHUNK statement[#3])	Normal termination	Y	EVENT	Y	Y	MANIPULATE	PURGE CHUNK
90			SQL parsing error		Y	EVENT	Y	Y	MANIPULATE	UNKNOWN
91			Acquiring data stored in chunks (#GETDATA	Normal termination	Y	EVENT	Y	Y	MANIPULATE	GETDATA

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			subcommand of adbsql command)							
92			Acquiring the number of data items stored in a chunk (#GETCOUNT subcommand of adbsql command)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	GETCOUNT
93			Displaying table information (#TABLES subcommand of adbsql command)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	TABLES
94			Displaying column information (#COLUMNS subcommand of adbsql command)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	COLUMNS
95			Displaying index information (#INDEXES subcommand of adbsql command)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	INDEXES
96			Displaying chunk information (#CHUNKS subcommand of adbsql command)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	CHUNKS
97			Displaying authorization identifiers (#GETUSER subcommand of adbsql command)	Normal termination	Y	EVENT	Y	Y	MANIPULATE	GETUSER
98		Command operation event	Importing data (adbimport command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBIMPORT

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
99			Exporting data (adbexport command without -q option specified)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBEXPORT
100			Exporting data (adbexport command with -q option specified)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBEXPORT
101			Rebuilding indexes (adbidxrebuild command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBIDXREBUILD
102			Collecting cost information (adbgetcst command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBGETCST
103			Analyzing DB status (adbdbstatus command)	Normal termination[#2]	Y	EVENT	Y		COMMAND	ADBDBSTATUS
104			Merging chunks (adbmergechunk command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBMERGECHUNK
105			Setting, changing, and deleting chunk comments (adbchgchunkcomment command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBCHGCHUNKCOMMENT
106			Changing the chunk status (adbchgchunkstatus command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBCHGCHUNKSTATUS
107			Archiving chunks (adbarchivechunk command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBARCHIVECHUNK
108			Unarchiving chunks (adbunarchivechunk command)	Normal termination[#2]	Y	EVENT	Y	Y	COMMAND	ADBUNARCHIVECHUNK
109			Reorganizing system tables (base tables)	Normal termination[#2]	Y	EVENT	Y		COMMAND	ADBREORGSYSTEMDATA

Audit event					Column structure of table function derived table for audit trail (1 of 5)					
No.	Event category	Event type	Event Description		HADB version (HADB_VERSION)	Audit trail type (AUDIT_TRAIL_TYPE)	Event execution time (EXEC_TIME)	Event execution user (USER_NAME)	Event type (EVENT_TYPE)	Event subtype (EVENT_SUBTYPE)
			(adbreorgsystemdata command)							
110			Registering and deleting synonym dictionaries (adbsyndict command)	Normal termination[#2]	Y	EVENT	Y		COMMAND	ADBSYNDICT

Table 12-22: Audit trail output triggers and output items (2 of 5)

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
1	Mandatory audit event	System event	Starting the HADB server (adbstart command)		SUCCESS		Y				Y	Y
2			Stopping the HADB server (adbstop command)	Normal termination	SUCCESS		Y		F	F	Y	Y
3			Changing the HADB server operation mode (adbchgsrvmode command)	Normal termination	SUCCESS		Y				Y	Y
4			Changing the node type (adbchgnodetype command)	Normal termination	SUCCESS		Y				Y	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
5			Starting and stopping output of SQL trace information (`adbchgsqltrc` command)	Normal termination	`SUCCESS`		Y				Y	Y
6			Centrally managing client definitions (`adbclientdefmang` command with `--update` option specified)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
7			Centrally managing client definitions (`adbclientdefmang` command with `-i` option specified)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
8			Adding and modifying DB areas (`adbmodarea` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
9			Changing a buffer (`adbmodbuff` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
10			Managing the updated-row columnizing facility (`adbcolumnize` command)	Normal termination	`SUCCESS`		Y				Y	Y
11		Audit event	Granting audit admin privilege (`GRANT` statement with `AUDIT ADMIN` specified)	Normal termination[#6, #7]	`SUCCESS`		Y	Y	Y	A	B	Y
12			Granting audit viewer privilege (`GRANT` statement with `AUDIT VIEWER` specified)	Normal termination[#6, #7]	`SUCCESS`		Y	Y	Y	A	B	Y
13			Revoking audit admin privilege (`REVOKE` statement with `AUDIT ADMIN` specified)	Normal termination[#6, #7]	`SUCCESS`		Y	Y	Y	A	B	Y
14			Revoking audit viewer privilege (`REVOKE` statement with `AUDIT`	Normal termination[#6, #7]	`SUCCESS`		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			VIEWER specified)									
15			Defining audit targets (CREATE AUDIT statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
16			Deleting audit target definition information (DROP AUDIT statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
17			Changing auditor passwords (ALTER USER statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
18			Enabling the audit trail facility (adbaudittrail command with --start option specified)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
19			Disabling the audit trail facility (adbaudittrail command with --stop option specified)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
20			Swapping the audit trail file	Normal termina	SUCCESS		Y	Y	F	F	Y	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			(adbaudittrail command with --swap option specified)	tion[#2, #9]								
21			Referencing information related to the audit trail facility (adbaudittrail command with -d option specified)	Normal termination[#2, #9]	SUCCESS		Y	Y	F	F	Y	Y
22			Executing a system-defined function for audit trails (ADB_AUDITREAD function) (data manipulation SQL)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
23			Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function (data manipulation SQL)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
24			Executing a system-defined function for audit trails (ADB_AUDITREAD function) (adbexport command with -q option specified)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
25			Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function (adbexport command with -q option specified)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
26			Retrieving data from SQL_AUDITS dictionary table (data manipulation SQL)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
27			Retrieving data from viewed tables that depend on the SQL_AUDITS	Normal termination	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			dictionary table (data manipulation SQL)									
28			Retrieving data from SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
29			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
30			Retrieving data from SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
31			Retrieving data from viewed table that depends on	Normal termination[#2]	SUCCESS		Y	Y			Y	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			SQL_AUDITS dictionary table (adbexport command with -q option specified)									
32			Converting audit trail files (adbconvertaudittrailfile command)	Normal termination[#2, #9]	SUCCESS		Y	Y			Y	Y
33	Optional audit event	Session event	Connecting to the HADB server	Normal termination	SUCCESS		Y	Y	D	E	B	Y
34			Disconnecting from the HADB server	Normal termination[#1]	SUCCESS		Y	Y	D	E	B	Y
35		Privilege management event	Granting the DBA privilege (GRANT statement with DBA specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
36			Granting the CONNECT privilege (GRANT statement with CONNECT specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
37			Granting the SCHEMA privilege	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			(GRANT statement with SCHEMA specified)									
38			Granting the SELECT privilege (GRANT statement with SELECT specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
39			Granting the INSERT privilege (GRANT statement with INSERT specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
40			Granting the UPDATE privilege (GRANT statement with UPDATE specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
41			Granting the DELETE privilege (GRANT statement with DELETE specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
42			Granting the TRUNCATE privilege (GRANT statement	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			with TRUNCATE specified)									
43			Granting the REFERENCES privilege (GRANT statement with REFERENCES specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
44			Granting the IMPORT TABLE privilege (GRANT statement with IMPORT TABLE specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
45			Granting the REBUILD INDEX privilege (GRANT statement with REBUILD INDEX specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
46			Granting the GETCOSTINFO privilege (GRANT statement with GETCOSTINFO specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
47			Granting the EXPORT TABLE privilege (GRANT statement with EXPORT TABLE specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
48			Granting the MERGE CHUNK privilege (GRANT statement with MERGE CHUNK specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
49			Granting the CHANGE CHUNK COMMENT privilege (GRANT statement with CHANGE CHUNK COMMENT specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
50			Granting the CHANGE CHUNK STATUS privilege (GRANT statement with CHANGE CHUNK STATUS specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
51			Granting the ARCHIVE CHUNK privilege (GRANT statement with ARCHIVE CHUNK specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
52			Granting the UNARCHIVE CHUNK privilege (GRANT statement with UNARCHIVE CHUNK specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
53			Revoking the DBA privilege (REVOKE statement with DBA specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
54			Revoking the CONNECT privilege (REVOKE statement with CONNECT specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
55			Revoking the SCHEMA privilege (REVOKE statement with	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			SCHEMA specified)									
56			Revoking the SELECT privilege (REVOKE statement with SELECT specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
57			Revoking the INSERT privilege (REVOKE statement with INSERT specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
58			Revoking the UPDATE privilege (REVOKE statement with UPDATE specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
59			Revoking the DELETE privilege (REVOKE statement with DELETE specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
60			Revoking the TRUNCATE privilege (REVOKE statement with TRUNCAT	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			E specified)									
61			Revoking the REFERENCES privilege (REVOKE statement with REFERENCES specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
62			Revoking the IMPORT TABLE privilege (REVOKE statement with IMPORT TABLE specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
63			Revoking the REBUILD INDEX privilege (REVOKE statement with REBUILD INDEX specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
64			Revoking the GET COSTINFO privilege (REVOKE statement with GET COSTINFO specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
65			Revoking the EXPORT TABLE privilege (REVOKE statement with EXPORT TABLE specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
66			Revoking the MERGE CHUNK privilege (REVOKE statement with MERGE CHUNK specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
67			Revoking the CHANGE CHUNK COMMENT privilege (REVOKE statement with CHANGE CHUNK COMMENT specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y
68			Revoking the CHANGE CHUNK STATUS privilege (REVOKE statement with CHANGE CHUNK STATUS specified)	Normal termination#6	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
69			Revoking the ARCHIVE CHUNK privilege (REVOKE statement with ARCHIVE CHUNK specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
70			Revoking the UNARCHIVE CHUNK privilege (REVOKE statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	SUCCESS		Y	Y	Y	A	B	Y
71			Creating an HADB user (CREATE USER statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
72			Deleting an HADB user (DROP USER statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
73			Changing user information for an HADB user (ALTER USER statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
74		Definition SQL event	Defining an index (CREATE INDEX statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
75			Defining a schema (CREATE SCHEMA statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
76			Defining a base table (CREATE TABLE statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
77			Defining a viewed table (CREATE VIEW statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
78			Deleting an index (DROP INDEX statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
79			Deleting a schema (DROP SCHEMA statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
80			Deleting a base table (DROP TABLE statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
81			Deleting a viewed table (DROP VIEW statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
82			Changing the definition	Normal termination	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			of a base table (ALTER TABLE statement)									
83			Changing the definition of a viewed table (ALTER VIEW statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
84		Data manipulation SQL event	Retrieving data from tables (SELECT statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
85			Inserting rows into tables (INSERT statement[#3])	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
86			Updating table row data (UPDATE statement[#3])	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
87			Deleting table row data (DELETE statement[#3])	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
88			Deleting all row data from a table (TRUNCATE TABLE statement)	Normal termination	SUCCESS		Y	Y	Y	A	B	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
89			Deleting all row data in a chunk (PURGE CHUNK statement[#3])	Normal termination	SUCCESS		Y	Y	Y	A	B	Y
90			SQL parsing error		FAILURE		Y	Y	Y	A	B	Y
91			Acquiring data stored in chunks (#GETDATA subcommand of adbsql command)	Normal termination	SUCCESS		Y	Y	Y	A	Y	Y
92			Acquiring the number of data items stored in a chunk (#GETCOUNT subcommand of adbsql command)	Normal termination	SUCCESS		Y	Y	Y	A	Y	Y
93			Displaying table information (#TABLES subcommand of adbsql command)	Normal termination	SUCCESS		Y	Y	Y	A	Y	Y
94			Displaying column information (#COLUMNS	Normal termination	SUCCESS		Y	Y	Y	A	Y	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			subcommand of `adbsql` command)									
95			Displaying index information (`#INDEXES` subcommand of `adbsql` command)	Normal termination	`SUCCESS`		Y	Y	Y	A	Y	Y
96			Displaying chunk information (`#CHUNKS` subcommand of `adbsql` command)	Normal termination	`SUCCESS`		Y	Y	Y	A	Y	Y
97			Displaying authorization identifiers (`#GETUSER` subcommand of `adbsql` command)	Normal termination	`SUCCESS`		Y	Y	Y	A	Y	Y
98		Command operation event	Importing data (`adbimport` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
99			Exporting data (`adbexport` command without `-q` option specified)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
100			Exporting data (`adbexport` command with `-q` option specified)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
101			Rebuilding indexes (`adbidxrebuild` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
102			Collecting cost information (`adbgetcst` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
103			Analyzing DB status (`adbdbstatus` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
104			Merging chunks (`adbmergechunk` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
105			Setting, changing, and deleting chunk comments (`adbchgchunkcomment` command)	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y
106			Changing the chunk status (`adbchgchunkst`	Normal termination[#2]	`SUCCESS`		Y	Y			Y	Y

Audit event					Column structure of table function derived table for audit trail (2 of 5)							
No.	Event category	Event type	Event description		Event result (EVENT_RESULT)	Confirmed privilege (USED_PRIVILEGE)	OS account name on client side. (OS_USER_NAME)	Application identifier (AP_NAME)	Client IP address (CLIENT_IP_ADDRESS)	Client port number (CLIENT_PORT_NUMBER)	Client process name (CLIENT_PROCESS_NAME)	Client process ID (CLIENT_PROCESS_ID)
			atus command)									
107			Archiving chunks (adbarchivechunk command)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
108			Unarchiving chunks (adbunarchivechunk command)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
109			Reorganizing system tables (base tables) (adbreorgsystemdata command)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y
110			Registering and deleting synonym dictionaries (adbsyndict command)	Normal termination[#2]	SUCCESS		Y	Y			Y	Y

Table 12-23: Audit trail output triggers and output items (3 of 5)

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
1	Mandatory audit event	System event	Starting the HADB server (`adbstart` command)		Y	Y	Y	Y	Y	Y				
2			Stopping the HADB server (`adbstop` command)	Normal termination	Y	Y	Y	Y	Y	Y				
3			Changing the HADB server operation mode (`adbchgsrvmode` command)	Normal termination	Y	Y	Y	Y	Y	Y				
4			Changing the node type (`adbchgnodetype` command)	Normal termination	Y	Y	Y	Y	Y	Y				
5			Starting and stopping output of SQL trace information (`adbchgsqltrc`	Normal termination	Y	Y	Y	Y	Y	Y				

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			command)											
6			Centrally managing client definitions (`adbclientdefmang` command with `--update` option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
7			Centrally managing client definitions (`adbclientdefmang` command with `-i` option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
8			Adding and modifying DB areas (`adbmodarea` command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
9			Changing a buffer (adbmodbuff command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
10			Managing the updated-row columnizing facility (adbcolumnize command)	Normal termination	Y	Y	Y	Y	Y	Y				
11		Audit event	Granting audit admin privilege (GRANT statement with AUDIT ADMIN specified)	Normal termination[#6, #7]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
12			Granting audit viewer privilege (GRANT statement with AUDIT VIEWER specified)	Normal termination[#6, #7]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
13			Revoking audit admin privilege (REVOKE statement with AUDIT ADMIN specified)	Normal termination[#6, #7]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
14			Revoking audit viewer privilege (REVOKE statement with AUDIT VIEWER specified)	Normal termination[#6, #7]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
15			Defining audit targets (CREATE AUDIT statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
16			Deleting audit target definition information (DROP AUDIT statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
17			Changing auditor passwords (`ALTER USER` statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
18			Enabling the audit trail facility (`adbaudittrail` command with `--start` option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
19			Disabling the audit trail facility (`adbaudittrail` command with `--stop` option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
20			Swapping the audit trail file (`adbaudittrail`	Normal termination[#2, #9]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			command with `--swap` option specified)											
21			Referencing information related to the audit trail facility (`adbauditttrail` command with `-d` option specified)	Normal termination[#2, #9]	Y	Y	Y	Y	Y	Y	Y	Y		
22			Executing a system-defined function for audit trails (`ADB_AUDITREAD` function) (data manipulation SQL)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
23			Retrieving data from a viewed table that	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			depends on a derived table that specifies an ADB_AUDITREAD function (data manipulation SQL)											
24			Executing a system-defined function for audit trails (ADB_AUDITREAD function) (adbexport command with -q option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
25			Retrieving data from a viewed table that depends on a derived table that specifie	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			s an ADB_AUDITREAD function (adbexport command with -q option specified)											
26			Retrieving data from SQL_AUDITS dictionary table (data manipulation SQL)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
27			Retrieving data from viewed tables that depend on the SQL_AUDITS dictionary table (data manipulation SQL)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
28			Retrieving data from SQL_AUDITS dictiona	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			ry table (adbexport command without -q option specified)											
29			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
30			Retrieving data from SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
31			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
32			Converting audit trail files (adbconvertaudittrailfile command)	Normal termination[#2, #9]	Y	Y	Y	Y	Y	Y	Y	Y		
33	Optional audit event	Session event	Connecting to the HADB server	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		
34			Disconnecting from the HADB server	Normal termination[#1]	Y	Y	Y	Y	Y	Y	Y	Y		
35		Privilege ma	Granting the DBA privileg	Normal termi	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
		nagement event	e (GRANT statement with DBA specified)	nation[#6]										
36			Granting the CONNECT privilege (GRANT statement with CONNECT specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
37			Granting the SCHEMA privilege (GRANT statement with SCHEMA specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
38			Granting the SELECT privilege (GRANT statement with SELEC	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			T specified)											
39			Granting the INSERT privilege (GRANT statement with INSERT specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
40			Granting the UPDATE privilege (GRANT statement with UPDATE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
41			Granting the DELETE privilege (GRANT statement with DELETE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
42			Granting the TRUNCATE privilege (GRANT statement with TRUNCATE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
43			Granting the REFERENCES privilege (GRANT statement with REFERENCES specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
44			Granting the IMPORT TABLE privilege (GRANT statement with IMPORT TABLE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
45			Granting the REBUI	Normal termi	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			LD INDEX privilege (GRANT statement with REBUILD INDEX specified)	nation[#6]										
46			Granting the GET COSTINFO privilege (GRANT statement with GET COSTINFO specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
47			Granting the EXPORT TABLE privilege (GRANT statement with EXPORT TABLE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
48			Granting the	Normal	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			MERGE CHUNK privilege (GRANT statement with MERGE CHUNK specified)	termination[#6]										
49			Granting the CHANGE CHUNK COMMENT privilege (GRANT statement with CHANGE CHUNK COMMENT specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
50			Granting the CHANGE CHUNK STATUS privilege (GRANT statement with CHANGE CHUNK	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			STATUS specified)											
51			Granting the ARCHIVE CHUNK privilege (GRANT statement with ARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
52			Granting the UNARCHIVE CHUNK privilege (GRANT statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
53			Revoking the DBA privilege (REVOKE statement with DBA	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			specified)											
54			Revoking the CONNECT privilege (REVOKE statement with CONNECT specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
55			Revoking the SCHEMA privilege (REVOKE statement with SCHEMA specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
56			Revoking the SELECT privilege (REVOKE statement with SELECT specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
57			Revoking the	Normal	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			INSERT privilege (REVOKE statement with INSERT specified)	termination[#6]										
58			Revoking the UPDATE privilege (REVOKE statement with UPDATE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
59			Revoking the DELETE privilege (REVOKE statement with DELETE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
60			Revoking the TRUNCATE privilege (REVO	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			KE statement with TRUNCATE specified)											
61			Revoking the REFERENCES privilege (REVOKE statement with REFERENCES specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
62			Revoking the IMPORT TABLE privilege (REVOKE statement with IMPORT TABLE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
63			Revoking the REBUILD INDEX privilege (REVOKE stateme	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			nt with REBUILD INDEX specified)											
64			Revoking the GETCOSTINFO privilege (REVOKE statement with GETCOSTINFO specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
65			Revoking the EXPORTTABLE privilege (REVOKE statement with EXPORTTABLE specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
66			Revoking the MERGECHUNK privilege (REVOKE stateme	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			nt with MERGE CHUNK specified)											
67			Revoking the CHANGE CHUNK COMMENT privilege (REVOKE statement with CHANGE CHUNK COMMENT specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
68			Revoking the CHANGE CHUNK STATUS privilege (REVOKE statement with CHANGE CHUNK STATUS specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
69			Revoking the	Normal	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			ARCHIVE CHUNK privilege (REVOKE statement with ARCHIVE CHUNK specified)	termination[#6]										
70			Revoking the UNARCHIVE CHUNK privilege (REVOKE statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
71			Creating an HADB user (CREATE USER statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
72			Deleting an HADB user (DROP USER statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
73			Changing user information for an HADB user (ALTER USER statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
74		Definition SQL event	Defining an index (CREATE INDEX statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
75			Defining a schema (CREATE SCHEMA statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
76			Defining a base table (CREATE TABLE statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
77			Defining a viewed table (CREATE VIEW statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
78			Deleting an index (DROP INDEX statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
79			Deleting a schema (DROP SCHEMA statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
80			Deleting a base table (DROP TABLE statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
81			Deleting a viewed table (DROP VIEW statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
82			Changing the definition of a base table (ALTER TABLE statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
83			Changing the definition of a	Normal termi	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			viewed table (ALTER VIEW statement)	nation										
84		Data manipulation SQL event	Retrieving data from tables (SELECT statement)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
85			Inserting rows into tables (INSERT statement[#3])	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
86			Updating table row data (UPDATE statement[#3])	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
87			Deleting table row data (DELETE statement[#3])	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
88			Deleting all row data from a table	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			(TRUNCATE TABLE statement)											
89			Deleting all row data in a chunk (PURGE CHUNK statement[#3])	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
90			SQL parsing error		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
91			Acquiring data stored in chunks (#GETDATA subcommand of adbsql command)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		
92			Acquiring the number of data items stored in a chunk (#GETCOUNT subcommand of adbsql command)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
93			Displaying table information (#TABLES subcommand of adbsql command)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		
94			Displaying column information (#COLUMNS subcommand of adbsql command)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		
95			Displaying index information (#INDEXES subcommand of adbsql command)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		
96			Displaying chunk information (#CHUNKS subcom	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			mand of `adbsql` command)											
97			Displaying authorization identifiers (`#GETUSER` subcommand of `adbsql` command)	Normal termination	Y	Y	Y	Y	Y	Y	Y	Y		
98		Command operation event	Importing data (`adbimport` command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
99			Exporting data (`adbexport` command without `-q` option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
100			Exporting data (`adbexport` command with `-q` option specified)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
101			Rebuilding indexes (adbidxrebuild command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
102			Collecting cost information (adbgetcst command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
103			Analyzing DB status (adbdbstatus command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
104			Merging chunks (adbmergechunk command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
105			Setting, changing, and deleting chunk comments (adbchgchunkcomment command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event					Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description		HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
106			Changing the chunk status (adbchgchunkstatus command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
107			Archiving chunks (adbarchivechunk command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
108			Unarchiving chunks (adbunarchivechunk command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
109			Reorganizing system tables (base tables) (adbreorgsystemdata command)	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		
110			Registering and deleting synonym dictiona	Normal termination[#2]	Y	Y	Y	Y	Y	Y	Y	Y		

Audit event				Column structure of table function derived table for audit trail (3 of 5)									
No.	Event category	Event type	Event Description	HADB host name (HADB_HOST_NAME)	HADB IP address (HADB_IP_ADDRESS)	HADB port number (HADB_PORT_NUMBER)	HADB process ID (HADB_PROCESS_ID)	HADB server directory (ADBDIR)	HADB node number[#8] (HADB_NODE_NUMBER)	Connection ID (CONNECTION_ID)	Connection sequence number (CONNECTION_NUMBER)	Statement handle (STATEMENT_HANDLE)	SQL statement serial number (SQL_SERIAL_NUMBER)
			ries (adbsyndict command)										

Table 12-24: Audit trail output triggers and output items (4 of 5)

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
1	Mandatory audit event	System event	Starting the HADB server (adbstart command)		Y	Y					
2			Stopping the HADB server (adbstop command)	Normal termination	Y	Y					
3			Changing the HADB server operation mode (adbchgsrvmode command)	Normal termination	Y	Y					
4			Changing the node type (adbchgnodetype command)	Normal termination	Y	Y					
5			Starting and stopping output of SQL trace	Normal termination	Y	Y					

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			information (`adbchgsqltrc` command)								
6			Centrally managing client definitions (`adbclientdefmang` command with `--update` option specified)	Normal termination[#2]	Y	Y					
7			Centrally managing client definitions (`adbclientdefmang` command with `-i` option specified)	Normal termination[#2]	Y	Y					
8			Adding and modifying DB areas (`adbmodarea` command)	Normal termination[#2]	Y	Y					
9			Changing a buffer (`adbmodbuff` command)	Normal termination[#2]	Y	Y					
10			Managing the updated-row columnizing facility (`adbcolumnize` command)	Normal termination	Y	Y					
11		Audit event	Granting audit admin privilege (`GRANT`	Normal termination[#6, #7]	Y	Y					Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			statement with AUDIT ADMIN specified)								
12			Granting audit viewer privilege (GRANT statement with AUDIT VIEWER specified)	Normal termination[#6, #7]	Y	Y					Y
13			Revoking audit admin privilege (REVOKE statement with AUDIT ADMIN specified)	Normal termination[#6, #7]	Y	Y					Y
14			Revoking audit viewer privilege (REVOKE statement with AUDIT VIEWER specified)	Normal termination[#6, #7]	Y	Y					Y
15			Defining audit targets (CREATE AUDIT statement)	Normal termination	Y	Y					Y
16			Deleting audit target definition information (DROP AUDIT statement)	Normal termination	Y	Y					Y
17			Changing auditor passwords (ALTER USER statement)	Normal termination	Y	Y					Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
18			Enabling the audit trail facility (`adbaudittrail` command with `--start` option specified)	Normal termination[#2]	Y	Y					
19			Disabling the audit trail facility (`adbaudittrail` command with `--stop` option specified)	Normal termination[#2]	Y	Y					
20			Swapping the audit trail file (`adbaudittrail` command with `--swap` option specified)	Normal termination[#2, #9]	Y	Y					
21			Referencing information related to the audit trail facility (`adbaudittrail` command with `-d` option specified)	Normal termination[#2, #9]	Y	Y					
22			Executing a system-defined function for audit trails (`ADB_AUDITREAD` function) (data	Normal termination	Y	Y	`TABLE FUNCTION`	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			manipulation SQL)								
23			Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function (data manipulation SQL)	Normal termination	Y	Y	VIEW	Y	Y	Y	Y
24			Executing a system-defined function for audit trails (ADB_AUDITREAD function) (adbexport command with -q option specified)	Normal termination[#2]	Y	Y	TABLE FUNCTION	Y	Y	Y	Y
25			Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function (adbexport command with -q option specified)	Normal termination[#2]	Y	Y	VIEW	Y	Y	Y	Y
26			Retrieving data from SQL_AUDITS dictionary	Normal termination	Y	Y	VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			table (data manipulation SQL)								
27			Retrieving data from viewed tables that depend on the SQL_AUDITS dictionary table (data manipulation SQL)	Normal termination	Y	Y	VIEW	Y	Y	Y	Y
28			Retrieving data from SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]	Y	Y	VIEW	Y	Y	Y	Y
29			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]	Y	Y	VIEW	Y	Y	Y	Y
30			Retrieving data from SQL_AUDITS dictionary table (adbexport	Normal termination[#2]	Y	Y	VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			command with -q option specified)								
31			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]	Y	Y	VIEW	Y	Y	Y	Y
32			Converting audit trail files (adbconvertaudittrailfile command)	Normal termination[#2, #9]	Y	Y					
33	Optional audit event	Session event	Connecting to the HADB server	Normal termination	Y	Y					
34			Disconnecting from the HADB server	Normal termination[#1]	Y	Y					
35		Privilege management event	Granting the DBA privilege (GRANT statement with DBA specified)	Normal termination[#6]	Y	Y					Y
36			Granting the CONNECT privilege (GRANT statement with	Normal termination[#6]	Y	Y					Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			CONNECT specified)								
37			Granting the SCHEMA privilege (GRANT statement with SCHEMA specified)	Normal termination[#6]	Y	Y					Y
38			Granting the SELECT privilege (GRANT statement with SELECT specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
39			Granting the INSERT privilege (GRANT statement with INSERT specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
40			Granting the UPDATE privilege (GRANT statement with UPDATE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
41			Granting the DELETE privilege (GRANT statement with DELETE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
42			Granting the TRUNCATE privilege (GRANT statement with	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			TRUNCATE specified)								
43			Granting the REFERENCES privilege (GRANT statement with REFERENCES specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
44			Granting the IMPORT TABLE privilege (GRANT statement with IMPORT TABLE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
45			Granting the REBUILD INDEX privilege (GRANT statement with REBUILD INDEX specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
46			Granting the GET COSTINFO privilege (GRANT statement with GET COSTINFO specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
47			Granting the EXPORT TABLE privilege (GRANT statement with EXPORT	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			TABLE specified)								
48			Granting the MERGE CHUNK privilege (GRANT statement with MERGE CHUNK specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
49			Granting the CHANGE CHUNK COMMENT privilege (GRANT statement with CHANGE CHUNK COMMENT specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
50			Granting the CHANGE CHUNK STATUS privilege (GRANT statement with CHANGE CHUNK STATUS specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
51			Granting the ARCHIVE CHUNK privilege (GRANT statement with ARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
52			Granting the UNARCHIVE CHUNK privilege	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			(GRANT statement with UNARCHIVE CHUNK specified)								
53			Revoking the DBA privilege (REVOKE statement with DBA specified)	Normal termination[#6]	Y	Y					Y
54			Revoking the CONNECT privilege (REVOKE statement with CONNECT specified)	Normal termination[#6]	Y	Y					Y
55			Revoking the SCHEMA privilege (REVOKE statement with SCHEMA specified)	Normal termination[#6]	Y	Y					Y
56			Revoking the SELECT privilege (REVOKE statement with SELECT specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
57			Revoking the INSERT privilege (REVOKE statement with INSERT specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
58			Revoking the UPDATE privilege (REVOKE statement with UPDATE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
59			Revoking the DELETE privilege (REVOKE statement with DELETE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
60			Revoking the TRUNCATE privilege (REVOKE statement with TRUNCATE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
61			Revoking the REFERENCES privilege (REVOKE statement with REFERENCES specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
62			Revoking the IMPORT TABLE privilege (REVOKE statement with IMPORT TABLE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
63			Revoking the REBUILD INDEX	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			privilege (REVOKE statement with REBUILD INDEX specified)								
64			Revoking the GET COSTINFO privilege (REVOKE statement with GET COSTINFO specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
65			Revoking the EXPORT TABLE privilege (REVOKE statement with EXPORT TABLE specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
66			Revoking the MERGE CHUNK privilege (REVOKE statement with MERGE CHUNK specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
67			Revoking the CHANGE CHUNK COMMENT privilege (REVOKE statement with CHANGE CHUNK COMMENT specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
68			Revoking the CHANGE CHUNK STATUS privilege (REVOKE statement with CHANGE CHUNK STATUS specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
69			Revoking the ARCHIVE CHUNK privilege (REVOKE statement with ARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
70			Revoking the UNARCHIVE CHUNK privilege (REVOKE statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
71			Creating an HADB user (CREATE USER statement)	Normal termination	Y	Y					Y
72			Deleting an HADB user (DROP USER statement)	Normal termination	Y	Y					Y
73			Changing user information for an HADB user	Normal termination	Y	Y					Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			(ALTER USER statement)								
74		Definition SQL event	Defining an index (CREATE INDEX statement)	Normal termination	Y	Y	INDEX	Y	Y	Y	Y
75			Defining a schema (CREATE SCHEMA statement)	Normal termination	Y	Y	SCHEMA	Y		Y	Y
76			Defining a base table (CREATE TABLE statement)	Normal termination	Y	Y	TABLE	Y	Y	Y	Y
77			Defining a viewed table (CREATE VIEW statement)	Normal termination	Y	Y	VIEW	Y	Y	Y	Y
78			Deleting an index (DROP INDEX statement)	Normal termination	Y	Y	INDEX	Y	Y	Y	Y
79			Deleting a schema (DROP SCHEMA statement)	Normal termination	Y	Y	SCHEMA	Y		Y	Y
80			Deleting a base table (DROP TABLE statement)	Normal termination	Y	Y	TABLE	Y	Y	Y	Y
81			Deleting a viewed table (DROP VIEW statement)	Normal termination	Y	Y	VIEW	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
82			Changing the definition of a base table (ALTER TABLE statement)	Normal termination	Y	Y	TABLE	Y	Y	Y	Y
83			Changing the definition of a viewed table (ALTER VIEW statement)	Normal termination	Y	Y	VIEW	Y	Y	Y	Y
84		Data manipulation SQL event	Retrieving data from tables (SELECT statement)	Normal termination	Y	Y	TABLE, VIEW, or TABLE FUNCTION	Y	Y	Y	Y
85			Inserting rows into tables (INSERT statement[#3])	Normal termination	Y	Y	TABLE or VIEW	Y	Y	Y	Y
86			Updating table row data (UPDATE statement[#3])	Normal termination	Y	Y	TABLE or VIEW	Y	Y	Y	Y
87			Deleting table row data (DELETE statement[#3])	Normal termination	Y	Y	TABLE or VIEW	Y	Y	Y	Y
88			Deleting all row data from a table (TRUNCATE TABLE statement)	Normal termination	Y	Y	TABLE	Y	Y	Y	Y
89			Deleting all row data in a chunk (PURGE	Normal termination	Y	Y	TABLE	Y	Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			CHUNK statement[#3])								
90			SQL parsing error		Y	Y					Y
91			Acquiring data stored in chunks (#GETDATA subcommand of adbsql command)	Normal termination	Y	Y					Y
92			Acquiring the number of data items stored in a chunk (#GETCOUNT subcommand of adbsql command)	Normal termination	Y	Y					Y
93			Displaying table information (#TABLES subcommand of adbsql command)	Normal termination	Y	Y					Y
94			Displaying column information (#COLUMNS subcommand of adbsql command)	Normal termination	Y	Y					Y
95			Displaying index information (#INDEXES subcommand of	Normal termination	Y	Y					Y

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			adbsql command)								
96			Displaying chunk information (#CHUNKS subcommand of adbsql command)	Normal termination	Y	Y					Y
97			Displaying authorization identifiers (#GETUSER subcommand of adbsql command)	Normal termination	Y	Y					Y
98		Command operation event	Importing data (adbimport command)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	Y
99			Exporting data (adbexport command without -q option specified)	Normal termination[#2]	Y	Y	TABLE or VIEW	Y	Y	Y	Y
100			Exporting data (adbexport command with -q option specified)	Normal termination[#2]	Y	Y	TABLE, VIEW, or TABLE FUNCTION	Y	Y	Y	Y
101			Rebuilding indexes (adbidxrebuild command)	Normal termination[#2]	Y	Y	INDEX	Y	Y	Y	

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
102			Collecting cost information (adbgetcst command)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	
103			Analyzing DB status (adbdbstatus command)	Normal termination[#2]	Y	Y					
104			Merging chunks (adbmergechunk command)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	
105			Setting, changing, and deleting chunk comments (adbchgchunkcomment command)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	
106			Changing the chunk status (adbchgchunkstatus command)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	
107			Archiving chunks (adbarchivechunk command)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	
108			Unarchiving chunks (adbunarchivechunk command)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	
109			Reorganizing system tables (base tables)	Normal termination[#2]	Y	Y	TABLE	Y	Y	Y	

Audit event					Column structure of table function derived table for audit trail (4 of 5)						
No.	Event category	Event type	Event Description		Message log information (MESSAGE_LOG_INFO)	End code (EXIT_STATUS)	Object type[#4] (OBJECT_TYPE)	Object owner[#4] (OBJECT_OWNER_NAME)	Object schema name[#4] (OBJECT_SCHEMA_NAME)	Object name[#4] (OBJECT_NAME)	Access count[#5] (ACCESS_COUNT)
			(`adbreorgsystemdata` command)								
110			Registering and deleting synonym dictionaries (`adbsynddict` command)	Normal termination[#2]	Y	Y					

Table 12-25: Audit trail output triggers and output items (5 of 5)

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
1	Mandatory audit event	System event	Starting the HADB server (`adbstart` command)						C	Y	Y			
2			Stopping the HADB server (`adbstop` command)	Normal termination										
3			Changing the HADB server operatio	Normal termination					Y	Y				

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			n mode (adbchgsrvmode command)											
4			Changing the node type (adbchgnodetype command)	Normal termination					Y	Y				
5			Starting and stopping output of SQL trace information (adbchgsqltrc command)	Normal termination					Y	Y				
6			Centrally managing client definitions (adbclientdefmng command with --	Normal termination[#2]					Y	Y				

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			update option specified)											
7			Centrally managing client definitions (adbclientdefmang command with -i option specified)	Normal termination[#2]										
8			Adding and modifying DB areas (adbmodarea command)	Normal termination[#2]					Y	Y				
9			Changing a buffer (adbmodbuff command)	Normal termination[#2]					Y	Y				
10			Managing the	Normal					Y	Y				

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			updated-row columnizing facility (`adbcolumnize` command)	termination										
11		Audit event	Granting audit admin privilege (`GRANT` statement with `AUDIT ADMIN` specified)	Normal termination[#6, #7]	Y	Y	Y					Y	Y	Y
12			Granting audit viewer privilege (`GRANT` statement with `AUDIT VIEWER` specified)	Normal termination[#6, #7]	Y	Y	Y					Y	Y	Y
13			Revoking audit admin privilege	Normal termi natio	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			(REVOKE statement with AUDIT ADMIN specified)	n[#6, #7]										
14			Revoking audit viewer privilege (REVOKE statement with AUDIT VIEWER specified)	Normal termination[#6, #7]	Y	Y	Y					Y	Y	Y
15			Defining audit targets (CREATE AUDIT statement)	Normal termination			Y					Y	Y	Y
16			Deleting audit target definition information (DROP AUDIT statement)	Normal termination			Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
17			Changing auditor passwords (ALTER USER statement)	Normal termination		Y	Y					Y	Y	Y
18			Enabling the audit trail facility (adbauditt rail command with --start option specified)	Normal termination[#2]										
19			Disabling the audit trail facility (adbauditt rail command with --stop option specified)	Normal termination[#2]										
20			Swapping the	Normal										

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			audit trail file (`adbauditt rail` command with `--swap` option specified)	termination[#2, #9]										
21			Referencing information related to the audit trail facility (`adbauditt rail` command with `-d` option specified)	Normal termination[#2, #9]										
22			Executing a system-defined function for audit trails (`ADB_AUDIT READ` function) (data	Normal termination			Y	Y				Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			manipulation SQL)											
23			Retrieving data from a viewed table that depends on a derived table that specifies an ADB_AUDITREAD function (data manipulation SQL)	Normal termination			Y	Y				Y	Y	Y
24			Executing a system-defined function for audit trails (ADB_AUDITREAD function) (adbexport command with -q option	Normal termination[#2]			Y							

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			specified)											
25			Retrieving data from a viewed table that depends on a derived table that specifies an `ADB_AUDITREAD` function (`adbexport` command with `-q` option specified)	Normal termination[#2]			Y							
26			Retrieving data from `SQL_AUDITS` dictionary table (data manipulation SQL)	Normal termination			Y	Y				Y	Y	Y
27			Retrieving data from	Normal termi			Y	Y				Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			viewed tables that depend on the SQL_AUDITS dictionary table (data manipulation SQL)	nation										
28			Retrieving data from SQL_AUDITS dictionary table (adbexport command without -q option specified)	Normal termination[#2]										
29			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport	Normal termination[#2]										

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			command without -q option specified)											
30			Retrieving data from SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]			Y							
31			Retrieving data from viewed table that depends on SQL_AUDITS dictionary table (adbexport command with -q option specified)	Normal termination[#2]			Y							

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
32			Converting audit trail files (`adbconvertaudittrailfile` command)	Normal termination[#2, #9]										
33	Optional audit event	Session event	Connecting to the HADB server	Normal termination										
34			Disconnecting from the HADB server	Normal termination[#1]								Y	Y	Y
35		Privilege management event	Granting the `DBA` privilege (`GRANT` statement with `DBA` specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
36			Granting the `CONNECT` privilege (`GRAN`	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			T statement with CONNECT specified)											
37			Granting the SCHEMA privilege (GRANT statement with SCHEMA specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
38			Granting the SELECT privilege (GRANT statement with SELECT specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
39			Granting the INSERT privilege (GRAN	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			T statement with INSERT specified)											
40			Granting the UPDATE privilege (GRANT statement with UPDATE specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
41			Granting the DELETE privilege (GRANT statement with DELETE specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
42			Granting the TRUNCATE privilege (GRAN	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			T statement with TRUNCATE specified)											
43			Granting the REFERENCES privilege (GRANT statement with REFERENCES specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
44			Granting the IMPORT TABLE privilege (GRANT statement with IMPORT TABLE specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
45			Granting the REBUILD INDEX	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			privilege (GRANT statement with REBUILD INDEX specified)											
46			Granting the GET COSTINFO privilege (GRANT statement with GET COSTINFO specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
47			Granting the EXPORT TABLE privilege (GRANT statement with EXPORT TABLE specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
48			Granting the MERGE CHUNK privilege (GRANT statement with MERGE CHUNK specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
49			Granting the CHANGE CHUNK COMMENT privilege (GRANT statement with CHANGE CHUNK COMMENT specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
50			Granting the CHANGE CHUNK STATUS privilege	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			(GRANT statement with CHANGE CHUNK STATUS specified)											
51			Granting the ARCHIVE CHUNK privilege (GRANT statement with ARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
52			Granting the UNARCHIVE CHUNK privilege (GRANT statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
53			Revoking the DBA privilege (REVOKE statement with DBA specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
54			Revoking the CONNECT privilege (REVOKE statement with CONNECT specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
55			Revoking the SCHEMA privilege (REVOKE statement with SCHEMA specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
56			Revoking the	Normal	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			SELECT privilege (REVOKE statement with SELECT specified)	termination[#6]										
57			Revoking the INSERT privilege (REVOKE statement with INSERT specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
58			Revoking the UPDATE privilege (REVOKE statement with UPDATE specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
59			Revoking the	Normal	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			DELETE privilege (REVOKE statement with DELETE specified)	termination[#6]										
60			Revoking the TRUNCATE privilege (REVOKE statement with TRUNCATE specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
61			Revoking the REFERENCES privilege (REVOKE statement with REFERENCES specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
62			Revoking the	Normal	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			IMPORT TABLE privilege (REVOKE statement with IMPORT TABLE specified)	termination[#6]										
63			Revoking the REBUILD INDEX privilege (REVOKE statement with REBUILD INDEX specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
64			Revoking the GET COSTINFO privilege (REVOKE statement with GET COSTI	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			NFO specified)											
65			Revoking the EXPORT TABLE privilege (REVOKE statement with EXPORT TABLE specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
66			Revoking the MERGE CHUNK privilege (REVOKE statement with MERGE CHUNK specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
67			Revoking the CHANGE CHUNK COMMENT privilege	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			(REVOKE statement with CHANGE CHUNK COMMENT specified)											
68			Revoking the CHANGE CHUNK STATUS privilege (REVOKE statement with CHANGE CHUNK STATUS specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
69			Revoking the ARCHIVE CHUNK privilege (REVOKE statement with ARCHI	Normal termination[#6]	Y	Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			VE CHUNK specified)											
70			Revoking the UNARCHIVE CHUNK privilege (REVOKE statement with UNARCHIVE CHUNK specified)	Normal termination[#6]	Y	Y	Y					Y	Y	Y
71			Creating an HADB user (CREATE USER statement)	Normal termination		Y	Y					Y	Y	Y
72			Deleting an HADB user (DROP USER statement)	Normal termination		Y	Y					Y	Y	Y
73			Changing user information for	Normal termi		Y	Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			an HADB user (ALTER USER statement)	nation										
74		Definition SQL event	Defining an index (CREATE INDEX statement)	Normal termination			Y					Y	Y	Y
75			Defining a schema (CREATE SCHEMA statement)	Normal termination			Y					Y	Y	Y
76			Defining a base table (CREATE TABLE statement)	Normal termination			Y					Y	Y	Y
77			Defining a viewed table (CREATE VIEW	Normal termination			Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			statement)											
78			Deleting an index (DROP INDEX statement)	Normal termination			Y					Y	Y	Y
79			Deleting a schema (DROP SCHEMA statement)	Normal termination			Y					Y	Y	Y
80			Deleting a base table (DROP TABLE statement)	Normal termination			Y					Y	Y	Y
81			Deleting a viewed table (DROP VIEW statement)	Normal termination			Y					Y	Y	Y
82			Changing the definition of a base table (ALTER	Normal termination			Y					Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			TABLE statement)											
83			Changing the definition of a viewed table (ALTER VIEW statement)	Normal termination			Y					Y	Y	Y
84		Data manipulation SQL event	Retrieving data from tables (SELECT statement)	Normal termination			Y	Y				Y	Y	Y
85			Inserting rows into tables (INSERT statement[#3])	Normal termination			Y	Y				Y	Y	Y
86			Updating table row data (UPDATE statement[#3])	Normal termination			Y	Y				Y	Y	Y
87			Deleting table	Normal			Y	Y				Y	Y	Y

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			row data (DELETE statement[#3])	termination										
88			Deleting all row data from a table (TRUNCATE TABLE statement)	Normal termination			Y					Y	Y	Y
89			Deleting all row data in a chunk (PURGE CHUNK statement[#3])	Normal termination			Y	Y				Y	Y	Y
90			SQL parsing error				Y					Y	Y	Y
91			Acquiring data stored in chunks (#GETDATA subcommand of adbsql	Normal termination										

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			command)											
92			Acquiring the number of data items stored in a chunk (#GETCOUNT subcommand of adbsql command)	Normal termination										
93			Displaying table information (#TABLES subcommand of adbsql command)	Normal termination										
94			Displaying column information (#COLUMNS subcommand of adbsql	Normal termination										

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			command)											
95			Displaying index information (#INDEXES subcommand of adbsql command)	Normal termination										
96			Displaying chunk information (#CHUNKS subcommand of adbsql command)	Normal termination										
97			Displaying authorization identifiers (#GETUSER subcommand of adbsql command)	Normal termination										

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
98		Command operation event	Importing data (`adbimport` command)	Normal termination[#2]										
99			Exporting data (`adbexport` command without `-q` option specified)	Normal termination[#2]										
100			Exporting data (`adbexport` command with `-q` option specified)	Normal termination[#2]			Y							
101			Rebuilding indexes (`adbidxrebuild` command)	Normal termination[#2]										
102			Collecting cost information (`adbgetcst`	Normal termination[#2]										

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
			command)											
103			Analyzing DB status (adbdbstatus command)	Normal termination[#2]										
104			Merging chunks (adbmergechunk command)	Normal termination[#2]										
105			Setting, changing, and deleting chunk comments (adbchgchunkcomment command)	Normal termination[#2]										
106			Changing the chunk status (adbchgchunkstatus command)	Normal termination[#2]										

Audit event					Column structure of table function derived table for audit trail (5 of 5)									
No.	Event category	Event type	Event Description		Privilege that was granted or revoked (PRIVILEGE_TYPE)	Authorization identifier targeted by the event (PRIVILEGE_USER_NAME)	SQL statement (SQL_SOURCE)	Dynamic parameters (PARAM)	System change information (from before change) (BEFORE_SYSTEM_INFO)	System change information (from after change) (AFTER_SYSTEM_INFO)	Database configuration information (SERVER_OPERAND)	Additional user information 1 (USER_INFO_1)	Additional user information 2 (USER_INFO_2)	Additional user information 3 (USER_INFO_3)
107			Archiving chunks (`adbarchivechunk` command)	Normal termination[#2]										
108			Unarchiving chunks (`adbunarchivechunk` command)	Normal termination[#2]										
109			Reorganizing system tables (base tables) (`adbreorgsystemdata` command)	Normal termination[#2]										
110			Registering and deleting synonym dictionaries (`adbsyndict` command)	Normal termination[#2]										

Legend:

Y

If there is output information, the applicable information is output. If there is no output information, NULL is output.

Blank

NULL is output.

A

If the connection uses UNIX domain communication, NULL is output.

B

If the client is the JDBC driver, NULL is output.

C

When using the multi-node function, different nodes will output different information.

- For the master node, the output information is the HADB server operation mode of the node that was the master node the last time the HADB server started.
- For the slave node, NULL is always output in this column.

D

NULL is output if this operation occurs as an additional part of command processing.

E

If either of the following applies, NULL is output:

- The connection uses UNIX domain communication
- The operation occurs as an additional part of command processing

F

When using the multi-node function, the IP address and port number used by the master node for inter-server communication are output for audit trails to which any of the following apply. For audit trails to which none of the following apply, NULL is output.

- An audit trail for which ADBSTOP is output in the event subtype (EVENT_SUBTYPE) column when the adbstop command is executed on the master node without specifying the --node option

  This applies to audit trails output by the slave node.
- An audit trail for which ADBAUDITTRAIL SWAP is output in the event subtype (EVENT_SUBTYPE) column when the adbauditttrail --swap command is executed with the node number of the slave node specified in the -n option

  This applies to audit trails whose processing was executed on the slave node and which were output by the slave node.
- An audit trail for which ADBAUDITTRAIL DISPLAY is output in the event subtype (EVENT_SUBTYPE) column when the adbauditttrail -d command is executed with the node number of the slave node specified in the -n option

  This applies to audit trails whose processing was executed on the slave node and which were output by the slave node.

Note:

For output information (such as EVENT and SYSTEM) that does not link to an entry in the preceding legend, the information described in the table is output. For details, see Table 12-10: Column structure of table function derived table when retrieving audit trails in 12.9.2 Column structure of table function derived table when retrieving audit trails.

Note that if there is no output information in the object type (OBJECT_TYPE) column, NULL is output.

#1

This does not include use of the close method of the Connection object when using connection pooling (it is output in the SQL trace information as the DISP call type).

#2

Before and after this event, audit trails for which CONNECT and DISCONNECT are output in the event subtype column (EVENT_SUBTYPE) are each output. For each of these audit trails, the executed command name is output in the following columns:

- Application identifier (AP_NAME) column
- Client process name (CLIENT_PROCESS_NAME) column

#3

The information output to audit trails differs when an SQL statement is executed that includes a query expression body in data manipulation SQL other than a SELECT statement. In this case, audit trails related to the objects described in the query expression body show the same information in this table (Retrieving data from tables (SELECT statement).

#4

If either of the following applies, NULL is output even if the table contains output information for that column:

- The objects targeted by the event are all table value constructors
- During command execution, processing was not executed with respect to any object

#5

NULL is output if the statement handle was closed with only the SQL preprocessing completed. This is to distinguish from situations where execution of the SQL statement has completed. This does not apply to commands.

#6

If privileges did not need to be granted to or revoked from any of the authorization identifiers specified in the SQL statement, NULL is output in the following columns:

- Object type (OBJECT_TYPE) column
- Object owner (OBJECT_OWNER_NAME) column
- Object schema name (OBJECT_SCHEMA_NAME) column
- Object name (OBJECT_NAME) column
- Privilege that was granted or revoked (PRIVILEGE_TYPE) column
- Authorization identifier targeted by the event (PRIVILEGE_USER_NAME) column

#7

If privileges did not need to be granted to or revoked from any of the authorization identifiers specified in the SQL statement, PRIVILEGE is output in the event type (EVENT_TYPE) column.

#8

Information is output in this column when using the multi-node function. If the multi-node function is not being used, NULL is output.

#9

If the multi-node function is being used, audit trails for which CONNECT and DISCONNECT are output in the event subtype column (EVENT_SUBTYPE) are always output on the master node.

12.10 Notes about using the audit trail facility

This section provides notes that apply to the use of the audit trail facility.

- 12.10.1 Situations where multiple audit trails are output for a single event
- 12.10.2 Notes about audit trails output during command execution

12.10.1 Situations where multiple audit trails are output for a single event

If multiple objects are targets of an audit target event, an audit trail is output for each of these objects. For example, when the following SQL statement is executed, an audit trail is output for table `T1` and another is output for table `T2`.

```
SELECT * FROM "T1","T2"
```

Similarly, if an event targets multiple privileges or authorization identifiers, a number of audit trails are output equivalent to the number of combinations of privileges and authorization identifiers. The following table lists the situations in which multiple audit trails are output for a single event:

Table 12-26: Situations where multiple audit trails are output for a single event

No.	Executed event	Number of output audit trails
1	Granting audit admin privilege	Number of authorization identifiers to which privilege was granted
2	Granting audit viewer privilege	
3	Revoking audit admin privilege	Number of authorization identifiers from which privilege was revoked
4	Revoking audit viewer privilege	
5	Granting privileges (excluding access privileges)	MAX (1, number of granted privileges × number of authorization identifiers to which privileges were granted)
6	Granting privileges (access privileges)	MAX (1, number of granted access privileges[#1] × number of target objects[#2] × number of authorization identifiers to which privileges were granted)[#6, #7]
7	Revoking privileges (excluding access privileges)	MAX (1, number of revoked privileges × number of authorization identifiers whose privileges were revoked)
8	Revoking privileges (access privileges)	MAX (1, number of revoked access privileges[#1] × number of target objects[#2] × number of authorization identifiers whose privileges were revoked)[#6, #7]
9	Retrieving data from tables (using `SELECT` statement)	MAX (1, number of target objects specified in query expression body[#4])
10	Retrieving data from tables (using data manipulation SQL other than `SELECT` statement[#3])	Number of target objects specified in query expression body[#4]
11	Re-creating indexes	MAX (1, number of rebuilt indexes[#5])
12	Collecting cost information	MAX (1, number of tables for which cost information was collected[#5])
13	Exporting data (using `adbexport` command with `-q` option specified)	MAX (1, number of target objects specified in query expression bodies in SQL statements in the SQL statement file[#4])

#1

If a `GRANT` statement or `REVOKE` statement is executed with `ALL PRIVILEGES` specified, the audit trail facility operates as if all access privileges were specified.

#2

If a GRANT statement or REVOKE statement is executed with ALL TABLES specified, the number of target objects is the total number of base tables in the schema of the HADB user who executed the GRANT or REVOKE statement. However, if the SQL statement results in an error, the number of output audit trails might not be equivalent to the number of base tables.

#3

If data manipulation SQL such as an INSERT statement is executed that includes a query expression body, audit trails related to the objects specified in the query expression body are output as events that retrieve data from tables.

#4

If the same object is specified multiple times, duplicate audit trails are eliminated. Table value constructors are not counted in the number of objects.

#5

Information about the success or failure of the event is output in audit trails for each object. Because command processing results in an error when an event fails, audit trails will not be output for objects specified after the object for which the event failed.

#6

If a GRANT statement terminates normally without granting any of the specified privileges, only one audit trail is output for the GRANT statement. In this case, NULL is output as the object, privilege type, and authorization identifier of the HADB user. The same applies to REVOKE statements.

#7

If an SQL statement results in an error when granting or revoking access privileges, because no access privileges will have been granted or revoked, all of the access privileges specified in the SQL statement are subject to audit trail output. If the target object type cannot be identified due to an error, NULL is output as the object type.

12.10.2 Notes about audit trails output during command execution

(1) Audit trails output when commands are executed that connect to the HADB server

When you execute a command that connects to the HADB server, audit trails are output for the following events, in addition to the audit trails for the command itself.

- Connecting to the HADB server
- Disconnecting from the HADB server

For details about the commands that connect to the HADB server, see *List of commands* in the manual *HADB Command Reference*.

(2) Notes about executing the adbcancel command

The information output in audit trails and command return values differ as shown in the following table, depending on the timing with which the following event occurs: the adbcancel command is executed for a command that connects to the HADB server or the HADB server detects a communication error during execution of a command that connects to the HADB server.

Table 12-27: Information output in audit trails and command return values

Timing with which `adbcancel` command was executed or HADB server detected communication error	Audit trails for connecting to the HADB server	Audit trails for command	Audit trails for disconnecting from the HADB server	Command return value
Between connecting to the HADB server and command processing	Output as SUCCESS	Not output	Output as OCCURRENCE#	Error
Between command processing and disconnecting from the HADB server	Output as SUCCESS	Output as SUCCESS if command processing is successful	Output as OCCURRENCE#	Error

#

If OCCURRENCE is output as the event result, disconnection from the HADB server itself has been completed. For details about the reason why OCCURRENCE was output, see the message log.

(3) Notes about executing the adbaudittrail command

- If renaming of the current audit trail file fails when either of the following is executed, the end code output in the audit trail will not match the return code of the command.
  - The HADB server is terminated
  - The audit trail facility is disabled
- The following table shows the nodes to which audit trails are output when executing the `adbaudittrail` command in an environment that uses the multi-node function:

Table 12-28: Node to which audit trails are output when `adbaudittrail` command is executed

No.	Option specified in `adbaudittrail` command	Output destination node for audit trails		
		Audit trails for connecting to the HADB server	Audit trails for `adbaudittrail` command	Audit trails for disconnecting from the HADB server
1	`--start`	Master node	Master node	Master node
2	`--stop`			
3	`--swap`	Master node#1	Node on which the command was executed	Master node#1
4	`--swap -n` *node-number*	Master node	Node whose node number is specified in the `-n` option#2, #3	Master node
5	`-d`	Master node#1	Node on which the command was executed	Master node#1
6	`-d -n` *node-number*	Master node	Node whose node number is specified in the `-n` option#2, #3	Master node

#1

If the command is executed on the master node, `NULL` is output for `CLIENT_IP_ADDRESS` and `CLIENT_PORT_NUMBER`. If the command is executed on the slave node, the IP address and port number that the `adbaudittrail` command used to connect to the HADB server on the master node are output for `CLIENT_IP_ADDRESS` and `CLIENT_PORT_NUMBER`. `CLIENT_IP_ADDRESS` and `CLIENT_PORT_NUMBER` are the column names of the table function derived table used when retrieving audit trail data. For details, see 12.9.2 Column structure of table function derived table when retrieving audit trails.

#2

If you specify the node number of the slave node in the -n option of the adbauditttrail command, the IP address and port number used by the master node for inter-server communication are output for CLIENT_IP_ADDRESS and CLIENT_PORT_NUMBER. If you specify the node number of the master node, NULL is output for CLIENT_IP_ADDRESS and CLIENT_PORT_NUMBER. CLIENT_IP_ADDRESS and CLIENT_PORT_NUMBER are the column names of the table function derived table used when retrieving audit trail data. For details, see 12.9.2 Column structure of table function derived table when retrieving audit trails.

#3

If either of the following errors occurs, audit trails for the adbauditttrail command are output to the master node:

- The adbauditttrail command was executed by an HADB user who does not have the audit admin privilege (or any audit privilege in the case of the -d option)
- The adbauditttrail command is executed when the audit trail facility is disabled

(4) Notes about executing the adbconvertaudittrailfile command

The following table shows the nodes to which audit trails are output when executing the adbconvertaudittrailfile command in an environment that uses the multi-node function.

Table 12-29: Node to which audit trails are output when adbconvertaudittrailfile command is executed

Output destination node for audit trails		
Audit trails for connecting to the HADB server	**Audit trails for the adbconvertaudittrailfile command**	**Audit trails for disconnecting from the HADB server**
Master node#	Node on which the command was executed	Master node#

#

If the command is executed on the master node, NULL is output for CLIENT_IP_ADDRESS and CLIENT_PORT_NUMBER. If the command is executed on the slave node, the IP address and port number that the adbconvertaudittrailfile command used to connect to the HADB server on the master node are output for CLIENT_IP_ADDRESS and CLIENT_PORT_NUMBER. CLIENT_IP_ADDRESS and CLIENT_PORT_NUMBER are the column names of the table function derived table used when retrieving audit trail data. For details, see 12.9.2 Column structure of table function derived table when retrieving audit trails.

13 Tuning

This chapter explains how to tune HADB.

13.1 Tuning to improve processing performance

This section explains how to tune the HADB server in order to improve its processing performance.

For details about the tuning method for improving each command's processing performance, see 13.3 Tuning to shorten command execution time.

13.1.1 Pre-reading of range indexes

If the table scan during the first table retrieval following startup of the HADB server is slow, consider applying pre-reading of range indexes. For details about pre-reading of range indexes, see 5.5.3 Pre-reading of range indexes.

13.1.2 Re-evaluating the defined range indexes

This subsection explains how to re-evaluate the defined range indexes by checking the status of retrieval processing that uses the range indexes.

Check the SQL statement statistical information in the normal manner. Check the status of retrieval processing that uses range indexes, and then, if necessary, re-evaluate the defined range indexes.

If you are checking the status of retrieval processing on multiple tables for which range indexes have been defined, check the access path statistical information. The SQL statement statistical information does not provide the information needed to identify the range indexes that might need to be re-evaluated.

The following subsections explain how to re-evaluate the defined range indexes.

- If you will be checking SQL statement statistical information:
  See (1) Re-evaluating range indexes by checking the SQL statement statistical information.
- If you will be checking access path statistical information:
  See (2) Re-evaluating range indexes by checking the access path statistical information.

(1) Re-evaluating range indexes by checking the SQL statement statistical information

Check the SQL statement statistical information to determine the status of retrieval processing that uses a range index.

Check the SQL statement statistical information by using one of the following methods:

- Executing the `adbstat` command
  See *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.
- Checking the SQL trace files
  See (9) SQL statement statistical information in 10.11.2 Information that is output as SQL trace information.

The following procedure explains how to determine the status of retrieval processing that uses a range index by checking the SQL statement statistical information.

Procedure

1. Check chunk skipping to determine the range index's usage status.

Determine how many times chunks were skipped during retrieval processing that used the range index.

Check the following items in the SQL statement statistical information:

- `Ridx_chunk_skip_cnt` (number of times table chunks were skipped during retrievals that used the range index)
- `Ridx_chunk_judge_cnt` (number of times a check was made as to whether a table chunk stored data in the value range satisfying the search condition during retrieval processing that used the range index)

Based on the checked items, obtain the value resulting from the following formula:

Formula

Ridx_chunk_skip_cnt ÷ Ridx_chunk_judge_cnt

2. Check segment skipping to determine the range index's usage status.

   Determine how many times segments were skipped during retrieval processing that used the range index.

   Check the following items in the SQL statement statistical information:

   - `Ridx_sgmt_skip_cnt` (number of times table segments were skipped during retrievals that used the range index)
   - `Ridx_sgmt_judge_cnt` (number of times a check was made as to whether a table chunk stored data in the value range satisfying the search condition during retrieval processing that used the range index)

   Based on the checked items, obtain the value resulting from the following formula:

   Formula

   Ridx_sgmt_skip_cnt ÷ Ridx_sgmt_judge_cnt

3. Re-evaluate the defined range index.

   If the values obtained in steps 1 and 2 are both zero or close to zero, there is no benefit of using the range index (chunks and segments are not being skipped). Therefore, consider deleting the defined range index.

   When an ineffective range index is used, it might take more time to perform retrieval processing than when no range index is used.

   If the values obtained in steps 1 and 2 are both close to 1 or one of them is close to 1, there are benefits of using the range index (chunks or segments are being skipped). Therefore, there is no need to re-evaluate the defined range index.

Important

Perform step 1, even for a single-chunk table. With range indexes, a single-chunk table is treated as one chunk when chunks are skipped. Therefore, you need to check whether the defined range index is effective.

(2) Re-evaluating range indexes by checking the access path statistical information

Check the access path statistical information to determine the status of retrieval processing that uses a range index.

Check the access path statistical information in the SQL trace files. For details about access path statistical information, see 10.11.3 Examples of output of and output items for access path statistical information.

The following procedure explains how to determine the status of retrieval processing that uses a range index by checking the access path statistical information.

Procedure

1. Check chunk skipping to determine the range index's usage status.

   Determine how many times chunks were skipped during retrieval processing that used the range index.

   Check the following items in the access path statistical information:

   - `Data_ridx_chunk_skip_cnt` (number of times table chunks were skipped during retrievals that used the range index)
   - `Data_ridx_chunk_judge_cnt` (number of times table chunks were checked during retrievals that used the range index)

   Based on the checked items, obtain the value resulting from the following formula:

   Formula

   Data_ridx_chunk_skip_cnt ÷ Data_ridx_chunk_judge_cnt

2. Check segment skipping to determine the range index's usage status.

   Determine how many times segments were skipped during retrieval processing that used the range index.

   Check the following items in the access path statistical information:

   - `Data_ridx_sgmt_skip_cnt` (number of times table segments were skipped during retrievals that used the range index)
   - `Data_ridx_sgmt_judge_cnt` (number of times table segments were checked during retrievals that used the range index)

   Based on the checked items, obtain the value resulting from the following formula:

   Formula

   Data_ridx_sgmt_skip_cnt ÷ Data_ridx_sgmt_judge_cnt

3. Re-evaluate the defined range index.

   If the values obtained in steps 1 and 2 are both zero or close to zero, there is no benefit of using the range index (chunks and segments are not being skipped). Therefore, consider deleting the defined range index.

   When an ineffective range index is used, it might take more time to perform retrieval processing than when no range index is used.

   If the values obtained in steps 1 and 2 are both close to 1 or one of them is close to 1, there are benefits of using the range index (chunks or segments are being skipped). Therefore, there is no need to re-evaluate the defined range index.

Important

Perform step 1, even for a single-chunk table. With range indexes, a single-chunk table is treated as one chunk when chunks are skipped. Therefore, you need to check whether the defined range index is effective.

13.1.3 Re-evaluating the defined text indexes

This subsection describes how to check whether text indexes are efficiently used.

Method of checking

Check the following output item in the access path statistical information output to the SQL trace file:

- `Data_tidx_all_search_cnt`

In the preceding item, the number of all data searches is output when data in a chunk is retrieved by using text indexes. If a value no less than 1 is output to the preceding item, consider taking a corrective action.

> **Note**
>
> For details about access path statistical information, see 10.11.3 Examples of output of and output items for access path statistical information.

Action to take

If you take either of the following actions, performance might be improved:

- Modifying the SQL statement so that individual retrieval results are unified by the use of `UNION`
- Modifying the SQL statement so that text indexes are not used for retrieval (by specifying the index specification)

For an example of modifying the SQL statement, see (2) Checking whether text indexes are effectively used in 11.16.11 Tuning synonym search function.

If an SQL statement you specified satisfies all the following conditions, the SQL statement might be the cause of performance deterioration:

- The `LIKE` predicate, `LIKE_REGEX` predicate, or scalar function `CONTAINS` is specified for a search condition.
- Multiple search conditions in each of which the preceding predicate or scalar function is specified are specified by using logical operator `OR`.
- In a search condition in which the preceding predicate or scalar function is specified, a text indexed column is specified.

An SQL statement that satisfies all the preceding conditions performs retrieval by using the search-target character strings specified in all search conditions. If there are too many search conditions, narrowing down of data by using text indexes is not performed. Instead, all data search by using text indexes is performed, so the search time might be considerably long.

In addition, when a synonym search operation is performed, internal processing unifies the search conditions (each of which has a synonym specified) by using logical operator `OR` such that multiple synonyms are specified. Therefore, if too many synonyms are registered in a synonym group, all data search by using text indexes is performed, instead of narrowing down data by using text indexes.

13.1.4 Reducing the HADB server startup time (by applying HugePages)

This section explains how to reduce the HADB server's startup time.

HADB's global buffers use the OS's shared memory. Consequently, if you specify in the HADB server definition that a large number of global buffers are to be allocated, it will occur frequently that a requested address will not be found in the TLB, which manages the OS's memory. This is called a *TLB miss*. As a result, the access performance of the shared memory will deteriorate, resulting in slowing down the HADB server's startup time.

In such a case, apply `HugePages` to HADB's shared memory. `HugePages` is a function that expands the TLB's page table. Expanding the TLB's page table reduces TLB misses and improves the access performance of the shared memory.

The following shows the procedure for applying `HugePages` to HADB's shared memory.

Setup method

1. Terminate the HADB server.

Terminate the HADB server by executing the `adbstop` command.

2. To enable `HugePages`, specify the following kernel parameters:
   - `vm.nr_hugepages` parameter
   - `vm.hugetlb_shm_group` parameter
   - `soft memlock` parameter
   - `hard memlock` parameter

   For details about individual kernel parameters, see 6.2 Estimating the kernel parameters.
3. Restart the OS.

   If you changed the kernel parameter specifications, restart the OS.
4. Change the server definition.

   Specify the `adb_sys_shm_huge_page_size` operand in the server definition.

   If you changed the value of the `vm.nr_hugepages` parameter in step 2, change also the value of the `adb_sys_memory_limit` operand in the server definition.
5. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the `adb_sys_shm_huge_page_size` and `adb_sys_memory_limit` operands in the server definition, see the adb_sys_shm_huge_page_size and adb_sys_memory_limit operands in 7.2.2 Operands related to performance (set format).

13.1.5 Preventing SQL statement execution wait status from occurring

This subsection explains how to improve processing performance by re-evaluating the number of processing real threads specified for the `adb_sys_rthd_num` operand in the server definition.

If the specified number of processing real threads is small, an SQL statement execution wait status might occur. When this wait status occurs for an SQL statement, processing of the SQL statement might take a long time. Therefore, check whether an appropriate number of processing real threads has been specified.

Procedure

1. Check the SQL statement execution wait status.

   Check the values of the following items in the HADB server statistical information that is output when the `adbstat` command is executed:
   - `SQL_execute_wait_total_time` (total SQL statement execution wait time)
   - `SQL_execute_wait_cnt` (number of times SQL statements waited for execution)

   If non-zero values were output to both `SQL_execute_wait_total_time` and `SQL_execute_wait_cnt`, an SQL statement execution wait status has occurred. In such a case, increasing the value specified for the `adb_sys_rthd_num` operand might prevent SQL statement execution wait status from occurring and improve processing performance in some cases.
2. Terminate the HADB server.

Terminate the HADB server by executing the adbstop command.

3. Change the server definition.

   Increase the value specified for the adb_sys_rthd_num operand in the server definition to increase the number of processing real threads used by the HADB server.

   Important

   If you changed the value of the adb_sys_rthd_num operand, always re-estimate the memory requirements for the HADB server. For details about estimation of the HADB server's memory requirement, see 6.3 Estimating the HADB server's memory requirement.

4. Start the HADB server.

   Execute the adbstart command to start the HADB server.

Note

For details about the adb_sys_rthd_num operand in the server definition, see the description of the adb_sys_rthd_num operand in 7.2.2 Operands related to performance (set format).

13.1.6 Reducing the SQL statement processing time

This subsection explains how to improve processing performance by re-evaluating the maximum number of SQL processing real threads specified in the adb_sql_exe_max_rthd_num operand in the server definition and client definition.

If the specified maximum number of SQL processing real threads is small, a small number of processing real threads will be used for executing SQL statements, and consequently processing of SQL statements might take a long time. Therefore, check whether an appropriate value is specified for the maximum number of SQL-processing real threads.

(1) How to reduce the SELECT statement's processing time

The following procedure explains how to reduce the SELECT statement's processing time.

Procedure

1. Check the execution time of the SELECT statement.

   Check the following item in the connection operation information that is output when the adbstat command is executed:

   - SELECT_total_time (SELECT statement execution time)

2. Revise the value specified for the adb_sql_exe_max_rthd_num operand.

   If the SELECT_total_time value is large, processing of SQL statements might be taking too long. In such a case, increasing the value specified for the adb_sql_exe_max_rthd_num operand in the server definition and client definition might improve processing performance in some cases.

   Important

   - Check the values of both the adb_sql_exe_max_rthd_num operand in the client definition and the adb_sql_exe_max_rthd_num operand in the server definition. If the value specified

in the client definition exceeds the value specified in the server definition as a result of increasing the value in the client definition only, the value specified in the server definition is applied. The new value specified in the client definition is not applied. If this case applies, increase the values of the adb_sql_exe_max_rthd_num operands in both the server definition and client definition.

- If you increase the value of the adb_sql_exe_max_rthd_num operand, increase also the value of the adb_sys_rthd_num operand in the server definition. If you increase only the value specified for the adb_sql_exe_max_rthd_num operand, SQL statement execution wait status might result.
- When you have changed the values of the adb_sql_exe_max_rthd_num and adb_sys_rthd_num operands, always re-estimate the HADB server's memory requirement. For details about estimation of the HADB server's memory requirement, see 6.3 Estimating the HADB server's memory requirement.

3. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
4. Change the server definition and client definition.
   Increase the values of the following operands:
   - The adb_sql_exe_max_rthd_num operands in the server definition and client definition
   - The adb_sys_rthd_num operand in the server definition
5. Start the HADB server.
   Execute the adbstart command to start the HADB server.

Note

- For details about the adb_sql_exe_max_rthd_num and adb_sys_rthd_num operands in the server definition, see the adb_sql_exe_max_rthd_num and adb_sys_rthd_num operands in 7.2.2 Operands related to performance (set format).
- For details about the adb_sql_exe_max_rthd_num operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

(2) How to reduce the processing time of definition SQL statements and data manipulation SQL statements

This subsection explains how to reduce the processing time of definition SQL statements and data manipulation SQL statements.

The methods provided here apply to the following SQL statements:

Definition SQL statements

- ALTER TABLE statement (when changing an archivable multi-chunk table to a regular multi-chunk table)
- DROP TABLE statement
- DROP INDEX statement
- DROP USER statement
- DROP SCHEMA statement

- REVOKE statement (revoking schema operation privileges)

Data manipulation SQLs

- PURGE CHUNK statement
- TRUNCATE TABLE statement

The following procedure explains how to reduce the processing time of definition SQL statements and data manipulation SQL statements.

Procedure

1. Check the number of DB area files.

   In the DB area summary information that is output when the adbdbstatus command is executed, check the following information:

   - Number_of_files (number of DB area files)

   Check the DB areas that store the tables and indexes subject to SQL statement processing. Identify the DB area containing the largest number of DB area files.

2. Revise the value specified for the adb_sql_exe_max_rthd_num operand.

   Compare the largest number of DB area files making up a DB area as displayed under Number_of_files in step 1 with the value of the adb_sql_exe_max_rthd_num operand. If the number of DB area files is larger than the operand's value, SQL statement processing time might be taking too long. If this is the case, increase the value of the adb_sql_exe_max_rthd_num operand so that it matches the number of DB area files. Processing performance might improve.

   Important

   - Check the values of both the adb_sql_exe_max_rthd_num operand in the client definition and the adb_sql_exe_max_rthd_num operand in the server definition. If the value specified in the client definition exceeds the value specified in the server definition as a result of increasing the value in the client definition only, the value specified in the server definition is applied. The new value specified in the client definition is not applied. If this case applies, increase the values of the adb_sql_exe_max_rthd_num operands in both the server definition and client definition.
   - If you increase the value of the adb_sql_exe_max_rthd_num operand, increase also the value of the adb_sys_rthd_num operand in the server definition. Unless you do so, SQL statement execution wait status might result.
   - When you have changed the values of the adb_sql_exe_max_rthd_num and adb_sys_rthd_num operands, always re-estimate the HADB server's memory requirement. For details about estimation of the HADB server's memory requirement, see 6.3 Estimating the HADB server's memory requirement.

3. Terminate the HADB server.

   Terminate the HADB server by executing the adbstop command.

4. Change the server definition and client definition.

   Increase the values of the following operands:

   - The adb_sql_exe_max_rthd_num operands in the server definition and client definition
   - The adb_sys_rthd_num operand in the server definition

5. Start the HADB server.

   Execute the adbstart command to start the HADB server.

Note

- For details about the adb_sql_exe_max_rthd_num and adb_sys_rthd_num operands in the server definition, see the adb_sql_exe_max_rthd_num and adb_sys_rthd_num operands in 7.2.2 Operands related to performance (set format).
- For details about the adb_sql_exe_max_rthd_num operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

13.1.7 Expanding the user log buffers

This subsection explains how to improve processing performance by re-evaluating the value of the adb_log_usrbuf_num operand in the server definition.

If the user log buffers become full, user logs are purged from the user log buffers and output to user log files. Because I/O operations on the user log files occur when the user log buffers are purged, update processing performance might deteriorate. Therefore, check whether the allocated number of user log buffers is appropriate.

Procedure

1. Check the number of times the user log buffers were purged because they became full.
   Check the following item in the information that was output when the adbstat command was executed:
   - Log_usrbuf_out_cnt (number of times the user log buffers were purged because they became full)

   Note

   For details about the adbstat command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.
2. Revise the value specified for the adb_log_usrbuf_num operand.
   If the value that is output to Log_usrbuf_out_cnt, which you checked in step 1, is at least 1, purging of the user log buffers has occurred. In such a case, expand the number of user log buffers by increasing the value of the adb_log_usrbuf_num operand in the server definition. Expanding the number of user log buffers will prevent purging of the user log buffers caused by full buffers during update processing.

   Important

   Expanding the number of user log buffers will increase the amount of process common memory required when the HADB server starts. Re-estimate the amount of process common memory required when the HADB server starts by following the explanation in (g) Determining the variable RECCTL under (3) Determining the process common memory requirement (for starting the HADB server) in 6.3.3 Determining the memory requirement for starting the HADB server.
3. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
4. Change the server definition.
   Increase the value specified for the adb_log_usrbuf_num operand in the server definition.
5. Start the HADB server.
   Execute the adbstart command to start the HADB server.

> **Note**
>
> For details about the adb_log_usrbuf_num operand in the server definition, see the adb_log_usrbuf_num operand in 7.2.3 Operands related to system logs (set format).

13.1.8 Expanding the initial size of user log files

This subsection explains how to improve processing performance by re-evaluating the value of the adb_log_usrfile_size operand in the server definition.

If an increase in the number of user logs causes the initial size of user log files to be exceeded, user log files are expanded automatically. Because expansion of user log files will have a significant impact on update processing performance, check whether the initial size of user log files is appropriate.

Procedure

1. Check the maximum size of user log files.
   Check the following item in the information that was output when the adbstat command was executed:
   - Log_usrfile_max_size (maximum size of user log files)

   > **Note**
   >
   > For details about the adbstat command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

2. Re-evaluate the value specified for the adb_log_usrfile_size operand.
   If the Log_usrfile_max_size value you checked in step 1 is larger than the *initial-size-of-user-log-files* value specified for the adb_log_usrfile_size operand in the server definition, expansion of user log files has occurred. In such a case, increase the *initial-size-of-user-log-files* value specified for the adb_log_usrfile_size operand by using the value that is output to Log_usrfile_max_size as a guideline. Increasing the *initial-size-of-user-log-files* value will prevent expansion of user log files from occurring during update processing.

   > **Important**
   >
   > Before you increase the *initial-size-of-user-log-files* value, re-estimate the size as explained in 6.12 Estimating the size of the system log files.

3. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
4. Change the server definition.
   Increase the *initial-size-of-user-log-files* value specified for the adb_log_usrfile_size operand in the server definition.
5. Start the HADB server.
   Execute the adbstart command to start the HADB server.

> **Note**
>
> For details about the adb_log_usrfile_size operand in the server definition, see the adb_log_usrfile_size operand in 7.2.3 Operands related to system logs (set format).

13.1.9 Re-evaluating the trigger size for reducing user log files

This subsection explains how to improve processing performance by re-evaluating the trigger value (specified for the adb_log_usrfile_size operand in the server definition) for reducing the size of user log files.

When the update transaction is settled, the size of a user log file used by that update transaction might be larger than the size specified for the trigger value for reducing the size of user log files. In this case, the HADB server reduces the size of the user log file to the initial size. An increase in the number of times the user log file size is reduced to the initial size might affect update performance. Therefore, perform the following procedure to re-evaluate the value specified for the adb_log_usrfile_size operand.

Procedure:

1. Check the number of times the user log file size was reduced to the initial size.
   Check the following item in the information that is output when the adbstat command is executed:
   - Log_usrfile_reduced_cnt (the number of times the user log file size was reduced to the initial size)

   > **Note**
   >
   > For details about the adbstat command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

2. Re-evaluate the value specified for the adb_log_usrfile_size operand.
   A large value output to Log_usrfile_reduced_cnt that you checked in step 1 indicates that the user log file size was reduced to the initial size many times. In this case, we recommend that you increase the following values specified for the adb_log_usrfile_size operand in the server definition:
   - *Initial size of user log files*
   - *Trigger value for reducing the size of user log files*

   As the preceding specification values increase, the number of times the user log file size is reduced to the initial size decreases.

   > **Tip**
   >
   > In the execution result of the adbstat command, the value output under Log_usrfile_max_size indicates the maximum size of user log files during normal operations. We recommend that you specify, as the trigger value for reducing the size of user log files for the adb_log_usrfile_size operand, the value obtained by multiplying the preceding value by 1.5 to 2 as a safety factor. This enables you to perform processing for reducing the size of user log files only when the database is massively updated (for example, by an unscheduled operation).

3. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
4. Change the server definition.

Increase the following values specified for the `adb_log_usrfile_size` operand in the server definition:

- *Initial size of user log files*
- *Trigger value for reducing the size of user log files*

5. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the `adb_log_usrfile_size` operand in the server definition, see the adb_log_usrfile_size operand in 7.2.3 Operands related to system logs (set format).

13.1.10 Preventing automatic extension of DB areas

This subsection explains how to prevent automatic extension of DB areas from reducing update processing performance.

If DB area automatic extension occurs when a regular file is used for a data DB area file, dictionary DB area file, or system-table DB area file, performance of update processing during DB area file expansion might deteriorate. Therefore, check whether automatic extension has occurred in the target DB areas.

Check the following item in the information that was output when the `adbstat` command was executed:

- `DBarea_extension_cnt` (number of times the DB areas were extended automatically)

> **Note**
>
> For details about the `adbstat` command, see *adbstat (Perform Statistical Analysis of the HADB Server)* in the manual *HADB Command Reference*.

If the value that was output to `DBarea_extension_cnt` is at least `1`, DB area automatic extension has occurred. To prevent DB area automatic extension from occurring and causing degradation of update processing, change the data DB area file, dictionary DB area file, or system-table DB area file from a regular file to a block special file. When changing any of these files to a block special file, use one of the methods described in the following to determine the size of the target DB area, and then prepare a block special file with the required size allocated.

- **Estimating the size of the target DB area**
  Determine the size of the data DB area based on the explanation in 5.8 Estimating the size of the data DB area.
  Determine the size of the dictionary DB area based on the explanation in 5.11 Estimating the size of the dictionary DB area.
  Determine the size of the system-table DB area based on the explanation in 5.12 Estimating the size of the system-table DB area.
- **Executing the `adbdbstatus` command**
  Determine the usage in the target DB area based on the explanation in *Checking the usage of the entire database (output of summary information for DB areas)* in *Examples* under *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.

After preparing a block special file of the necessary size, change the target DB area file to a block special file. For details about how to change the target DB area file to a block special file, see (1) Changing a data DB area file from a regular file to a block special file in 11.10.5 Changing the storage location of data DB area files.

13.1.11 Changing the file configuration of the work table DB area

This subsection explains how to prevent degradation in the processing performance of SQL statements caused by automatic extension of the work table DB area.

If a work table DB area file consists of regular files, a work table DB area with the smallest size is created when the `adbinit` command is executed. After that, the necessary size is allocated to the work table DB area through DB area automatic extension when an SQL statement that creates work tables is executed.

If automatic extension of the work table DB area occurs, it impacts the processing performance of an SQL statement that creates work tables. To prevent automatic extension from degrading processing performance of SQL statements, consider using block special files to configure the work table DB area files.

Check the following item in the HADB server statistical information that is output when the `adbstat` command is executed:

- `Wrktbl_page_use_max` (maximum number of work table pages allocated)

Substitute the value that is output to `Wrktbl_page_use_max` for the *WRK_PAGE_NUM* variable in the formula provided in 5.9 Estimating the size of the work table DB area to estimate the size of the work table DB area. Based on the estimated size, prepare a block special file with the necessary size allocated.

After preparing a block special file of the necessary size, change the work table DB area file to a block special file. For details about how to change the work table DB area file to a block special file, see 11.11.1 Changing the storage location of a work table DB area file.

13.2 Tuning to shorten SQL statement execution time by re-examining the buffers

This section explains how to improve processing performance by re-examining the values specified for the following buffer-related operands:

- `adbbuff` operand in the server definition
- `adb_dbbuff_wrktbl_glb_blk_num` operand in the server definition
- `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition

When you modify the values specified for these operands, be sure to re-estimate the memory requirements.

13.2.1 Points to be checked before performing tuning

This subsection describes the points to be checked before performing the tuning to reduce the execution time of SQL statements and commands. Please review the following two points:

Revise the data DB areas that store base tables and indexes

Make sure that the following conditions are met:

- One data DB area stores only one base table.

  For details about how to check the base table (or base tables) stored in a data DB area, see (14) When identifying the table name of a table stored in a DB area in B.22 Searching a dictionary table.

- One data DB area stores only one index.

  For details about how to check the index (or indexes) stored in a data DB area, see (15) When identifying the index name of an index stored in a DB area in B.22 Searching a dictionary table.

As described in 5.6.1 Points to consider when designing a data DB area, if multiple base tables or indexes are stored in one data DB area, the resulting increase in I/O accesses might degrade performance. Especially, do not store both a row store table and column store table in one data DB area. To avoid this, make sure that no more than one base table or index is stored in a data DB area.

For details about how to revise the data DB areas that store base tables and indexes, see 11.1.14 Changing the data DB area that stores a base table and 11.3.8 Changing the data DB area that stores indexes.

Revise the global buffers to be allocated to data DB areas

Check whether one global buffer is exclusively allocated to one data DB area. To check the global buffer (or global buffers) allocated to a data DB area, use the `adbls -d gbuf` command. For details about the `adbls -d gbuf` command, see *adbls -d gbuf (Display Global Buffer Information)* in the manual *HADB Command Reference*.

Allocating one global buffer to multiple data DB areas might degrade performance. Therefore, make sure that each global buffer is allocated to only one data DB area. To revise the global buffers, check and, if necessary, revise each `adbbuff` operand specification in the server definition. For details about the `adbbuff` operand in the server definition, see 7.2.11 Operands and options related to global buffers (command format). For details about how to modify the server definition, see 8.5.2 Modifying the server definition.

13.2.2 Reducing the SQL statement execution time

By re-evaluating the value specified for the `adbbuff` operand, you might be able to reduce the SQL statement execution time.

The tuning method to use depends on the following information, output by the adbstat command:

- Global buffer statistical information
- SQL statement statistical information

First, check the global buffer statistical information and perform tuning. Afterward, if you want to reduce the execution time of a specific SQL statement, check the SQL statement's statistical information and perform tuning again.

(1) Using the global buffer statistical information to reduce SQL statement execution time

Procedure

1. Check the global buffer hit rate.
   Check the value of the following item in the global buffer statistical information that is output when the adbstat command is executed:
   - DBbuff_page_hit_rate (global buffer page hit rate)
2. Reassess the global buffer allocation method.
   If the DBbuff_page_hit_rate value is 50 or smaller, consider the following three actions:
   - **Increase the number of pages in global buffers.**
     The number of pages in the global buffers allocated to DB areas might be too small. If so, increase the value specified for the -p option of the adbbuff operand.
   - **Allocate a dedicated global buffer.**
     If a single global buffer is allocated to multiple DB areas, use the adbbuff operand to allocate a dedicated global buffer to each DB area.
   - **Allocate a table scan buffer.**
     If there is an SQL statement that performs table scans, specify the -v option in the adbbuff operand to allocate a table scan buffer.
3. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
4. Change the server definition.
   Change the specification of the adbbuff operand in the server definition based on the policies determined in step 2.
5. Start the HADB server.
   Execute the adbstart command to start the HADB server.

> **Note**
>
> For details about the adbbuff operand in the server definition, see the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

(2) Using the SQL statement statistical information to reduce SQL statement execution time

Procedure

1. Check the global buffer hit rate.

   Check the value of the following item in the SQL statement statistical information that was output when the `adbstat` command was executed:

   - `DBbuff_page_hit_rate` (global buffer page hit rate)

2. Reassess the global buffer allocation method.

   If the `DBbuff_page_hit_rate` value, which you checked in step 1, is `50` or smaller, consider taking one or more of the following three actions, as appropriate:

   - **Increase the number of pages in global buffers.**

     There might be too few pages in the global buffers allocated to DB areas. If so, increase the value specified for the `-p` option of the `adbbuff` operand.

   - **Allocate a dedicated global buffer.**

     If a single global buffer is allocated to multiple DB areas, check `DBbuff_page_request_cnt` (number of global buffer page requests) in the SQL statement statistical information. If the number of page requests is large for any DB area, use the `adbbuff` operand to allocate a dedicated global buffer to that DB area.

   - **Allocate a table scan buffer.**

     If there is an SQL statement that performs table scans, specify the `-v` option in the `adbbuff` operand to allocate a table scan buffer.

3. Terminate the HADB server.

   Terminate the HADB server by executing the `adbstop` command.

4. Change the server definition.

   Change the specification of the `adbbuff` operand in the server definition based on the policies determined in step 2.

5. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the `adbbuff` operand in the server definition, see the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

13.2.3 Reducing the execution time of an SQL statement that creates a global work table

When you execute an SQL statement that creates a global work table, you can sometimes reduce its execution time by increasing the number of pages in the global buffer for global work tables.

For details about SQL statements that create global work tables, see *Work tables created when SQL statements are executed* under *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

The tuning method to use depends on the following information, output by the `adbstat` command:

- Global buffer statistical information
- SQL statement statistical information

First, check the global buffer statistical information and perform tuning. Afterward, if you want to reduce the execution time of a specific SQL statement, check the SQL statement's statistical information and perform tuning again.

(1) Using the global buffer statistical information to reduce the execution time of an SQL statement that creates a global work table

Procedure

1. Check the number of times files are written from the global buffer.
   In the global buffer statistical information that is output when the `adbstat` command is executed, check the following types of information first:
   - `DBbuff_name` (global buffer name)

   For the global buffer whose `DBbuff_name` is `ADBWRK`, check the following information:
   - `DBbuff_page_write_cnt` (number of times files are written from the global buffer)

   If the value that is output to `DBbuff_page_write_cnt` is at least 1, data has been written from the global buffer to files. If this case applies, go to step 2.
2. Terminate the HADB server.
   Terminate the HADB server by executing the `adbstop` command.
3. Change the server definition.
   Increase the value specified for the `adb_dbbuff_wrktbl_glb_blk_num` operand in the server definition.
4. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.
5. Re-execute the SQL statement that creates a global work table.
   After re-executing the SQL statement that creates a global work table, perform step 1 to check whether the value that was output to `DBbuff_page_write_cnt` has decreased.
6. Repeat steps 1 to 5.
   If you make adjustments by repeating steps 1 to 5 until the value that is output to `DBbuff_page_write_cnt` becomes `0`, you can sometimes reduce the execution time of the SQL statement that creates a global work table.

> **Note**
>
> For details about the `adb_dbbuff_wrktbl_glb_blk_num` operand in the server definition, see the adb_dbbuff_wrktbl_glb_blk_num operand in 7.2.2 Operands related to performance (set format).

(2) Using the SQL statement statistical information to reduce the execution time of an SQL statement that creates a global work table

Procedure

1. Check the number of times files are written from the global buffer.
   Check the values of the following items in the SQL statement statistical information that was output when the `adbstat` command was executed:
   - `DBbuff_name` (global buffer name)

   For the global buffer whose `DBbuff_name` is `ADBWRK`, check the following information:

- DBbuff_page_write_cnt (number of files written from the global buffer)
- DBbuff_page_put_cnt (number of files written to the global buffer)

2. Check whether a write from the global buffer to a file has occurred frequently.

   If the relationship between the DBbuff_page_write_cnt value and DBbuff_page_put_cnt value, which you checked in step 1, satisfies the following formula, it is possible that files are being written from the global buffer frequently.

   Formula

   ```
   value output to DBbuff_page_write_cnt > value output to DBbuff_page_put_cnt ÷ 2
   ```

   If the preceding formula is satisfied, go to step 3.

3. Terminate the HADB server.

   Terminate the HADB server by executing the adbstop command.

4. Change the server definition.

   Increase the value specified for the adb_dbbuff_wrktbl_glb_blk_num operand in the server definition.

5. Start the HADB server.

   Execute the adbstart command to start the HADB server.

6. Re-execute the SQL statement that creates a global work table.

   After re-executing the SQL statement that creates a global work table, perform step 1 to confirm that the value that was output to DBbuff_page_write_cnt does not satisfy the formula.

7. Repeat steps 1 to 6.

   If you make adjustments by repeating steps 1 to 6 until the value that is output to DBbuff_page_write_cnt no longer satisfies the formula, you can sometimes reduce the execution time of the SQL statement that creates a global work table.

> **Note**
>
> For details about the adb_dbbuff_wrktbl_glb_blk_num operand in the server definition, see the adb_dbbuff_wrktbl_glb_blk_num operand in 7.2.2 Operands related to performance (set format).

13.2.4 Reducing the execution time of an SQL statement that creates a local work table

When you execute an SQL statement that creates a local work table, you can sometimes reduce its execution time by increasing the number of pages in the buffer for local work tables.

For details about SQL statements that create local work tables, see *Work tables created when SQL statements are executed* under *Considerations when executing an SQL statement that creates work tables* in *Designs Related to Improvement of Application Program Performance* in the *HADB Application Development Guide*.

The tuning method to use depends on the following information, output by the adbstat command:

- Connection operation information
- SQL statement statistical information

First, check the connection operation information and perform tuning. Afterward, if you want to reduce the execution time of a specific SQL statement, check the SQL statement's statistical information and perform tuning again.

> **Note**
>
> If the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition is not specified for a connection, you can use the `adbmodbuff` command while the HADB server is running to change the number of pages in local work table buffers that was specified in the `adb_dbbuff_wrktbl_clt_blk_num` operand in the server definition. See 11.12.1 Changing the local work table buffer.

(1) Using the connection operation information to reduce the execution time of an SQL statement that creates a local work table

Procedure

1. Check the number of times files are written from the buffer for local work tables.

   In the connection operation information that is output when the `adbstat` command is executed, check the following information:

   - `DBbuff_wrktbl_clt_write_cnt` (number of times files are written from the buffer for local work tables)

2. Change the client definition.

   If the value that is output to `DBbuff_wrktbl_clt_write_cnt`, which you checked in step 1, is at least 1, files have been written from the buffer for local work tables. Increase the value specified for the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition.

   When you change the client definition, follow the procedure explained in *Notes about changing a client definition* in the *HADB Application Development Guide*.

3. Re-execute the SQL statement that creates a local work table.

   After re-executing the SQL statement that creates a local work table, perform step 1 to check whether the value that was output to `DBbuff_wrktbl_clt_write_cnt` has decreased.

4. Repeat steps 1 to 3.

   If you make adjustments by repeating steps 1 to 3 until the value that is output to `DBbuff_wrktbl_clt_write_cnt` becomes `0`, you can sometimes reduce the execution time of the SQL statement that creates a local work table.

> **Note**
>
> For details about the `adb_dbbuff_wrktbl_clt_blk_num` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

(2) Using the SQL statement statistical information to reduce the execution time of an SQL statement that creates a local work table

Procedure

1. Check the number of times files are written from the local work table buffer.

   Check the values of the following items in the SQL statement statistical information that was output when the `adbstat` command was executed:

- DBbuff_wrktbl_clt_write_cnt (number of files written from the local work table buffer)
- DBbuff_wrktbl_clt_put_cnt (number of files written to the local work table buffer)

2. Change the client definition.

   If the relationship between the DBbuff_wrktbl_clt_write_cnt value and DBbuff_wrktbl_clt_put_cnt value, which you checked in step 1, satisfies the following formula, it is possible that files are being written from the local work table buffer frequently.

   Formula

   ```
   value output to DBbuff_wrktbl_clt_write_cnt > value output to DBbuff_wrktbl_clt_put_cnt ÷ 2
   ```

   If the preceding formula is satisfied, increase the value specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the client definition.

   When you change the client definition, follow the procedure explained in *Notes about changing a client definition* in the *HADB Application Development Guide*.

3. Re-execute the SQL statement that creates a local work table.

   After re-executing the SQL statement that creates a local work table, perform step 1 to confirm that the value that was output to DBbuff_wrktbl_clt_write_cnt does not satisfy the formula.

4. Repeat steps 1 to 3.

   If you make adjustments by repeating steps 1 to 3 until the value that is output to DBbuff_wrktbl_clt_write_cnt no longer satisfies the formula, you can sometimes reduce the execution time of the SQL statement that creates a local work table.

> **Note**
>
> For details about the adb_dbbuff_wrktbl_clt_blk_num operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

13.2.5 Reducing the execution time of SQL statements that perform table scans

If the adbbuff operand has been specified with the -v option, you might be able to reduce the execution time of SQL statements that perform table scans by adjusting the value specified for the -v option.

The tuning method to use depends on the following information, output by the adbstat command:

- Global buffer statistical information
- SQL statement statistical information

First, check the global buffer statistical information and perform tuning. Afterward, if you want to reduce the execution time of a specific SQL statement, check the SQL statement's statistical information and perform tuning again.

(1) Using the global buffer statistical information to reduce the execution time of an SQL statement that performs table scans

Procedure

1. Check the number of times a buffer shortage occurred in the table scan buffer, and the table scan buffer's page hit rate.
   Check the value of the following item in the global buffer statistical information that is output when the `adbstat` command is executed:
   - `DBbuff_tblscan_failed_cnt` (number of times a buffer shortage occurred in the table scan buffer)
   - `DBbuff_tblscan_hit_rate` (page hit rate in the table scan buffer)

   If the value that is output to `DBbuff_tblscan_failed_cnt` is at least `1` and the value that is output to `DBbuff_tblscan_hit_rate` is low, a page shortage has occurred in the table scan buffer. Consequently, data was read into the global buffer during the table scan. If this case applies, go to step 2.
2. Terminate the HADB server.
   Terminate the HADB server by executing the `adbstop` command.
3. Change the server definition.
   To the value specified for the `-v` option of the `adbbuff` operand in the server definition, add the value specified for the `adb_sys_rthd_num` operand in the server definition.
4. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.
5. Re-execute the SQL statement that performs a table scan.
   After re-executing the SQL statement that performs a table scan, perform step 1 to check whether the value output to `DBbuff_tblscan_failed_cnt` has decreased. Also, check whether the value output to `DBbuff_tblscan_hit_rate` has increased.
6. Repeat steps 1 to 5.
   If you make adjustments by repeating steps 1 to 5 until the value that is output to `DBbuff_tblscan_failed_cnt` approaches `0`, the value that is output to `DBbuff_tblscan_hit_rate` increases, improving the page hit rate in the table scan buffer. As a result, you can sometimes reduce the execution time of an SQL statement that performs table scans.

> **Note**
>
> For details about the `adbbuff` operand in the server definition, see the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

(2) Using the SQL statement statistical information to reduce the execution time of an SQL statement that performs table scans

Procedure

1. Check the number of times a buffer shortage occurred in the table scan buffer.
   Check the values of the following items in the SQL statement statistical information that was output when the `adbstat` command was executed:
   - `DBbuff_tblscan_failed_cnt` (number of times a buffer shortage occurred in the table scan buffer)
   - `DBbuff_tblscan_insufficient_buff_num` (number of table scan buffer sectors in which a shortage occurred)

   If the value that was output to `DBbuff_tblscan_failed_cnt` is at least `1`, a table scan buffer shortage has occurred. Consequently, data was read into the global buffer during the table scan. If this case applies, go to step 2.

2. Terminate the HADB server.

   Terminate the HADB server by executing the `adbstop` command.

3. Change the server definition.

   To the value specified for the `-v` option of the `adbbuff` operand in the server definition, add the value that was output to `DBbuff_tblscan_insufficient_buff_num`.

4. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

5. Re-execute the SQL statement that performs a table scan.

   By making these adjustments, you can sometimes reduce the execution time of an SQL statement that performs table scans.

Note

For details about the `adbbuff` operand in the server definition, see the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

13.3 Tuning to shorten command execution time

This section explains how to shorten command execution time.

> **Note**
>
> Before you perform the tuning to shorten the command execution time, check and, if necessary, revise the data DB areas and global buffers. For details, see 13.2.1 Points to be checked before performing tuning.

13.3.1 Reducing the adbimport command's execution time

If you specify appropriate values for the `adbimport` command options that are related to import performance, the import performance might improve and the `adbimport` command's execution time might be reduced.

For details about the specification format for the `adbimport` command's import options, see *Format of import options* under *Specification format for the adbimport command* in *adbimport (Import Data)* in the manual *HADB Command Reference*.

13.3.2 Reducing the execution time of the adbidxrebuild command

This subsection explains how to reduce the execution time of the `adbidxrebuild` command.

Specify the `-v` option for the `adbbuff` operand in the server definition before you execute the `adbidxrebuild` command. Specifying the option might be able to reduce the execution time of the `adbidxrebuild` command. If the `-v` option has already been specified, you might be able to reduce the execution time of the `adbidxrebuild` command by revising the value specified for the option.

Procedure

1. Estimate the value to be specified for the `-v` option.

   Estimate the value to be specified for the -v option of the adbbuff server definition operand in 7.2.11 Operands and options related to global buffers (command format). At this time, substitute the value obtained from the following expression for the *sql_rthd_num* variable, rather than the value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition:

   ```
   ↓(value-specified-for-index-rebuild-option-adb_idxrebuild_rthd_num - 1) ÷ 2↓
   ```

2. Terminate the HADB server.

   Terminate the HADB server by executing the `adbstop` command.

3. Change the server definition.

   Add the `-v` option to the `adbbuff` operand in the server definition. Alternatively, change the value specified for the `-v` option. For the `-v` option, specify the value that you estimated in step 1.

4. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

5. Execute the `adbidxrebuild` command.

6. Execute the `adbstat` command to output the global buffer statistical information.

Make adjustments until the value that is output to DBbuff_tblscan_failed_cnt in the global buffer statistical information output when the adbstat command is executed approaches 0, based on the explanation in (1) Using the global buffer statistical information to reduce the execution time of an SQL statement that performs table scans under 13.2.5 Reducing the execution time of SQL statements that perform table scans. By making these adjustments, you can sometimes reduce the execution time of the adbidxrebuild command.

When you read the description in the preceding reference-target subsection, replace *an SQL statement that performs table scans* with *the adbidxrebuild command*.

13.3.3 Reducing the execution time of the adbgetcst command

This subsection explains how to reduce the execution time of the adbgetcst command.

Specify the -v option for the adbbuff operand in the server definition before you execute the adbgetcst command. Specifying this option might reduce the execution time of the command. If the -v option has already been specified, you might be able to reduce the execution time of the adbgetcst command by revising the value specified for the option.

Procedure

1. Estimate the value to be specified for the -v option.

   Estimate the value to be specified for the -v option of the adbbuff server definition operand in 7.2.11 Operands and options related to global buffers (command format). At this time, substitute the value obtained from the following expression for the *sql_rthd_num* variable, rather than the value specified for the adb_sql_exe_max_rthd_num operand in the server definition:

   ```
   value-specified-for-cost-information-collection-option-adb_getcst_rthd_num - 1
   ```

2. Terminate the HADB server.

   Terminate the HADB server by executing the adbstop command.

3. Change the server definition.

   Add the -v option to the adbbuff operand in the server definition. Alternatively, change the value specified for the -v option. For the -v option, specify the value that you estimated in step 1.

4. Start the HADB server.

   Execute the adbstart command to start the HADB server.

5. Execute the adbgetcst command.

6. Execute the adbstat command to output the global buffer statistical information.

   Make adjustments until the value that is output to DBbuff_tblscan_failed_cnt in the global buffer statistical information output when the adbstat command is executed approaches 0, based on the explanation in (1) Using the global buffer statistical information to reduce the execution time of an SQL statement that performs table scans under 13.2.5 Reducing the execution time of SQL statements that perform table scans. By making these adjustments, you can sometimes reduce the execution time of the adbgetcst command.

   When you read the description in the preceding reference-target subsection, replace *an SQL statement that performs table scans* with *the adbgetcst command*.

13.3.4 Reducing the execution time of the adbexport command

This subsection explains how to reduce the execution time of the adbexport command.

Before you execute the adbexport command to export the retrieval result of an SQL statement, read the following subsections. By using the methods described in these subsections, you might be able to reduce the execution time of the adbexport command.

- 13.2.2 Reducing the SQL statement execution time
- 13.2.3 Reducing the execution time of an SQL statement that creates a global work table
- 13.2.4 Reducing the execution time of an SQL statement that creates a local work table

  In the subsections listed above, substitute *the value specified for the adb_export_wrktbl_blk_num export option* for *the value specified for the adb_dbbuff_wrktbl_clt_blk_num operand in the server definition*. Also, substitute *the adbexport command* for *an SQL statement that creates a local work table*.
- 13.5 Tuning to shorten SQL statement execution time by re-examining the hash table area size
- 13.8 Tuning to shorten SQL statement execution time by re-examining the hash filter area size

Perform tuning based on the explanation in the above subsections.

Additionally, you can sometimes reduce the execution time of the adbexport command by specifying the -v option of the adbbuff operand in the server definition. If the -v option has already been specified, you might be able to reduce the execution time of the adbexport command by revising the value specified for the option.

Procedure

1. Estimate the value to be specified for the -v option.

   Estimate the value to be specified for the -v option of the adbbuff server definition operand in 7.2.11 Operands and options related to global buffers (command format). At this time, substitute the value obtained from the following expression for the *sql_rthd_num* variable, rather than the value specified for the adb_sql_exe_max_rthd_num operand in the server definition:

   ```
   ↓(value-specified-for-export-option-adb_export_rthd_num - 1) ÷ 2↓
   ```
2. Terminate the HADB server.

   Terminate the HADB server by executing the adbstop command.
3. Change the server definition.

   Add the -v option to the adbbuff operand in the server definition. Alternatively, change the value specified for the -v option. For the -v option, specify the value that you estimated in step 1.
4. Start the HADB server.

   Execute the adbstart command to start the HADB server.
5. Execute the adbexport command.
6. Execute the adbstat command to output the global buffer statistical information.

   Make adjustments until the value that is output to DBbuff_tblscan_failed_cnt in the global buffer statistical information output when the adbstat command is executed approaches 0, based on the explanation in (1) Using the global buffer statistical information to reduce the execution time of an SQL statement that performs table scans under 13.2.5 Reducing the execution time of SQL statements that perform table scans. By making these adjustments, you can sometimes reduce the execution time of the adbexport command.

   When you read the description in the preceding reference-target subsection, replace *an SQL statement that performs table scans* with *the adbexport command*.

13.3.5 Reducing the execution time of the adbmergechunk command

This subsection explains how to reduce the execution time of the adbmergechunk command.

Specify the -v option for the adbbuff operand in the server definition before you execute the adbmergechunk command. Specifying this option might reduce the execution time of the adbmergechunk command. If the -v option has already been specified, you might be able to reduce the execution time of the adbmergechunk command by revising the value specified for the option.

Procedure

1. Estimate the value to be specified for the -v option.

   Estimate the value to be specified for the -v option of the adbbuff server definition operand in 7.2.11 Operands and options related to global buffers (command format). At this time, substitute the value obtained from the following expression for the *sql_rthd_num* variable, rather than the value specified for the adb_sql_exe_max_rthd_num operand in the server definition:

   ```
   ↓(value-specified-for-merge-chunk-option-adb_mergechunk_rthd_num - 1) ÷ 2↓
   ```

2. Terminate the HADB server.

   Terminate the HADB server by executing the adbstop command.

3. Change the server definition.

   Add the -v option to the adbbuff operand in the server definition. Alternatively, change the value specified for the -v option. For the -v option, specify the value that you estimated in step 1.

4. Start the HADB server.

   Execute the adbstart command to start the HADB server.

5. Execute the adbmergechunk command.

6. Execute the adbstat command to output the global buffer statistical information.

   Make adjustments until the value that is output to DBbuff_tblscan_failed_cnt in the global buffer statistical information output when the adbstat command is executed approaches 0, based on the explanation in (1) Using the global buffer statistical information to reduce the execution time of an SQL statement that performs table scans under 13.2.5 Reducing the execution time of SQL statements that perform table scans. By making these adjustments, you can sometimes reduce the execution time of the adbmergechunk command.

   When you read the description in the preceding reference-target subsection, replace *an SQL statement that performs table scans* with *the adbmergechunk command*.

13.3.6 Reducing the execution time of the adbarchivechunk command

This subsection describes how to reduce the execution time of the adbarchivechunk command.

Specify the -v option for the adbbuff operand in the server definition before you execute the adbarchivechunk command. Specifying this option might reduce the execution time of the adbarchivechunk command. If the -v option has already been specified, you might be able to reduce the execution time of the adbarchivechunk command by revising the value specified for the option.

Procedure:

1. Estimate the value to be specified for the -v option.

Estimate the value to be specified for the -v option of the adbbuff server definition operand in 7.2.11 Operands and options related to global buffers (command format). At this time, substitute the value obtained from the following expression for the *sql_rthd_num* variable, rather than the value specified for the `adb_sql_exe_max_rthd_num` operand in the server definition:

```
↓(value-specified-for-archive-chunk-option-adb_arcv_rthd_num - 1) ÷ 2↓
```

2. Terminate the HADB server.

   Terminate the HADB server by executing the `adbstop` command.

3. Change the server definition.

   Add the `-v` option to the `adbbuff` operand in the server definition. Alternatively, change the value specified for the `-v` option. For the `-v` option, specify the value that you estimated in step 1.

4. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

5. Execute the `adbarchivechunk` command.

6. Execute the `adbstat` command to output the global buffer statistical information.

   Make adjustments until the value that is output to `DBbuff_tblscan_failed_cnt` in the global buffer statistical information output when the `adbstat` command is executed approaches `0`, based on the explanation in (1) Using the global buffer statistical information to reduce the execution time of an SQL statement that performs table scans under 13.2.5 Reducing the execution time of SQL statements that perform table scans. By making these adjustments, you can sometimes reduce the execution time of the `adbarchivechunk` command.

   When you read the description in the preceding reference-target subsection, replace *an SQL statement that performs table scans* with *the adbarchivechunk command*.

13.4 Tuning to reduce memory usage

This section explains how to reduce the HADB server's memory usage by checking the statistical information output by the `adbstat` command and the SQL trace information output to SQL trace files.

13.4.1 Reducing the usage of shared memory to which HugePages is applied

When `HugePages` is applied to the HADB server's shared memory, an unnecessarily large amount of memory might be allocated. As a result, performance of the Linux kernel, for example, might decline.

Whenever `HugePages` is applied to the HADB server's shared memory, check whether the allocated amount of memory is appropriate. The following procedure explains how to perform tuning:

Procedure

1. Check the maximum usage of all memory.
   Check the following item in the HADB server statistical information that is output when the `adbstat` command is executed:
   - `Total_memory_max_size` (maximum usage of all memory)
2. Check the kernel parameter.
   Check the value specified for the `vm.nr_hugepages` kernel parameter that was set in the operating system of the machine where the HADB server was installed.
3. Check whether the allocated amount of memory is appropriate.
   If the value specified for `vm.nr_hugepages` (step 2) is greater than the value output to `Total_memory_max_size` (step 1), an unnecessarily large amount of memory has been allocated. Reduce the value specified for `vm.nr_hugepages` to match the value output to `Total_memory_max_size`.

   > **Important**
   >
   > After you terminate the HADB server by executing the `adbstop` command, change the value specified for `vm.nr_hugepages`.

4. Restart the OS.
5. Change the server definition.
   If you changed the value specified for `vm.nr_hugepages` in step 3, change also the value of the `adb_sys_memory_limit` operand in the server definition.
6. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the `adb_sys_memory_limit` operand in the server definition, see the adb_sys_memory_limit operand in 7.2.2 Operands related to performance (set format).

13.4.2 Reducing memory usage by re-evaluating the global buffers

You can reduce the HADB server's memory usage by, for example, canceling the allocation of unused global buffers and re-evaluating the number of pages specified for global buffers.

(1) Canceling the allocation of unused global buffers

The following describes the tuning method to be used when allocating global buffers dedicated to range indexes and table scan buffers.

■ If the `-a` option is specified for the `adbbuff` operand

1. Check the number of range index buffer page requests.

   In the global buffer statistical information that is output when the `adbstat` command is executed, check whether `0` is output for the following item:

   - `DBbuff_page_rng_request_cnt` (number of range index buffer page requests)

   If the value output to `DBbuff_page_rng_request_cnt` is `0`, no range indexes might have been defined or a global buffer dedicated to range indexes might be allocated to a DB area that does not use range indexes.

   In such a case, delete the `-a` option from the `adbbuff` operand. This might reduce the amount of memory that the HADB server uses.

2. Terminate the HADB server.

   Terminate the HADB server by executing the `adbstop` command.

3. Change the server definition.

   Delete the `-a` option from the `adbbuff` operand in the server definition.

4. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

■ If the `-v` option is specified for the `adbbuff` operand

1. Check the number of table scan buffer page requests.

   In the global buffer statistical information that is output when the `adbstat` command is executed, check whether `0` is output for the following item:

   - `DBbuff_tblscan_request_cnt` (number of table scan buffer page requests)

   If the value output to `DBbuff_tblscan_request_cnt` is `0`, a table scan buffer might have been allocated to a DB area in which an SQL statement that performs a table scan is not executed.

   In such a case, delete the `-v` option from the `adbbuff` operand. This might reduce the amount of memory that the HADB server uses.

2. Terminate the HADB server.

   Terminate the HADB server by executing the `adbstop` command.

3. Change the server definition.

   Delete the `-v` option from the `adbbuff` operand in the server definition.

4. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

Note

For details about the adbbuff operand in the server definition, see the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

(2) Re-evaluating the number of pages specified for global buffers

The following describes the tuning method to be used when the number of pages specified for global buffers is too large.

■ If the -p option is specified for the adbbuff operand

1. Check the number of global buffers.
   In the global buffer statistical information that is output when the adbstat command is executed, check the values specified for the following two items:
   - DBbuff_page_use_cnt (number of allocated global buffers)
   - DBbuff_page_num (total number of global buffers)

   If the value output to DBbuff_page_use_cnt is smaller than the value output to DBbuff_page_num, the value specified for the -p option of the adbbuff operand might be too large. If this case applies, go to step 2.
2. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
3. Change the server definition.
   Change the value specified for the -p option of the adbbuff operand in the server definition to the value determined by using the following formula. This might reduce the amount of memory that the HADB server uses.

   Value output to DBbuff_page_use_cnt × 1.2
4. Start the HADB server.
   Execute the adbstart command to start the HADB server.

■ If the -a option is specified for the adbbuff operand

1. Check the number of range index buffers.
   In the global buffer statistical information that is output when the adbstat command is executed, check the values specified for the following two items:
   - DBbuff_page_rng_use_cnt (number of allocated range index buffers)
   - DBbuff_page_rng_num (total number of range index buffers)

   If the value output to DBbuff_page_rng_use_cnt is smaller than the value output to DBbuff_page_rng_num, the value specified for the -a option of the adbbuff operand might be too large. If this case applies, go to step 2.
2. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
3. Change the server definition.
   Change the value specified for the -a option of the adbbuff operand in the server definition to the value determined by using the following formula. This might reduce the amount of memory that the HADB server uses.

Value output to DBbuff_page_rng_use_cnt × 1.2

4. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the `adbbuff` operand in the server definition, see the adbbuff operand in 7.2.11 Operands and options related to global buffers (command format).

13.4.3 Reducing memory usage by re-evaluating buffers for local work tables

This subsection explains how to re-evaluate the number of pages in the buffers for local work tables by checking the access path statistical information that is output as SQL trace information.

You might be able to reduce memory usage by re-evaluating the number of pages in the buffers for local work tables.

Procedure

1. Check the access path statistical information.
   Check the access path statistical information that is output as SQL trace information. In the data access information that is output as access path statistical information, check the following output items:
   - `Data_dbbuff_wrktbl_clt_write_cnt` (number of write operations to files from buffers for local work tables)
   - `Data_wrktbl_page_use_max` (number of pages in the largest work table that was allocated)

   After you have checked the output items, go to step 2.
2. Check the values that have been specified for operands.
   Check the values that have specified for the following operands:
   - `adb_sql_exe_max_rthd_num`
   - `adb_dbbuff_wrktbl_clt_blk_num`

   After you have checked the operand values, go to step 3.
3. Determine whether the buffers for local work tables need to be re-evaluated.
   Based on the checked values, determine whether the condition shown below is satisfied.
   Condition (determining whether the buffers for local work tables need to be re-evaluated)

   A = 0, and *B* ÷ *C* > *D*

   Explanation of variables:
   A: Value of `Data_dbbuff_wrktbl_clt_write_cnt` in the access path statistical information
   B: Value of `Data_wrktbl_page_use_max` in the access path statistical information
   C: Value of the `adb_sql_exe_max_rthd_num` operand
   D: Value of the `adb_dbbuff_wrktbl_clt_blk_num` operand

   If the preceding formula is satisfied, the number of allocated pages of buffers for local work tables might be greater than the number of pages that are actually needed. If this case applies, go to step 4.
4. Change the client definition.

Change the value of the adb_dbbuff_wrktbl_clt_blk_num operand in the client definition for the relevant connection to the value obtained from the following formula.

Formula (adb_dbbuff_wrktbl_clt_blk_num operand value)

$A \div B \times 1.5$

Explanation of variables:
A: Value of Data_wrktbl_page_use_max in the access path statistical information
B: Value of the adb_sql_exe_max_rthd_num operand

By changing the value of the adb_dbbuff_wrktbl_clt_blk_num operand in the client definition for the relevant connection to the value determined from the preceding formula, you can reduce the memory usage without affecting processing performance.

When you change the client definition, follow the procedure explained in *Notes about changing a client definition* in the *HADB Application Development Guide*.

Note

For details about the adb_dbbuff_wrktbl_clt_blk_num operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

13.5 Tuning to shorten SQL statement execution time by re-examining the hash table area size

This section describes how to shorten SQL statement execution time by re-examining the hash table area size specified for the following operands:

- `adb_sql_exe_hashtbl_area_size` operand in the server definition
- `adb_sql_exe_hashtbl_area_size` operand in the client definition

Note that you can also shorten SQL statement execution time by specifying an SQL statement in the `adbexport` command. If you use this method, replace the `adb_sql_exe_hashtbl_area_size` operand that appears in this section with the `adb_export_hashtbl_area_size` export option.

When executing an SQL statement for which the following processes (which use the hash table area) are applied, you can possibly shorten SQL statement execution time by increasing the hash table area size.

- Hash join as a table joining method
- Global hash grouping as a grouping method
- Hash execution as a method for processing subqueries
- Hash execution as a method for processing `SELECT DISTINCT`
- Hash execution as a method for processing the set operation

Procedure:

1. Check whether there was a process in which the hash table area size became insufficient.

   Check the access path statistical information output to the SQL trace information. In the information related to hash table areas that is output to the access path statistical information, check whether there is a process for which the value for the following output item is `Y`:

   - `Hashtbl_area_shortage` (whether the hash table area size became insufficient)

   If there is a process for which the value for the preceding item is `Y`, go to step 2.

2. Check whether the `KFAA51130-W` message was output.

   In step 1, if there was a process for which the value for the relevant item is `Y`, compare the following two contents:

   - Content of `message_log_info` output to the SQL statement execution information for the relevant processing
   - Content of the server message log file

   After comparing the preceding two contents, check whether the `KFAA51130-W` message was output to the server message log file. If the `KFAA51130-W` message was output, go to step 3.

3. Re-examine the value specified for the `adb_sql_exe_hashtbl_area_size` operand.

   In step 2, if the `KFAA51130-W` message was output, the data that could not be processed in the hash table area might have been stored in the work table multiple times and processed. Therefore, the SQL statement execution time might be longer.

   In this case, increase the value specified for the `adb_sql_exe_hashtbl_area_size` operand in the server definition and client definition. The execution time of an SQL statement to which a process that uses the hash table area is applied might be shortened.

Important

- Check the values specified for both the `adb_sql_exe_hashtbl_area_size` operand in the client definition and the `adb_sql_exe_hashtbl_area_size` operand in the server definition. If the value specified in the client definition exceeds the value specified in the server definition as a result of increasing the value in the client definition only, the value specified in the server definition is applied. The new value specified in the client definition is not applied. If this case applies, increase the values of the `adb_sql_exe_hashtbl_area_size` operands in both the server definition and client definition.
- If you have changed the value specified for the `adb_sql_exe_hashtbl_area_size` operand, always re-estimate the HADB server's memory requirement. For details about how to re-estimate the memory requirements of the HADB server, see the explanation of the variable *HASHTBL_PROC_SIZE* in (c) Determining the variable RTHD_EXESQLSZ under (2) Determining the real thread private memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

4. Change the server definition and client definition.

   Increase the value specified for the `adb_sql_exe_hashtbl_area_size` operand in the server definition and client definition.

   - When you change the server definition:

     Use the `adbstop` command to terminate the HADB server temporarily. After that, change the server definition, and then use the `adbstart` command to start the HADB server.

   - When you change the client definition:

     Change the client definition by following the procedure explained in *Notes about changing a client definition* in the *HADB Application Development Guide*.

5. Re-execute the SQL statement.

   Re-execute the SQL statement, and check whether the execution time of the SQL statement is shortened.

Note

- For details about the information related to hash table areas that is output as access path statistical information, see (e) Information related to hash table areas in (2) Items that are output as access path statistical information under 10.11.3 Examples of output of and output items for access path statistical information.
- For details about the `adb_sql_exe_hashtbl_area_size` operand in the server definition, see the description of the adb_sql_exe_hashtbl_area_size operand in 7.2.2 Operands related to performance (set format).
- For details about the `adb_sql_exe_hashtbl_area_size` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

13.6 Tuning to shorten SQL statement execution time by re-examining the hash group area size

This section explains how to shorten SQL statement execution time by re-examining the hash group area size specified in the following operands:

- `adb_sql_exe_hashgrp_area_size` operand in the server definition
- `adb_sql_exe_hashgrp_area_size` operand in the client definition

If you execute SQL statements to which local hash grouping is applied, you might be able to shorten the SQL statement execution time by increasing the hash group area size.

Procedure

1. Check if a shortage of hash grouping area occurred during any processing.

   Check the access path statistical information that has been output as SQL trace information. In the information related to hash grouping areas that was output to access path statistical information, check if `Y` is displayed for the following output item:

   - `Hashgrp_area_shortage` (whether a space shortage occurred in the hash grouping area)

   If local hash grouping processing was performed and `Y` is displayed, go to step 2.

2. Re-evaluate the value of the `adb_sql_exe_hashgrp_area_size` operand.

   If step 1 revealed that there was a local hash grouping process that resulted in a shortage of hash grouping area, data that could not be processed in the hash grouping area might have been stored in a work table for processing. This might have caused an increase in the SQL statement execution time.

   In this case, check the following output item in the information related to hash grouping areas that was output as access path statistical information for all the local hash grouping processes whose `Hashgrp_area_shortage` is `Y`:

   - `Hashgrp_area_sufficient_size` (sufficient size (in kilobytes) of hash grouping area that will not result in a shortage of space)

   Then, in the server definition or client definition, set the `adb_sql_exe_hashgrp_area_size` operand to a value that is equal to or larger than the largest one of the values that have been output. This might shorten SQL statement execution time to which local hash grouping is applied.

   > **Important**
   >
   > If you have changed the value of the `adb_sql_exe_hashgrp_area_size` operands, always re-estimate the HADB server's memory requirement. For details about estimation of the HADB server's memory requirement, see 6.3 Estimating the HADB server's memory requirement.

3. Change the server definition or client definition.

   Increase the value specified for the `adb_sql_exe_hashgrp_area_size` operand in the server definition or client definition.

   - When you change the server definition:

     Use the `adbstop` command to terminate the HADB server temporarily. After that, change the server definition, and then use the `adbstart` command to start the HADB server.

   - When you change the client definition:

     Change the client definition by following the procedure explained in *Notes about changing a client definition* in the *HADB Application Development Guide*.

4. Re-execute the SQL statement.

   Re-execute the SQL statement, and check whether the execution time of the SQL statement is shortened.

Note

- For details about the information related to hash grouping areas that is output as access path statistical information, see (d) Information related to hash grouping areas in (2) Items that are output as access path statistical information under 10.11.3 Examples of output of and output items for access path statistical information.
- For details about the `adb_sql_exe_hashgrp_area_size` operand in the server definition, see the description of the adb_sql_exe_hashgrp_area_size operand in 7.2.2 Operands related to performance (set format).
- For details about the `adb_sql_exe_hashgrp_area_size` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

13.7 Tuning to shorten SQL statement execution time by re-examining the join order of INNER JOINs to which a hash join is applied

This section explains how to shorten SQL statement execution time by re-examining the join order of the INNER JOINs to which a hash join is applied.

If you execute SQL statements that contain INNER JOINs to which a hash join is applied, you might be able to shorten the SQL statement execution time by using a join method specification in which the outer tables and the inner tables of the INNER JOINs are exchanged.

Procedure

1. Check if there are INNER JOINs to which a hash join is applied.

   Check the executed SQL statement and access path information that is output as SQL trace information. Check if the access path information contains any INNER JOINs to which a hash join was applied.

   If there is an INNER JOIN to which a hash join was applied, go to step 2.

2. Check whether the hash join resulted in a shortage of space in the hash table area.

   If there is an INNER JOIN to which a hash join was applied, check the access path statistical information that has been output as SQL trace information for that hash join. In the information related to hash table areas that has been output as access path statistical information, check if Y is displayed for the following output item:

   - Hashtbl_area_shortage (whether a space shortage occurred in the hash table area)

   If Y is displayed, go to step 3.

3. Check the numbers of rows processed in the outer table and the inner table for the hash join.

   If a shortage of hash table area occurred, check the access path statistical information that has been output as SQL trace information. In the information related to retrieval processing that has been output as access path statistical information, check the following output item:

   - Scan_row_cnt (number of rows retrieved)

   Check if the number of rows processed in the outer table for the hash join is much larger than the number of rows processed in the inner table. If the former is much larger than the latter, go to step 4.

4. Re-examine the join order of the hash joins.

   In the INNER JOINs to which a hash join is applied, specify the join method specification in such a manner that the inner table whose processed row count was smaller becomes the outer table.

   If the number of rows processed in the outer table for a hash join is much larger than the number of rows processed in the inner table, the data in the outer table that could not be processed in the hash table area was stored temporarily in a work table. This might be the cause of the long SQL statement execution time.

   By using the join method specification, you might be able to shorten the execution time of SQL statements that contain INNER JOINs to which a hash join is applied.

Note

- For details about access path information, see (5) Executed SQL statement and access path information in 10.11.2 Information that is output as SQL trace information.
- For details about the information related to hash table areas that is output as access path statistical information, see (e) Information related to hash table areas in (2) Items that are output as access path statistical information under 10.11.3 Examples of output of and output items for access path statistical information.

- For details about the information related to retrieval processing that is output as access path statistical information, see (b) Information related to retrieval processing in (2) Items that are output as access path statistical information under 10.11.3 Examples of output of and output items for access path statistical information.
- For details about the join method specification, see *Join method specification* in *Constituent Elements* in the manual *HADB SQL Reference*.

13.8 Tuning to shorten SQL statement execution time by re-examining the hash filter area size

This section describes how to shorten SQL statement execution time by re-examining the hash filter area size specified for the following operands:

- `adb_sql_exe_hashflt_area_size` operand in the server definition
- `adb_sql_exe_hashflt_area_size` operand in the client definition

Note that you can also shorten SQL statement execution time by specifying an SQL statement in the `adbexport` command. If you use this method, replace the `adb_sql_exe_hashflt_area_size` operand that appears in this section with the `adb_export_hashflt_area_size` export option.

You might be able to shorten SQL statement execution time by increasing the size of the hash filter area if the following processes to which hash filters are applied are executed:

- Hash join as a table joining method
- Hash execution as a method for processing subqueries

Read the description in both of the following sections: ■ *Checking whether hash filters are applied* and ■ *Checking whether hash filters are working effectively*.

■ Checking whether hash filters are applied

Procedure

1. Check whether there are processes to which no hash filter is applied.

   In the output results of SQL trace information, check whether there are processes that satisfy the following conditions:

   - In the client-definition information, the value of `adb_sql_exe_hashflt_area_size` is `1` or larger.
   - In the SQL statement statistical information, the value of `Hashflt_disabled` (whether hash filters are disabled) is `Y`.

   If a process satisfies all of the preceding conditions, no hash filter is applied to the process. Proceed to step 2. If there is no process that satisfies all of the preceding conditions, read the description in ■ *Checking whether hash filters are working effectively*.

2. Check the total number of invalidated hash filters.

   If there are processes that satisfy the conditions shown in step 1, check the value of the following item in the output results of SQL trace information:

   - `Hashtbl_filter_disabled_num` in the information related to hash table areas that is output as access path statistical information (total number of invalidated hash filters)

3. Re-examine the value specified for the `adb_sql_exe_hashflt_area_size` operand.

   If the value of `Hashtbl_filter_disabled_num` that you checked in step 2 is `0`, hash filters might not be applied because the hash filter area is too small. Therefore, the SQL statement execution time might be longer.

   In this case, increase the values specified for the `adb_sql_exe_hashflt_area_size` operands in the server definition and client definition. This might cause hash filters to be applied, resulting in faster execution of the SQL statement. For the guideline to determine the value to be specified for the `adb_sql_exe_hashflt_area_size` operand, see the following sections in the *HADB Application Development Guide*:

   - *Conditions where a hash filter is applied* in *About hash join*

- *Hash execution* in *Methods for processing subqueries that do not contain an external reference column*
- *Hash execution* in *Methods for processing subqueries that contain an external reference column*

Important

- Check the values specified for both the `adb_sql_exe_hashflt_area_size` operand in the client definition and the `adb_sql_exe_hashflt_area_size` operand in the server definition. If the value specified in the client definition exceeds the value specified in the server definition as a result of increasing the value in the client definition only, the value specified in the server definition is applied. The new value specified in the client definition is not applied. If this case applies, increase the values of the `adb_sql_exe_hashflt_area_size` operands in both the server definition and client definition.
- If you change the value specified for the `adb_sql_exe_hashflt_area_size` operand, always re-estimate the HADB server's memory requirement. For details about how to re-estimate the memory requirements of the HADB server, see the explanation of the variable *HASHFLT_PROC_SIZE* in (c) Determining the variable RTHD_EXESQLSZ under (2) Determining the real thread private memory requirement (during normal operation) in 6.3.4 Determining the memory requirement during normal operation.

Even if the value of `Hashtbl_filter_disabled_num` that you checked in step 2 is `1` or larger, increasing the value of the `adb_sql_exe_hashflt_area_size` operand in the server definition or client definition might result in faster SQL statement execution.

4. Change the server definition and client definition.

   Increase the values specified for the `adb_sql_exe_hashflt_area_size` operands in the server definition and client definition.

   - When you change the server definition:

     Use the `adbstop` command to terminate the HADB server temporarily. After that, change the server definition, and then use the `adbstart` command to start the HADB server.
   - When you change the client definition:

     Change the client definition by following the procedure explained in *Notes about changing a client definition* in the *HADB Application Development Guide*.

5. Re-execute the SQL statement.

   Re-execute the SQL statement, and check whether the execution time of the SQL statement is shortened. If hash filters are applied, SQL statement execution time might be shortened.

■ Checking whether hash filters are working effectively

Procedure

1. Check whether hash filters are working effectively.

   Check the values of the following two items in the information related to hash table areas that is output as access path statistical information in the SQL trace information:

   - `Hashtbl_sum_filter_check_cnt` (total number of times a hash value was checked by using hash filters)
   - `Hashtbl_sum_filtering_cnt` (total number of times a hash value was excluded by hash filters)

   If the value of `Hashtbl_sum_filtering_cnt` is much smaller than the value of `Hashtbl_sum_filter_check_cnt`, the effectiveness of applying hash filters is diminished. Therefore, SQL statement execution time might be longer as a result of applying hash filters.

2. Re-examine the value specified for the `adb_sql_exe_hashflt_area_size` operand.

If the condition in step 1 is met, consider applying no hash filters for the processing of the SQL statement by specifying `0` for the `adb_sql_exe_hashflt_area_size` operand in the client definition.

When you change the client definition, follow the procedure explained in *Notes about changing a client definition* in the *HADB Application Development Guide*.

Note

- For details about client-definition information, see (3) Client-definition information in 10.11.2 Information that is output as SQL trace information.
- For details about SQL statement statistical information, see (9) SQL statement statistical information in 10.11.2 Information that is output as SQL trace information.
- For details about the information related to hash table areas that is output as access path statistical information, see (e) Information related to hash table areas in (2) Items that are output as access path statistical information under 10.11.3 Examples of output of and output items for access path statistical information.
- For details about the `adb_sql_exe_hashflt_area_size` operand in the server definition, see the description of the adb_sql_exe_hashflt_area_size operand in 7.2.2 Operands related to performance (set format).
- For details about the `adb_sql_exe_hashflt_area_size` operand in the client definition, see *Operands related to performance* in the *HADB Application Development Guide*.

13.9 Tuning to shorten SQL statement execution time by re-examining the table-definition pool size

This section describes how to shorten SQL statement execution time by re-examining the table-definition pool size specified for the following operand:

- adb_sql_tbldef_cache_size operand in the server definition

You might be able to shorten SQL statement execution time by increasing the table-definition pool size. The following shows the procedure for changing the size of the table-definition pool after checking whether the size is appropriate.

Procedure

1. Execute the adbstat command to obtain the statistical information of the HADB server.

   Information about the table-definition pool is output as the statistical information of the HADB server. Check the following items:

   - Tbldef_req_cnt (number of times acquisition of table-definition information was requested)

     The number of times acquisition of table-definition information was requested is output.
   - Tbldef_access_cnt (number of times table-definition information was obtained from the dictionary table)

     Because the relevant table-definition information cannot be obtained from the table-definition pool, the number of times table-definition information was obtained from the dictionary table is output.
   - Tbldef_cache_access_cnt (number of times table-definition information was obtained from the table-definition pool)

     The number of times the relevant table-definition information was obtained from the table-definition pool is output.
   - Tbldef_cache_register_cnt (number of times table-definition information was registered in the table-definition pool)

     The number of times table-definition information was registered in the table-definition pool is output.
   - Tbldef_cache_sweep_cnt (number of times table-definition information was swept out of the table-definition pool)

     The number of times table-definition information was swept out of the table-definition pool is output.
2. Check the hit rate of table-definition information.

   Use the following formula to obtain the hit rate of table-definition information. If the calculated hit rate is 80% or less, increase the value specified for the adb_sql_tbldef_cache_size operand in the server definition.

   ```
   Hit rate of table-definition information (%) = (A ÷ B) × 100
   ```

   A: Tbldef_cache_access_cnt (number of times table-definition information was obtained from the table-definition pool)

   B: Tbldef_req_cnt (number of times acquisition of table-definition information was requested)
3. Check the value of Tbldef_cache_sweep_cnt (number of times table-definition information was swept out of the table-definition pool).

   If the number of times table-definition information was swept out of the table-definition pool is large, increase the value specified for the adb_sql_tbldef_cache_size operand in the server definition.
4. Terminate the HADB server.

   Terminate the HADB server by executing the adbstop command.
5. Change the server definition.

Increase the value specified for the `adb_sql_tbldef_cache_size` operand in the server definition.

6. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the table-definition pool and the `adb_sql_tbldef_cache_size` operand in the server definition, see the description of the adb_sql_tbldef_cache_size operand in 7.2.2 Operands related to performance (set format).

14 Error Handling

This chapter explains the actions that the HADB administrator needs to take if an error occurs in HADB.

14.1 Error-handling flow

If an error occurs in the HADB server, you must identify the cause of the error and eliminate it. The action you take differs depending on the error that has occurred and whether the HADB server terminated abnormally.

First check whether the HADB server terminated abnormally. Execute the following command.

Command to be executed

```
adbls -d srv
```

The HADB server's status is displayed. Check the item `STATUS` in the execution result.

- **If `STATUS` is `ABORT`**

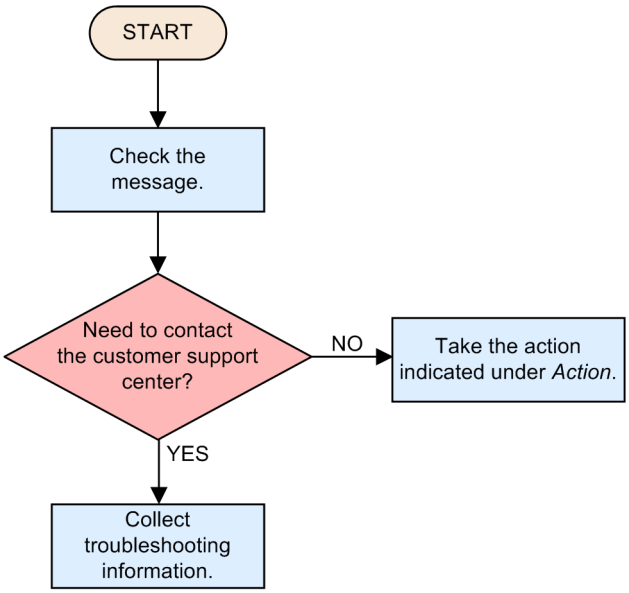
  The HADB server terminated abnormally. See 14.1.1 Steps to take when the HADB server terminated abnormally.
- **If `STATUS` is neither of the above**

  The HADB server did not terminate abnormally. See 14.1.2 Steps to take when the HADB server did not terminate abnormally.

14.1.1 Steps to take when the HADB server terminated abnormally

The following figure shows the steps to take when the HADB server terminated abnormally.

Figure 14-1: Flow of steps to take when the HADB server terminated abnormally



Procedure

1. Check the message

   If the HADB server terminated abnormally, use a text editor to open the server message log file and check the messages that have been output.

   The server message log file is `/spool/adbmessage`*XX*`.log` in the server directory. *XX* in the file name is a sequential number between `01` and `04`. Open the file with the latest modification date and time.

Based on the ID of the last error message that was output, see the manual *HADB Messages*, and check the cause of the error and the recommended action to take under Action.

If the output message text contains `Save troubleshooting information by the "adbinfoget" command` or if the action to take recommends that you contact the customer support center, proceed to step 2.

Otherwise, proceed to step 3.

2. If you need to contact the customer support center

   Also collect troubleshooting information based on the explanation in 14.2 Collecting troubleshooting information (adbinfoget command). Have the troubleshooting information ready for the customer support center when you contact it.

3. If you do not need to contact the customer support center

   Eliminate the cause of the error by taking the action prescribed in the output message.

   After eliminating the cause of the error, restart the HADB server. After the transactions that were being processed when the error occurred are rolled back, the HADB server starts.

14.1.2 Steps to take when the HADB server did not terminate abnormally

This subsection describes the steps to take when the HADB server did not terminate abnormally.

Procedure

1. Check the message.

   If the HADB server did not terminate abnormally, use a text editor to open the server message log file and check any error message that has been output.

   The server message log file is `/spool/adbmessage`*XX*`.log` in the server directory. *XX* in the file name is a sequential number between `01` and `04`. Open the file with the latest modification date and time.

   Based on the ID of the last error message that was output, see the manual *HADB Messages*, and check the cause of the error and the recommended action to take under Action.

2. Take the action described under Action.

   Eliminate the cause of the error by following the steps described under Action. After eliminating the cause of the error, restart the HADB server. After the transactions that were being processed when the error occurred are rolled back, the HADB server starts.

> **Important**
>
> Note that if an error occurs (for example, when an application program or command does not terminate or when the HADB server does not respond), a message might not be output. In such a case, collect troubleshooting information as explained in 14.2 Collecting troubleshooting information (adbinfoget command), and then contact the customer support center.

14.2 Collecting troubleshooting information (adbinfoget command)

To collect the troubleshooting information that will be sent to the customer support center, execute the `adbinfoget` command.

The `adbinfoget` command enables you to collect the following types of troubleshooting-information files.

Troubleshooting-information files that can be collected

- **Troubleshooting-information file (detailed version)**
  This file stores all of the information necessary for troubleshooting.
  Its file name is `adbinfo`*YYYYMMDDhhmmss*`_detail.tar.gz`.
- **Troubleshooting-information file (light version)**
  This file stores information obtained by deleting core files, shared memory, and dump files from the troubleshooting-information file (detailed version).
  Its file name is `adbinfo`*YYYYMMDDhhmmss*`_light.tar.gz`.
- **Troubleshooting-information file (mail version)**
  This file stores only the minimum amount of information necessary for troubleshooting, and is obtained by deleting several more files from the troubleshooting-information file (light version).
  Its file name is `adbinfo`*YYYYMMDDhhmmss*`_mail.tar.gz`.
- **Troubleshooting-information file (root version)**
  This file stores information that can be acquired by a user who has the `root` privilege.
  Its file name is `adbinfo`*YYYYMMDDhhmmss*`_root.tar.gz`.

Note

YYYYMMDDhhmmss in the file name indicates the date and time when the `adbinfoget` command was executed.

The procedure for executing the `adbinfoget` command is described below. For details about the `adbinfoget` command options, see *adbinfoget (Collect Troubleshooting Information)* in the manual *HADB Command Reference*.

Procedure

1. Output troubleshooting information.
   Note that if an error that prevents a message from being output occurs (for example, when an application program or command does not terminate or when the HADB server does not respond), no troubleshooting information is output. Therefore, first, execute the `adbinfoget` command to output troubleshooting information. Execute the following command.
   Note that if a message has been output (when the HADB server terminated abnormally), you do not need to output troubleshooting information. Proceed to step 2.
   Command to be executed

   ```
   adbinfoget -g
   ```

2. Determine the size of the troubleshooting-information files to be collected.
   Execute the `adbinfoget` command to determine the size of the troubleshooting-information files that are to be collected. Execute the command shown below. Because the `-r` option is specified, enter the superuser's password, if necessary.
   Command to be executed

   ```
   adbinfoget -m -r
   ```

Compare the command execution result with the free space on the disk, and determine if there is sufficient free space to collect all the troubleshooting-information files. If the detailed version troubleshooting-information files would be too large, collect the other troubleshooting-information files.

3. Create a directory for storing the troubleshooting-information files.

   Create a directory in which the troubleshooting-information files you plan to collect will be stored.

4. Collect the troubleshooting-information files.

   Execute the `adbinfoget` command to collect the troubleshooting-information files. Depending on the amount of available disk space, execute one of the commands shown below. Because the `-r` option is specified, enter the superuser's password, if necessary.

   ■ For collecting all the troubleshooting-information files

   ```
   adbinfoget -o troubleshooting-information-output-destination-directory -r
   ```

   ■ For collecting all the troubleshooting-information files except the detailed version files

   ```
   adbinfoget -l -o troubleshooting-information-output-destination-directory -r
   ```

   Note

   For the `-o` option, specify the directory that was created in step 3.

5. Send the collected troubleshooting-information files.

   Send the collected troubleshooting-information files (detailed version and root version) to the customer support center.

   If the detailed version troubleshooting-information files are too large to collect due to insufficient disk space or the collected detailed version troubleshooting-information files are too large to send, send only the light and root version troubleshooting-information files to the customer support center.

   If the light version troubleshooting-information files are too large to send, send the mail and root version troubleshooting-information files to the customer support center.

Important

- If the collected troubleshooting-information file is no longer needed, consider deleting it in order to make sure that sufficient free space remains on the disk.
- After you have executed the `adbinfoget` command, consider executing the `adbinfosweep` command based on the explanation in 14.3 Deleting troubleshooting information (adbinfosweep command).

14.3 Deleting troubleshooting information (adbinfosweep command)

By executing the `adbinfosweep` command while the HADB server is stopped, you can delete troubleshooting information that was output by the HADB server.

The troubleshooting information that was output by the HADB server is not automatically deleted in the following cases:

- The troubleshooting information was output when a server process was forcibly terminated.
- The troubleshooting information was output when an internal conflict was detected in a server process.

If troubleshooting information continues to accumulate, the disk might run out of space. Therefore, execute the `adbinfosweep` command regularly to make sure that sufficient free space remains on the disk.

For details about the `adbinfosweep` command, see *adbinfosweep (delete troubleshooting information)* in the manual *HADB Command Reference*.

Execute the `adbinfosweep` command at the following times.

When to execute the `adbinfosweep` command

If the HADB server terminated abnormally, first collect troubleshooting information by executing the `adbinfoget` command, and then execute the `adbinfosweep` command.

By executing the `adbinfosweep` command, you can delete the following types of troubleshooting information.

Types of troubleshooting information that can be deleted

- **Error information (core files)**
  - `$DBDIR/SPOOL/core.`*server-process-process-ID*
  - *path-defined-by-adb_core_path-operand-in-the-server-definition*`/core.`*server-process-process-ID*
- **Server definition storage file**
  - `$ADBDIR/spool/.defrslt`
- **HADB dump file**
  - `$ADBDIR/spool/adbdump`*YYYYMMDDhhmmss*`.`*server-process-process-ID*
    YYYYMMDDhhmmss indicates the time when the file was generated.
  - `$ADBDIR/spool/adbdumperror`*YYYYMMDDhhmmssSSSSSS_TTTTTTTTTTTTTTTTTT*`.`*server-process-process-ID*
    YYYYMMDDhhmmssSSSSSS indicates the time when the file was generated.
    TTTTTTTTTTTTTTTTTT indicates the thread ID of the real thread in which the HADB server's internal conflict error was detected.

> **Important**
>
> Executing the `adbinfosweep` command might not delete troubleshooting information that was output by OS users other than the HADB administrator. If such unnecessary troubleshooting information is not deleted, a superuser can delete it by executing a command such as the operating system's `rm` command.

15 Troubleshooting

This chapter describes the steps to take if problems occur in the HADB server.

15.1 Application-related problems

This section explains how to handle application-related problems.

15.1.1 Steps to take when an application does not terminate or terminates abnormally

If an application does not terminate, or terminates abnormally, check the application's processing status by following the procedure described in 10.8 Checking the transaction processing status.

If needed, execute the `adbcancel` command and terminate the transactions that the application is executing.

15.1.2 Steps to take when an application cannot be executed (when the KFAA40005-E message is output)

If, when you attempt to execute an application, the `KFAA40005-E` message is output even though no other application or command is concurrently being executed, there might be a transaction that has not terminated and to which a locked resource remains allocated.

Execute the `adbls -d cnct` command to check the transaction processing status. Check the output item `STATUS` to determine whether there is any application or command for which `STARTED` is displayed.

If there is any application or command for which `STARTED` is displayed, either wait for the applicable transaction to terminate or use the `adbcancel` command to terminate the applicable transaction.

15.1.3 Steps to take when connection from the application program to the HADB server takes time

If the application program is operating in a Linux environment, queries to the DNS server about the host name and other items might take time. In such cases, it might also take time for the application program to connect to the HADB server.

Make sure that the settings in the `/etc/nsswitch.conf` file are specified as follows:

```
hosts: files myhostname dns
```

15.2 Command-related problems

This section explains how to handle command-related problems.

15.2.1 Steps to take when a command results in a time-out

If executing a command results in a time-out, it might be because the load on the OS was too high, which prevented the OS from being able to finish executing the command within the command's response wait time. If this is the case, re-execute the command.

If re-executing the command still results in a time-out, stop all unnecessary processes to reduce the load on the OS. Then, re-execute the command again.

15.2.2 Steps to take when a command cannot be executed (when the KFAA40005-E message is output)

If, when you attempt to execute a command, the `KFAA40005-E` message is output even though no other application or command is concurrently being executed, there might be a transaction that has not terminated and to which a locked resource remains allocated.

Execute the `adbls -d cnct` command to check the transaction processing status. Check the output item `STATUS` to determine whether there is any application or command for which `STARTED` is displayed.

If there is any application or command for which `STARTED` is displayed, either wait for the applicable transaction to terminate or use the `adbcancel` command to terminate the applicable transaction.

15.2.3 Steps to take when the adbimport or adbidxrebuild command cannot be re-executed

If an interrupted `adbimport` or `adbidxrebuild` command cannot be re-executed, the command might no longer be able to access one of the files listed below. If this is the case, the indicated message is output to the message log file.

- Command status file
  The `KFAA50244-E` message is output.
- Temporary work file
  The `KFAA50247-E` message is output.

If a message other than `KFAA50244-E` or `KFAA50247-E` is output, you can determine the cause of the error based on that message.

The corrective action depends on the message that is output. The following subsections explain the corrective action to take in response to each message.

(1) When the KFAA50244-E message is output (the command status file cannot be accessed)

Execute the `adbidxrebuild` command with the `--force` option specified.

If the `adbimport` command to which background import was being applied (with the `-b` option specified) has been interrupted and then the `adbidxrebuild` command with the `--force` option specified is executed, storage information for the table data that was being processed by the `adbimport` command is deleted. If this happens, the interrupted `adbimport` command returns to unexecuted status. In such a case, re-execute the interrupted `adbimport` command with the `-b` option specified after you have executed the `adbidxrebuild` command.

For details about the corrective action, see the following topics:

- *If an error occurs during re-execution of the adbimport command* under *Handling abnormal termination of the adbimport command* in *adbimport (Import Data)* in the manual *HADB Command Reference*
- *If an error occurs during re-execution of the adbidxrebuild command* under *Handling abnormal termination of the adbidxrebuild command* in *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*

(2) When the KFAA50247-E message is output (temporary work files cannot be accessed)

Execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

If the `adbimport` command to which background import was being applied (with the `-b` option specified) has been interrupted, you can complete the interrupted `adbimport` command's execution by executing the `adbidxrebuild` command with the `--create-temp-file` option specified.

For details about the corrective action, see the following topics:

- *If an error occurs during re-execution of the adbimport command* under *Handling abnormal termination of the adbimport command* in *adbimport (Import Data)* in the manual *HADB Command Reference*
- *If an error occurs during re-execution of the adbidxrebuild command* under *Handling abnormal termination of the adbidxrebuild command* in *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*

(3) When a message other than KFAA50244-E or KFAA50247-E is output

Re-execute the interrupted command according to the corrective action shown in the output message.

15.2.4 Steps to take when the adbunarchivechunk command cannot be re-executed

If an interrupted `adbunarchivechunk` command cannot be re-executed, the command might no longer be able to access one of the following files.

- Command status file
- Temporary work file

You can determine which file caused the error based on the output message.

(1) Steps to take if the command status file cannot be accessed

Re-execute the `adbunarchivechunk` command with the `--force` option specified.

(2) Steps to take if the temporary work file cannot be accessed

Take the corrective action for the output message. Then, re-execute the `adbunarchivechunk` command with the `--force` option specified.

15.2.5 Steps to take in the event of a shortage of disk space for storing temporary work files during command execution

Temporary work files are created in the directory specified in the `-w` option while the following commands are executing:

- `adbimport` command
- `adbidxrebuild` command
- `adbmergechunk` command
- `adbunarchivechunk` command
- `adbreorgsystemdata` command

If any of the following messages is output during execution of one of these commands, a shortage might have occurred in the disk space for storing the temporary work files:

- `KFAA30959-E`
- `KFAA40204-E`
- `KFAA40205-E`
- `KFAA40207-E`
- `KFAA40208-E`
- `KFAA40214-E`
- `KFAA41205-E`
- `KFAA41206-I`

You can determine whether a shortage has occurred in the disk space for storing temporary work files from the information provided in the message. For details, see (1) How to determine whether a shortage has occurred in the disk space for storing temporary work files.

The following are possible causes of a shortage of disk space for storing temporary work files. Take the corrective action appropriate to the cause.

- There are unneeded temporary work files on the disk.
  See (2) When there are unneeded temporary work files on the disk.
- The disk capacity for storing temporary work files is too small to store temporary work files that will be created.
  - When the `adbimport` or `adbidxrebuild` command was interrupted
    See (3) When the disk capacity for storing temporary work files is too small (when the adbimport or adbidxrebuild command was interrupted).

- When the `adbmergechunk`, `adbunarchivechunk`, or `adbreorgsystemdata` command was interrupted

  See (4) When the disk capacity for storing temporary work files is too small (when the adbmergechunk, adbunarchivechunk, or adbreorgsystemdata command was interrupted).

(1) How to determine whether a shortage has occurred in the disk space for storing temporary work files

To determine whether a shortage has occurred in the disk space for storing temporary work files, check the output message. If the output message is any of those listed in the following table, a shortage has occurred in the disk space for storing temporary work files.

Table 15-1: Messages that are output when a shortage has occurred in the disk space for storing temporary work files and the information provided in the messages

No.	Message ID	Information provided in the message		
		File name	Error code	Error number
1	KFAA30959-E	File name#1	--	28 (ENOSPC)
2	KFAA40204-E	File path name#2	--	--
3	KFAA40205-E	--	--	28 (ENOSPC)
4	KFAA40207-E	--	-210 or -230	--
5	KFAA40208-E	--	-210 or -230	--
6	KFAA40214-E	--	--	28 (ENOSPC)
7	KFAA41205-E	File path name#2	--	28 (ENOSPC)
8	KFAA41206-I	File path name#2	--	28 (ENOSPC)

Legend:

--: This information is not output in the corresponding message.

#1

Check the directory that contains the file that is indicated in the message. The following steps explain how to check the directory that contains a specific file:

1. Use the OS's `find` command to locate the file whose file name is displayed in the message.

   The following shows an example of the OS's `find` command.

   ■ Example execution of the OS's `find` command

   ```
   find ~/ -name "xxxxx"
   ```

   Explanation

   Execute the `find` command in which is specified the file name (*xxxxx*) displayed in the message.

2. Check the command's execution results to determine whether the directory storing the file matches the directory specified in the `-w` option of each command.

   If the directories match, a shortage has occurred in the disk space for storing temporary work files.

#2

Check the directory that contains the file based on the path name displayed in the message.

Check if the directory contained in the path name matches the directory specified in the `-w` option of each command.

If the directories match, a shortage has occurred in the disk space for storing temporary work files.

Note

The directory specified in the command's `-w` option means one of the following:

- Directory specified in a command's `-w` option
- Directory specified in the directory path file that was specified in the `-w` option in the command

(2) When there are unneeded temporary work files on the disk

The following procedure explains how to delete the unneeded temporary work files.

Procedure:

1. Check if there are any commands that need to be re-executed.
   Execute the `adbdbstatus` command with `table` specified in the `-c` option. This command's execution results show in the `Rerun_command` column in the table summary information the names of the commands that need to be re-executed. If no commands need to be re-executed, nothing is displayed in the `Rerun_command` column.
   If there is any command that needs to be re-executed, go to step 2. If no commands need to be re-executed, go to step 3.
   For details about the `adbdbstatus` command, see *adbdbstatus (Analyze the Database Status)* in the manual *HADB Command Reference*.
2. Re-execute the commands.
   Re-execute the commands that were found to be needing re-execution in step 1. If the re-executed commands terminate normally, the temporary work files have been deleted, and there is no need to perform step 3.
3. Delete the remaining temporary work files.
   Because any remaining temporary work files are all unneeded, delete them (with the OS's `rm` command, for example).

(3) When the disk capacity for storing temporary work files is too small (when the adbimport or adbidxrebuild command was interrupted)

If the disk capacity for storing temporary work files is too small, you need to change the storage location for temporary work files to another disk with greater capacity. This subsection explains how to change the storage location of temporary work files when the `adbimport` or `adbidxrebuild` command has been interrupted.

For details about how to change the storage location of temporary work files when the `adbmergechunk`, `adbunarchivechunk`, or `adbreorgsystemdata` command has been interrupted, see (4) When the disk capacity for storing temporary work files is too small (when the adbmergechunk, adbunarchivechunk, or adbreorgsystemdata command was interrupted).

The procedure for changing the storage location for temporary work files depends on whether the directory specified in the `-w` option in the command is changed. The directory specified in the command's `-w` option means one of the following:

- Directory specified in the command's `-w` option
- Directory specified in the directory path file that was specified in the `-w` option in the command

The following subsections explain the procedure when the directory specified in the -w option in the command is and is not changed.

(a) When the directory specified in the -w option in the command is changed

Procedure:

1. Change the directory specified in the -w option in the command.

   Change the directory specified in the -w option in the command to a directory on another disk with greater capacity.

   The following shows example directory path file specifications before and after the directories specified in the directory path files are changed.

   ■ Example of directory path file specifications before change

   ```
   /mnt/disk1/xxxxx
   /mnt/disk2/yyyyy
   /mnt/disk3/zzzzz
   ```

   Explanation

   Directories on disks resulting in a space shortage (/mnt/disk1 to /mnt/disk3) are specified in the directory path files before the change as the storage locations for temporary work files.

   ■ Example of directory path file specifications after change

   ```
   /mnt/largedisk1/xxxxx
   /mnt/largedisk2/yyyyy
   /mnt/largedisk3/zzzzz
   ```

   Explanation

   This example changes the storage locations for temporary work files by specifying different disks with greater capacity (/mnt/largedisk1 to /mnt/largedisk3) in place of /mnt/disk1 to /mnt/disk3.

2. Execute the adbidxrebuild command.

   Execute the adbidxrebuild command with the -w option and the --create-temp-file option both specified. For details about the specification format for the adbidxrebuild command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

(b) When the directory specified in the -w option in the command is not changed

Procedure:

1. Copy the temporary work files to a different directory.

   Use the OS's cp command, for example, to copy all temporary work files stored immediately under the directory specified in the -w option of the command to a directory on another disk with a greater capacity.

   The following shows example directory path file specifications and an example of the OS's cp command for copying the temporary work files stored under the directories specified in the directory path files.

   ■ Example of directory path file specifications

   ```
   /mnt/disk1/xxxxx
   /mnt/disk2/yyyyy
   /mnt/disk3/zzzzz
   ```

   Explanation

   The storage locations for temporary work files are set to directories on disks resulting in a space shortage (/mnt/disk1 to /mnt/disk3).

   ■ Example of the OS's cp command

```
cp -R /mnt/disk1/xxxxx  /mnt/largedisk1/xxxxx
cp -R /mnt/disk2/yyyyy  /mnt/largedisk2/yyyyy
cp -R /mnt/disk3/zzzzz  /mnt/largedisk3/zzzzz
```

Explanation

The temporary work files are copied from the directories on the disks resulting in a space shortage (`/mnt/disk1` to `/mnt/disk3`) to directories on different disks with greater capacity (`/mnt/largedisk1` to `/mnt/largedisk3`).

2. Delete the directory specified in the `-w` option in the command.

Delete the directory specified in the `-w` option in the command by using, for example, the OS's `rm` command. The following shows an example of the OS's `rm` command for deleting directories.

■ Example of the OS's `rm` command

```
rm  -r  /mnt/disk1/xxxxx
rm  -r  /mnt/disk2/yyyyy
rm  -r  /mnt/disk3/zzzzz
```

Explanation

The directories are deleted from the disks resulting in a space shortage (`/mnt/disk1` to `/mnt/disk3`).

3. Create a symbolic link to the directory on the disk with the greater capacity.

Execute the OS's `ln` command to create a symbolic link to the directory on the other disk with greater capacity. In this case, specify the directory that was specified in the `-w` option in the command as the path name of the symbolic link. The following shows an example of the OS's `ln` command for creating a symbolic link.

■ Example of the OS's `ln` command

```
ln  -s /mnt/largedisk1/xxxxx  /mnt/disk1/xxxxx
ln  -s /mnt/largedisk2/yyyyy  /mnt/disk2/yyyyy
ln  -s /mnt/largedisk3/zzzzz  /mnt/disk3/zzzzz
```

Explanation

This example specifies directories (`/mnt/largedisk1/xxxxx` to `/mnt/largedisk3/zzzzz`) on disks with greater capacity as the link target and creates symbolic links (`/mnt/disk1/xxxxx` to `/mnt/disk3/zzzzz`) with the same names as the directories specified in the `-w` option in the command (`/mnt/disk1/xxxxx` to `/mnt/disk3/zzzzz`).

4. Re-execute the interrupted command.

Re-execute the interrupted `adbimport` or `adbidxrebuild` command. The command's processing resumes from the sort processing or the B-tree index and text index creation processing.

If the `KFAA50247-E` message is displayed after the command has re-executed, delete all the temporary work files that were copied in step 1. Finally, execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

(4) When the disk capacity for storing temporary work files is too small (when the adbmergechunk, adbunarchivechunk, or adbreorgsystemdata command was interrupted)

If the disk capacity for storing temporary work files is too small, you need to change the storage location for temporary work files to another disk with greater capacity. This subsection explains how to change the storage location of temporary work files when the `adbmergechunk`, `adbunarchivechunk`, or `adbreorgsystemdata` command has been interrupted.

For details about how to change the storage location of temporary work files when the `adbimport` or `adbidxrebuild` command has been interrupted, see (3) When the disk capacity for storing temporary work files is too small (when the adbimport or adbidxrebuild command was interrupted).

The procedure for changing the storage location for temporary work files depends on whether the directory specified in the `-w` option in the command is changed. The directory specified in the command's `-w` option means one of the following:

- Directory specified in the command's `-w` option
- Directory specified in the directory path file that was specified in the `-w` option in the command

The following subsections explain the procedure when the directory specified in the `-w` option in the command is and is not changed.

(a) When the directory specified in the -w option in the command is changed

Procedure:

1. Change the directory specified in the `-w` option in the command.

   Change the directory specified in the `-w` option in the command to a directory on another disk with greater capacity.

   For specification examples of directory path files when changing the directories specified in the file, see (3) When the disk capacity for storing temporary work files is too small (when the adbimport or adbidxrebuild command was interrupted).
2. Re-execute the interrupted command.

   Re-execute the interrupted command (`adbmergechunk`, `adbunarchivechunk`, or `adbreorgsystemdata` command).

(b) When the directory specified in the -w option in the command is not changed

Procedure:

1. Create a symbolic link to the directory on a disk with greater capacity.

   Execute the OS's `ln` command to create a symbolic link to a directory on another disk with greater capacity. In this case, specify the directory that was specified in the `-w` option in the command as the path name of the symbolic link.

   For an example of executing the `ln` OS command, see (3) When the disk capacity for storing temporary work files is too small (when the adbimport or adbidxrebuild command was interrupted).
2. Re-execute the interrupted command.

   Re-execute the interrupted command (`adbmergechunk`, `adbunarchivechunk`, or `adbreorgsystemdata` command).

15.3 Problems related to free space on the disk

This section explains the steps to take when increases in the size of the DB area files or files other than DB area files under the DB directory result in insufficient free space on the disk that stores data.

15.3.1 When a free space shortage is caused by an increase in the size of the DB area files

If the size of a DB area file increases and thus reduces the available space in the disk where the DB area file is stored, try one of the following methods to deal with this problem.

- Expand the size of the DB area files.
- Change the storage location for DB area files.
- Reduce the size of the work table DB area files.

For details about the actions that can be taken if the preceding methods are ineffective for securing free space in the DB areas, see 11.10.6 Securing free space in a data DB area.

(1) Expanding the size of the DB area files

The method of expanding the size of the DB area files varies depending on whether regular files or block special files are used. The following describes each method.

- **For regular files**

  Delete unnecessary files to increase free space on the disk.

  When free space on the disk is increased, the HADB server automatically extends DB area files to their maximum value. For details about the maximum value of DB area files, see 2.4.5 DB area automatic extension.

- **For block special files**

  If you are using the OS-supported LVM, expand the logical volume (LV). For details about how to expand the LV, see the documentation for the operating system you are using.

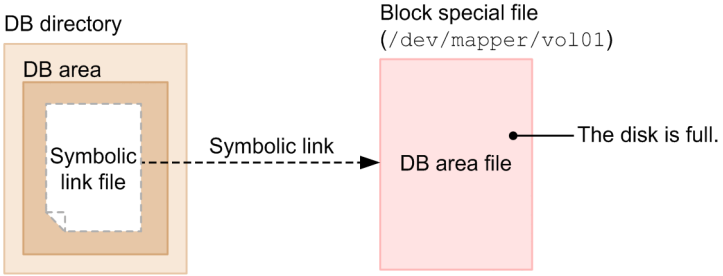
  When the LV is expanded, the HADB server automatically extends DB area files to their maximum value. For details about the maximum value of DB area files, see 2.4.5 DB area automatic extension.

(2) Changing the storage location for DB area files

Change the storage location for DB area files to a disk having a larger capacity. How to change the storage location for DB area file is explained below.

The following figure shows an example of the DB area file configuration before the storage location is changed.

Figure 15-1: Example of DB area file configuration before change



The disk containing the block special file (`/dev/mapper/vol01`) symbolically linked from the DB area under the DB directory is full. Since there is no more free space, the storage locations of these files must be changed.

For details about how to change the storage location of DB area files, see (2) Changing the storage location of a data DB area file (block special file) in 11.10.5 Changing the storage location of data DB area files.

(3) Reducing the size of the work table DB area files

If the disk becomes full due to the increased size of the work table DB area files consisting of regular files, use the following procedure to reduce the size of the work table DB area files.

Procedure:

1. Normally terminate the HADB server by using the `adbstop` command.
2. Delete work table DB area files by using an OS command or similar method.
3. Normally start the HADB server by using the `adbstart` command.

15.3.2 When a free space shortage is caused by failed DB area automatic extension

When automatic extension occurs because the free space in the disk that stores DB area files is insufficient, it's possible for the automatic extension to fail. When this happens, the `KFAA41206-I` message is output. This message displays an error number indicating that the free space in the disk that stores DB area files is insufficient. In this case, consider expanding the storage area for DB area files.

A DB area file that could not be extended automatically is placed temporarily in the full status[#] and is excluded as a candidate for storing table and index data. If all DB area files in the DB area are placed in the full status, the `KFAA30756-E` message, which indicates that the DB area is full, is output. In this case, expand the storage area for DB area files.

For details about how to expand the storage area for DB area files, see 15.3.1 When a free space shortage is caused by an increase in the size of the DB area files.

#

A temporary full status is canceled when:

- The HADB server restarts.
- All DB area files comprising the DB area become full.

> **Note**
>
> A work table DB area file will never become full.

15.3.3 When a free space shortage is caused by an increase in the size of files other than the DB area files

This subsection explains the steps to take when increases in the size of files stored in the DB directory, other than DB area files, result in insufficient free space on the disk being used to store files.

(1) When the space shortage is caused by entity files under the DB directory

Change the storage location of the DB directory to a disk having a larger capacity, as described in the following procedure. The following describes how to change the output destination.

Procedure

1. **Terminate the HADB server.**
   Terminate the HADB server using the `adbstop` command.
2. **Copy the files under the DB directory.**
   Using the operating system's `cp` command, copy all files under the DB directory to a different disk having a larger capacity.
3. Modify the `adb_db_path` operand in the server definition.
   In the `adb_db_path` operand, specify the absolute path name of the DB directory to which you copied the files. The server definition file is the `server.def` file stored under the server directory (`$ADBDIR/conf`). Open and modify it using a text editor.
4. **Start the HADB server.**
   Start the HADB server using the `adbstart` command.

(2) When the space shortage is caused by symbolically linked files

Change the storage location of the symbolically linked files under the DB directory to a disk having a larger capacity. Then, change the link destination of the symbolic link as described in the following procedure.

To change the link target of the symbolic link:

1. Terminate the HADB server.
   Terminate the HADB server using the `adbstop` command.
2. Copy the files at the link destination.
   Using the operating system's cp command, copy all files referenced in the symbolic link file to a different disk having a larger capacity.
3. Change the symbolic link.
   Using the operating system's `ln` command, change the link destination of the symbolic link to point to the disk having the larger capacity.
4. Start the HADB server.
   Start the HADB server using the `adbstart` command.

(3) When a free space shortage is caused by an increase in the size of the statistics log files

Change the output destination for statistics log files to another disk having larger capacity. The following describes how to change the output destination.

Procedure

1. **Terminate the HADB server normally.**
   Execute the `adbstop` command to terminate the HADB server normally.
2. **Change the output destination for statistics log files.**

Specify the output destination directory for statistics log files in the adb_sta_log_path operand. Make sure that statistics log files are output to a disk having a larger capacity.

For details about the adb_sta_log_path operand in the server definition, see the description of the adb_sta_log_path operand in 7.2.7 Operands related to statistical information (set format).

The server definition file is the server.def file stored under the server directory ($ADBDIR/conf). Open and modify it by using a text editor.

3. **Start the HADB server normally.**

   Execute the adbstart command to start the HADB server normally.

If you change the output destination for statistics log files, old statistics log files remain in the output destination that was used before change. If the old statistics log files are not needed, you must delete them manually. For details about how to delete statistics log files, see (4) Deleting statistics log files in 10.10.5 Using the statistics log files.

15.4 Disk-related problems

This section explains the steps to take when a disk-related problem occurs, making it necessary to recover the directories used by HADB.

15.4.1 Recovering the database

If you have to recover a database because a problem such as a disk error occurs, recover it from the backup collected according to 10.3.1 Backup acquisition method.

For details about how to recover a database, see 10.3.2 Recovering the database from the backup.

15.4.2 Recovering the server directory

If a problem occurs on a disk so that it is necessary to recover the server directory, recover the server directory by re-installing the HADB server.

Note that the server directory can be restored to the status it was in before the problem occurred only if the following conditions are satisfied:

- A problem such as a disk error has not occurred in the HADB server's DB directory.
- All files stored under the server directory (in `$ADBDIR/conf` and `$ADBDIR/sample`) have been backed up.
- If users have created files under the server directory (`$ADBDIR`), those files have been backed up.

To recover the server directory:

1. **Re-install the HADB server.**
   Rebuild the server directory by re-installing the HADB server. For details about how to install the HADB server, see 8.2.2 Installation procedure.
2. **Recover the definition files and sample application program from the backup.**
   From the backup, restore all files that were stored in the server directory (in `$ADBDIR/conf` and `$ADBDIR/sample`).
3. **Recover the files that were created by the user from the backup.**
   If users have created files in the server directory (`$ADBDIR`), restore those files from the backup.
4. **Restart the HADB server.**
   Execute the `adbstart` command. The HADB server is restarted.

15.4.3 Recovering the client directory

If a problem occurs on a disk that makes it necessary to recover the client directory, reinstall the HADB client by overwriting the existing client, and then recover the client directory.

Note that the client directory can be restored to the status it was in before the problem occurred only if the following conditions are satisfied:

- The definition files stored under the client directory (`%ADBCLTDIR%\conf` in Windows or `$ADBCLTDIR/conf` in Linux) have been backed up.
- If users have created files under the client directory (`%ADBCLTDIR%` in Windows or `$ADBCLTDIR` in Linux), those files have been backed up.

To recover the client directory:

1. Recover the HADB client.

   Reinstall the HADB client by overwriting the existing client, and then recover the client directory. For details, see *Installing and uninstalling an HADB client* in the *HADB Application Development Guide*.

2. Recover the definition files from the backup.

   From the backup, restore the definition files stored in the client directory (`%ADBCLTDIR%\conf` in Windows or `$ADBCLTDIR/conf` in Linux).

3. Recover the files that were created by the user from the backup.

   If users have created files in the client directory (`%ADBCLTDIR%` in Windows or `$ADBCLTDIR` in Linux), restore those files from the backup.

15.5 Problems related to files under the DB directory

This section explains the steps to take when a problem occurs in files stored under the DB directory.

15.5.1 Steps to take when a failure occurs in a DB area file

If a failure occurs in a DB area file, take corrective action for the output message.

If you cannot correct the failure in the DB area file, recover the database from the backup.

15.5.2 Steps to take when a problem occurs in a system log file

When a problem occurs in a system log file (master log file or user log files), one of the following messages is output:

- `KFAA30959-E`
- `KFAA41205-E`

If you cannot resolve the problem that caused the error message to be output after checking the return codes for the system calls that are found in the above messages, take the following actions.

To resolve the problem that caused the error message to be output:

1. **Check whether the HADB server can be restarted.**

   See if the HADB server can be terminated using the `adbstop` command and restarted using the `adbstart` command. When the HADB server is restarted, the master log file area that was being used previously is reinitialized. Therefore, if the problem was caused by a shortage in the master log file capacity, restarting the HADB server can solve the problem.

   If you cannot restart the HADB server, proceed to step 2.

2. **Check the free space on the disk in which the DB directory is stored.**

   System log files are stored in the DB directory (`$DBDIR/ADBSYS/ADBSLG`). Check whether there is sufficient free space on the disk on which the DB directory is stored. If the disk has no free space, allocate free space on the disk by following 15.3.3 When a free space shortage is caused by an increase in the size of files other than the DB area files.

   If the disk has sufficient free space, proceed to step 3.

3. **Check whether a system log file has been corrupted.**

   If you cannot resolve the problem even though the disk has sufficient free space, a system log file might have been corrupted. If an error is returned when you try to access the system log file, the system log file is corrupted. Proceed to step 4.

4. **Recover the system log file.**

   If a system log file is corrupted, you must recover it from the backup. If a database backup collected according to 10.3.1 Backup acquisition method is available, recover the system log file from the backup by following 10.3.2 Recovering the database from the backup.

   If the database has not been backed up, the system log file cannot be recovered, and therefore you need to re-create the database.

15.6 Memory-related problems

This subsection explains the steps to take when the KFAA30930-E or KFAA40007-E message is output. Using the method described below, make sure plenty of memory is available to the HADB server.

15.6.1 Steps to take when a memory shortage occurs during HADB server startup

This subsection explains the steps to take when the KFAA40007-E message is output during HADB server startup. The steps depend on whether the adb_sys_memory_limit operand is specified in the server definition.

(1) When the adb_sys_memory_limit operand is specified in the server definition

The memory used by the HADB server is insufficient. Increase the value specified for the adb_sys_memory_limit operand in the server definition. Perform the following procedure.

Procedure:

1. Change the server definition.
   In the server definition, set the adb_sys_memory_limit operand to a value that is equal to or larger than the value determined from the following formula.
   Formula (megabytes)

   Maximum size of memory to be used by the HADB server =
   value specified for the adb_sys_memory_limit operand in the server definition (megabytes)
   + ↑*get_size* ÷ (100 × 1,024 × 1,024)↑ × 100

   Explanation of variables

   get_size: Value of variable *bb....bb* in the KFAA40007-E message (bytes)
2. Start the HADB server.
   Execute the adbstart command to start the HADB server.

> **Note**
>
> For details about the adb_sys_memory_limit operand in the server definition, see the adb_sys_memory_limit operand in 7.2.2 Operands related to performance (set format).

(2) When the adb_sys_memory_limit operand is not specified in the server definition

Check the variable *aa....aa* in the KFAA40007-E message indicating the type of insufficient memory. The corrective action depends on the value of *aa....aa*.

(a) If the variable aa....aa is PROCESS

Process common memory is insufficient. Increase the value specified for the adb_sys_proc_area_max operand in the server definition. Perform the following procedure.

Procedure:

1. Change the server definition.
   In the server definition, set the `adb_sys_proc_area_max` operand to a value that is equal to or larger than the value determined from the following formula.
   Formula (megabytes)

   Maximum size of process common memory that can be allocated =
   (↑*max_size* ÷ 100↑ × 100) + (↑*get_size* ÷ (100 × 1,024 × 1,024)↑ × 100) + 1

   Explanation of variables
   max_size: Value specified in the `adb_sys_proc_area_max` operand for the maximum size of process common memory that can be allocated (megabytes)
   get_size: Value of variable *bb....bb* in message `KFAA40007-E` (bytes)
2. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the `adb_sys_proc_area_max` operand in the server definition, see the adb_sys_proc_area_max operand in 7.2.2 Operands related to performance (set format).

(b) If the variable aa....aa is THREAD

Real thread private memory is insufficient. Increase the value specified for the `adb_sys_rthd_area_max` operand in the server definition. Perform the following procedure.

Procedure:

1. Change the server definition.
   In the server definition, set the `adb_sys_rthd_area_max` operand to a value that is equal to or larger than the value determined from the following formula.
   Formula (megabytes)

   Maximum size of real thread private memory that can be allocated =
   (↑*max_size* ÷ 100↑ × 100) + (↑*get_size* ÷ (100 × 1,024 × 1,024)↑ × 100) + 1

   Explanation of variables
   max_size: Value specified in the `adb_sys_rthd_area_max` operand for the maximum size of real thread private memory that can be allocated (megabytes)
   get_size: Value of variable *bb....bb* in message `KFAA40007-E` (bytes)
2. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.

> **Note**
>
> For details about the `adb_sys_rthd_area_max` operand in the server definition, see the adb_sys_rthd_area_max operand in 7.2.2 Operands related to performance (set format).

(c) If the variable aa....aa is HEAP

Heap memory is insufficient. Perform the following procedure.

Procedure:

1. Check whether there are any unneeded processes. If you find any unneeded processes, terminate or delete them. Then, execute the `adbstart` command to start the HADB server.
2. If there is still not enough memory after performing step 1, take the following action.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.
3. If there is still not enough memory after performing step 2, take the following action.
   - Change the kernel parameter settings to increase the maximum size of memory that can be used by the process.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.

(d) If the variable aa....aa is SHARE

Shared memory is insufficient. Perform the following procedure.

Procedure:

1. Check whether there are any unneeded processes or shared memory. If you find any unneeded processes, terminate them. If you find any unneeded shared memory, delete it. Then, execute the `adbstart` command to start the HADB server.
2. If there is still not enough shared memory after performing step 1, take the following action.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.
3. If there is still not enough shared memory after performing step 2, take the following action.
   - Change the kernel parameter settings to increase the maximum size of shared memory.
     If you have applied `HugePages` to the HADB server's shared memory, make sure that the kernel parameters are set to make `HugePages` usable. For details, see 6.2 Estimating the kernel parameters.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.

15.6.2 Steps to take when a memory shortage occurs during execution of an SQL statement or command

This subsection explains the steps to take when either of the following messages is output during execution of an SQL statement or commands:

Messages that are output

- `KFAA30930-E`
- `KFAA40007-E`

The steps depend on whether the adb_sys_memory_limit operand is specified in the server definition.

(1) When the adb_sys_memory_limit operand is specified in the server definition

The memory used by the HADB server is insufficient. Increase the value specified for the adb_sys_memory_limit operand in the server definition. Perform the following procedure.

Procedure:

1. Terminate the HADB server.
   Terminate the HADB server by executing the adbstop command.
2. Change the server definition.
   In the server definition, set the adb_sys_memory_limit operand to a value that is equal to or larger than the value determined from the following formula.
   Formula (megabytes)

   Maximum size of memory to be used by the HADB server =
   value specified for the adb_sys_memory_limit operand in the server definition (megabytes)
   + ↑*get_size* ÷ (100 × 1,024 × 1,024)↑ × 100

   Explanation of variables
   get_size: Value of variable *bb....bb* in the KFAA30930-E or KFAA40007-E message (bytes)
3. Start the HADB server.
   Execute the adbstart command to start the HADB server.
4. Re-execute the SQL statement or command.

> **Note**
>
> For details about the adb_sys_memory_limit operand in the server definition, see the adb_sys_memory_limit operand in 7.2.2 Operands related to performance (set format).

(2) When the adb_sys_memory_limit operand is not specified in the server definition

Check the variable *aa....aa* in the KFAA30930-E or KFAA40007-E messages indicating the type of insufficient memory. The corrective action depends on the value of *aa....aa*.

(a) If the variable aa....aa is PROCESS

Process common memory is insufficient. Increase the value specified for the adb_sys_proc_area_max operand in the server definition. Perform the following procedure.

Procedure:

1. Estimate the maximum size of the memory used by the HADB server.
   Execute the adbls -d mem command. In the output result, look for the value in the column MAX on the line whose THREAD-NO is 0.

For details about the `adbls -d mem` command, see *adbls -d mem (display the memory usage status)* in the manual *HADB Command Reference*.

2. Terminate the HADB server.
   Terminate the HADB server by executing the `adbstop` command.
3. Change the server definition.
   In the server definition, set the `adb_sys_proc_area_max` operand to a value that is equal to or larger than the value determined from the following formula.
   Formula (megabytes)

   Maximum size of process common memory that can be allocated =
   (↑*max_size* ÷ (100 × 1,024 × 1,024)↑ × 100) + (↑*get_size* ÷ (100 × 1,024 × 1,024)↑ × 100) + 1

   Explanation of variables
   max_size: Value of `MAX` identified in step 1 (bytes)
   get_size: Value of the variable *bb....bb* in the output message (bytes)
4. Start the HADB server.
   Execute the `adbstart` command to start the HADB server.
5. Re-execute the SQL statement or command.

> **Note**
>
> For details about the `adb_sys_proc_area_max` operand in the server definition, see the adb_sys_proc_area_max operand in 7.2.2 Operands related to performance (set format).

(b) If the variable aa....aa is THREAD

Real thread private memory is insufficient. Increase the value specified for the `adb_sys_rthd_area_max` operand in the server definition. Perform the following procedure.

Procedure:

1. Estimate the maximum size of the memory used by the HADB server.
   Execute the `adbls -d mem` command. In the output result, look for the maximum value in the column `MAX` on all of the lines whose `THREAD-NO` is not `0`.
   For details about the `adbls -d mem` command, see *adbls -d mem (display the memory usage status)* in the manual *HADB Command Reference*.
2. Terminate the HADB server.
   Terminate the HADB server by executing the `adbstop` command.
3. Change the server definition.
   In the server definition, set the `adb_sys_rthd_area_max` operand to a value that is equal to or larger than the value determined from the following formula.
   Formula (megabytes)

   Maximum size of real thread private memory that can be allocated =
   (↑*max_size* ÷ (100 × 1,024 × 1,024)↑ × 100) + (↑*get_size* ÷ (100 × 1,024 × 1,024)↑ × 100) + 1

Explanation of variables

max_size: Maximum value of `MAX` identified in step 1 (bytes)

get_size: Value of the variable *bb....bb* in the output message (bytes)

4. Start the HADB server.

   Execute the `adbstart` command to start the HADB server.

5. Re-execute the SQL statement or command.

> **Note**
>
> For details about the `adb_sys_rthd_area_max` operand in the server definition, see the adb_sys_rthd_area_max operand in 7.2.2 Operands related to performance (set format).

(c) If the variable aa....aa is HEAP

Heap memory is insufficient. Perform the following procedure.

Procedure:

1. Check whether there are any unneeded processes. If you find any unneeded processes, terminate or delete them.
2. If there is still not enough memory after performing step 1, take the following action.
   - Execute the `adbstop` command to terminate the HADB server.
   - Execute the `adbstart` command to start the HADB server.
3. If there is still not enough memory after performing step 2, take the following action.
   - Execute the `adbstop` command to terminate the HADB server.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.

   If the problem occurred on a client, restart the client's OS.
4. If there is still not enough memory after performing step 3, take the following action.
   - Execute the `adbstop` command to terminate the HADB server.
   - Change the kernel parameter settings to increase the maximum size of memory that can be used by the process.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.

   If the problem occurred on a client, change the kernel parameter settings on the client to increase the maximum size of memory that can be used by the process. Then, restart the client's OS.
5. Re-execute the SQL statement or command.

(d) If the variable aa....aa is SHARE

Shared memory is insufficient. Perform the following procedure.

Procedure:

1. Check whether there are any unneeded processes or shared memory. If you find any unneeded processes, terminate them. If you find any unneeded shared memory, delete it.

2. If there is still not enough shared memory after performing step 1, take the following action.
   - Execute the `adbstop` command to terminate the HADB server.
   - Execute the `adbstart` command to start the HADB server.
3. If there is still not enough shared memory after performing step 2, take the following action.
   - Execute the `adbstop` command to terminate the HADB server.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.

   If the problem occurred on a client, restart the client's OS.
4. If there is still not enough shared memory after performing step 3, take the following action.
   - Execute the `adbstop` command to terminate the HADB server.
   - Change the kernel parameter settings to increase the maximum size of shared memory.
     If you have applied `HugePages` to the HADB server's shared memory, make sure that the kernel parameters are set to make `HugePages` usable. For details, see 6.2 Estimating the kernel parameters.
   - Restart the OS.
   - Execute the `adbstart` command to start the HADB server.

   If the problem occurred on a client, increase the maximum size of shared memory on the client. Then, restart the client's OS.
5. Re-execute the SQL statement or command.

15.7 Problems related to DB areas

This section explains how to handle problems related to data DB areas.

15.7.1 Steps to take when a data DB area becomes full

This subsection explains the steps to take when a data DB area becomes full.

When a data DB area becomes full, the `KFAA30756-E` message is output. A data DB area becomes full in one of the following cases:

- There is no more free space in the disk storing the data DB area files that comprise a data DB area.
- The data DB area files comprising a data DB area cannot be automatically extended because the upper limit for automatic extension has been reached.

When a data DB area becomes full, you need to execute the `adbmodarea` command to extend the applicable data DB area (by adding a data DB area file). Expand the applicable data DB area based on the explanation in 11.10.3 Expanding a data DB area (adding a data DB area file).

The upper limit for the number of data DB area files that comprise a data DB area is 1,024. You cannot add data DB area files beyond this upper limit. Therefore, if you want to expand a data DB area that consists of 1,024 data DB area files, see 15.7.3 Steps to take when a data DB area can no longer be expanded.

If the disk that stores data DB area files runs out of free space, you can take one of the following actions:

- Expand the size of the DB area files.
- Change the storage location for DB area files.

For details, see 15.3.1 When a free space shortage is caused by an increase in the size of the DB area files.

15.7.2 Steps to take when a data DB area can no longer be added

This subsection explains the steps to take when you can no longer add a new data DB area.

The maximum number of data DB areas that the HADB server can manage is 1,014. You cannot add data DB areas beyond this upper limit.

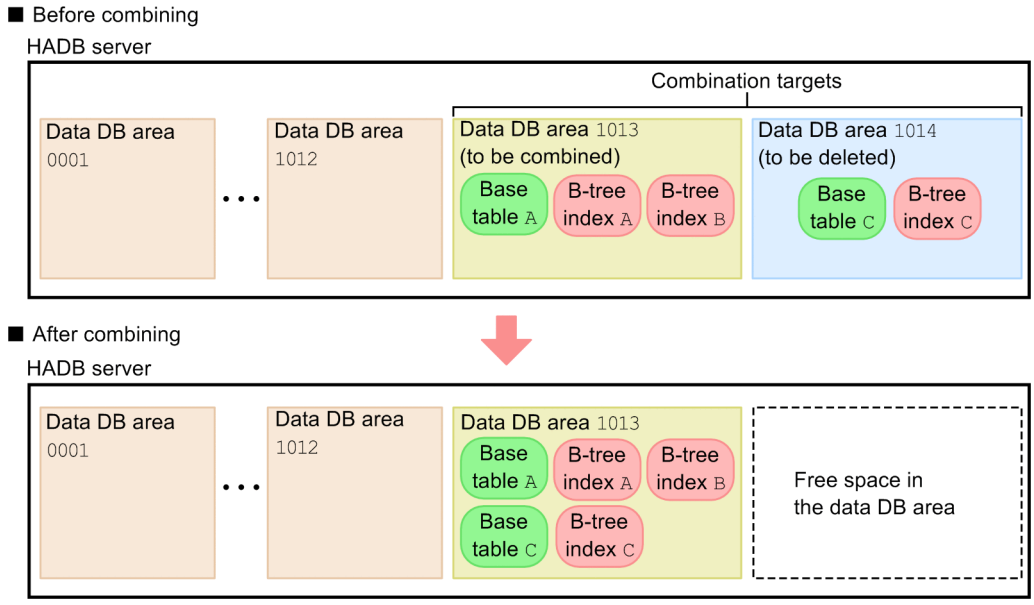
Therefore, if you want to add a new data DB area when there are already 1,014 data DB areas, you need to combine the existing data DB areas so that the resulting number of data DB areas does not exceed the upper limit.

> **Note**
>
> If you will be combining data DB areas, we recommend that you combine data DB areas that store base tables and indexes whose sizes are small (a few megabytes). For details, see 5.6.1 Points to consider when designing a data DB area.

(1) Deleting and combining data DB areas that store tables and indexes (when execution of the adbidxrebuild command is unnecessary)

The following figure shows how to delete and combine data DB areas that store tables and indexes.

Figure 15-2: Example 1 of combining existing data DB areas



■ Relationship between the base table and B-tree indexes

- B-tree indexes A and B are defined for base table A.
- B-tree index C is defined for base table C.

1. Output the data from the table stored in the data DB area to be deleted.

   Use the adbexport command to output the data of the base table into an output data file. Here, the data to output is the data in base table C, stored in data DB area 1014.

2. Delete the table and indexes stored in the data DB area to be deleted.

   Use the DROP TABLE statement to delete any base table or index that is defined for the data DB area to be deleted. Executing the DROP TABLE statement deletes the indexes defined for the base table at the same time. Here, the items to delete are base table C and B-tree index C, stored in data DB area 1014.

3. Delete the data DB area designated for deletion.

   Use the adbmodarea command to delete the data DB area designated for deletion. Here, that is data DB area 1014.

4. Redefine a table and indexes for the newly-combined data DB area.

   Use the CREATE TABLE or CREATE INDEX statement to redefine, for the newly-combined data DB area, any base table or index deleted in step 2. Here, redefine base table C and B-tree index C for data DB area 1013.

5. Re-store all table data that was output.

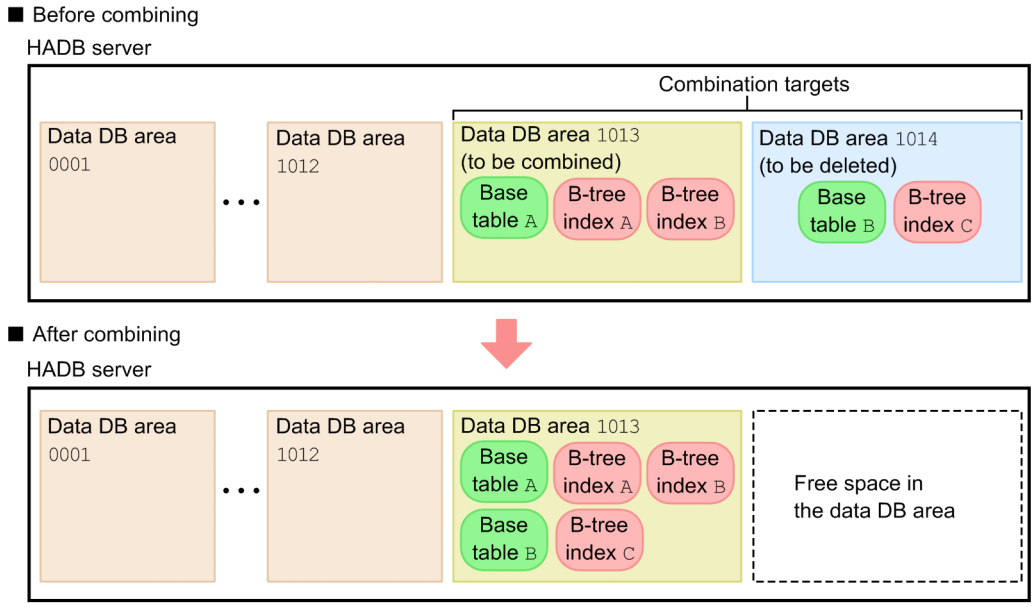
   Use the adbimport command to re-store all base table data output in step 1 in the base table that you redefined in step 4. Here, re-store the data in redefined base table C.

Because combining the existing data DB areas has brought the number of data DB areas below the upper limit, you can now add a new data DB area.

(2) Deleting and combining data DB areas that store tables and indexes (when execution of the adbidxrebuild command is necessary)

The following figure shows how to delete and combine data DB areas that store tables and indexes.

Figure 15-3: Example 2 of combining existing data DB areas



■ Relationship between the base table and B-tree indexes

- B-tree indexes A, B, and C are defined for base table A.
- No index is defined for base table B.

1. Output the data from the table stored in the data DB area to be deleted.

   Use the adbexport command to output the data of the base table into an output data file. Here, the data to output is the data in base table B, stored in data DB area 1014.

2. Delete the tables and indexes stored in the data DB area to be deleted.

   Use the DROP TABLE statement to delete the base table defined for the data DB area to be deleted. Also use the DROP INDEX statement to delete any index. Here, the items to delete are base table B and B-tree index C stored in data DB area 1014.

3. Delete the data DB area designated for deletion.

   Use the adbmodarea command to delete the data DB area designated for deletion. Here, that is data DB area 1014.

4. Redefine a table and indexes for the newly-combined data DB area.

   Use the CREATE TABLE or CREATE INDEX statement to redefine for the newly-combined data DB area any base table or index deleted in step 2. Here, redefine base table B for data DB area 1013. Also redefine B-tree index C for base table A.

5. Re-store all table data that was output.

   Use the adbimport command to re-store all base table data that was output in step 1in the base table that you redefined in step 4. Here, re-store the data in redefined base table B.

6. Rebuild indexes.

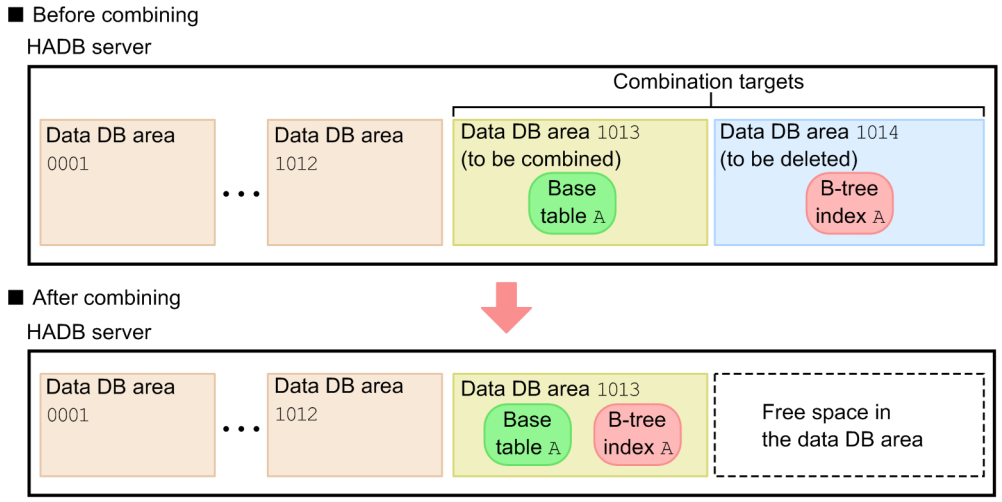
Because the index you redefined in step 4 is in unfinished status, you need to rebuild it. Use the `adbidxrebuild` command to rebuild the index. Here, execute the `adbidxrebuild` command on base table `A` for which B-tree index `C` was redefined.

Because combining the existing data DB areas has brought the number of data DB areas below the upper limit, you can now add a new data DB area.

(3) Deleting and combining data DB areas that store only indexes

The following figure shows how to delete and combine data DB areas that store only indexes.

Figure 15-4: Example 3 of combining existing data DB areas



■ Relationship between the base table and the B-tree index

B-tree index `A` is defined for base table `A`.

1. Delete the index stored in the data DB area to be deleted.

   Use the `DROP INDEX` statement to delete any index stored in the data DB area to be deleted. Here, delete B-tree index `A` stored in data DB area `1014`.

2. Delete the data DB area designated for deletion.

   Use the `adbmodarea` command to delete the data DB area designated for deletion. Here, that is data DB area `1014`.

3. Redefine indexes for the newly-combined data DB area.

   Use the `CREATE INDEX` statement to redefine for the newly-combined data DB area the index deleted in step 1. Here, redefine B-tree index `A` for base table `A` stored in data DB area `1013`.

4. Rebuild indexes.

   Because the index you redefined in step 3 is in unfinished status, you need to rebuild it. Use the `adbidxrebuild` command to rebuild the index. Here, execute the `adbidxrebuild` command on base table `A` for which B-tree index `A` was redefined.

Because combining the existing data DB areas has brought the number of data DB areas below the upper limit, you can now add a new data DB area.

15.7.3 Steps to take when a data DB area can no longer be expanded

This subsection explains the steps to take when an existing data DB area can no longer be expanded using the `adbmodarea` command.

The upper limit for the number of data DB area files that comprise a data DB area is 1,024. You cannot add data DB area files beyond this upper limit.

Therefore, if you want to expand a data DB area that already consists of 1,024 data DB area files, you need to rebuild the data DB area, as described in the following procedure:

1. Output all data from the table.

   Use the `adbexport` command to output all data from the base table to an output data file.

2. Delete the base table and indexes.

   Delete the base table and indexes (B-tree index, text index, and range index) that are defined for the data DB area you are rebuilding.

   Use the `DROP TABLE` statement for deletion if a base table alone is defined, or if both a base table and an index are defined. Executing the `DROP TABLE` statement deletes any indexes defined for the base table at the same time. If only indexes are defined, use the `DROP INDEX` statement for deletion.

3. Delete the data DB area.

   Use the `adbmodarea` command to delete the target data DB area.

4. Add a new data DB area.

   Use the `adbmodarea` command to add a new data DB area.

   When you do so, take into consideration the volume of data that will be added in the future, and specify sufficiently large values for the following: the initial allocation size specified for the `-i` option of the `adbaddarea` operand of the DB area addition/modification option, and the number of data DB area files to be created.

5. Redefine a table and indexes.

   Use the `CREATE TABLE` and `CREATE INDEX` statements to redefine the base table and indexes you deleted in step 2 for the new data DB area you added in step 4.

6. Re-store all table data that was output.

   Use the `adbimport` command to re-store all base table data that was output in step 1 in the base table you redefined in step 5.

15.8 Problems related to base tables

This section explains the steps to take when a problem occurs in a base table.

15.8.1 Steps to take when a base table becomes non-updatable

If one of the following commands is interrupted, the base table on which processing is being performed might become non-updatable due to inconsistencies between table information and index information. This is called a *non-updatable base table*.

■ Target commands

- `adbimport` command
- `adbidxrebuild` command
- `adbunarchivechunk` command

An error occurs if you perform the following operations on a non-updatable base table.

■ Operations that will cause an error

- Executing the `ALTER TABLE` statement
- Executing the `CREATE INDEX` statement
- Executing the `INSERT` statement
- Executing the `UPDATE` statement
- Executing the `DELETE` statement
- Executing the `PURGE CHUNK` statement
- Executing the `adbmergechunk` command
- Executing the `adbchgchunkstatus` command
- Executing the `adbchgchunkcomment` command
- Executing the `adbimport` command[#1]
- Executing the `adbidxrebuild` command[#1]
- Executing the `adbarchivechunk` command
- Executing the `adbunarchivechunk` command[#2]

#1

If the `adbunarchivechunk` command is interrupted and the base table becomes non-updatable, an error occurs. If the `adbimport` or `adbidxrebuild` command is interrupted and the base table becomes non-updatable, an error does not occur.

#2

If the `adbimport` or `adbidxrebuild` command is interrupted and the base table becomes non-updatable, an error occurs. If the `adbunarchivechunk` command is interrupted and the base table becomes non-updatable, an error does not occur.

> **Note**
>
> The `TRUNCATE TABLE` statement can be executed. Because this statement deletes all row data from a base table, the base table is released from non-updatable status.

(1) Releasing a base table from non-updatable status

The following describes how to release a base table from non-updatable status.

Procedure:

1. Check whether the base table in question is non-updatable.
   You can use the `adbdbstatus` command to check whether the base table is non-updatable. See 10.9.2 Checking the status and usage of a base table.
   If the base table is non-updatable, proceed to step 2.
2. Release the base table from non-updatable status.
   Execute the interrupted command.
   - **If the `adbimport` command was interrupted**
     Re-execute the `adbimport` command. For details, see *Handling abnormal termination of the adbimport command* in *adbimport (Import Data)* in the manual *HADB Command Reference*.
   - **If the `adbidxrebuild` command was interrupted**
     Re-execute the `adbidxrebuild` command. For details, see *Handling abnormal termination of the adbidxrebuild command* in *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.
   - **If the `adbunarchivechunk` command was interrupted**
     Re-execute the `adbunarchivechunk` command. For details, see *Handling abnormal termination of the adbunarchivechunk command* in *adbunarchivechunk (Unarchive Chunk)* in the manual *HADB Command Reference*.

   When the re-executed command terminates normally, the inconsistencies between the table information and the index information will be resolved and the base table will be released from non-updatable status.
   If an error occurs during re-execution of a command, see (2) Steps to take if an error occurs during re-execution of a command.

(2) Steps to take if an error occurs during re-execution of a command

During re-execution of a command, an error might occur due to any of the following causes. Take the corrective action appropriate to the cause.

■ Causes of errors that can occur during command re-execution

- A temporary work file or command status file that was created by the interrupted command cannot be accessed.
  See Action 1.
- An upgrade was performed without releasing a base table from the non-updatable status.
  See Action 1.
- The free space in the data DB area that stores an index is insufficient.
  See Action 2.

■ Action 1

Check whether either of the following messages has been output to the message log file:

- KFAA50244-E message
- KFAA50247-E message

The corrective action you must take varies depending on whether the preceding messages are output. If a message other than the preceding one is output, take corrective action according to the output message. When the corrective action is completed, the base table will be released from non-updatable status.

- **If the KFAA50244-E message is output**

  The corrective action you must take varies depending on the command that could not be re-executed.

  - **If an error occurs when the adbimport or adbidxrebuild command is re-executed**

    Execute the adbidxrebuild command with the --force option specified.

    If the adbimport command with background import applied (the -b option specified) has been interrupted, executing the adbidxrebuild command with the --force option deletes storage information for the table data that was being processed by the adbimport command. In this case, after executing the adbidxrebuild command, re-execute the interrupted adbimport command with the -b option specified.

    For details about the --force option of the adbidxrebuild command, see *Specification format for the adbidxrebuild command* in *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

  - **If an error occurs when the adbunarchivechunk command is re-executed**

    Execute the adbunarchivechunk command with the --force option specified.

    For details about the --force option of the adbunarchivechunk command, see *Specification format for the adbunarchivechunk command* in *adbunarchivechunk (Unarchive Chunk)* in the manual *HADB Command Reference*.

- **If the KFAA50247-E message is output**

  Execute the adbidxrebuild command with the --create-temp-file option specified.

  For details about the --create-temp-file option of the adbidxrebuild command, see *Specification format for the adbidxrebuild command* in *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

■ Action 2

The following messages have been output to the message log file:

- KFAA30756-E
- KFAA61210-E

Expand the data DB area, and then re-execute the command. For details about how to expand a data DB area, see 11.10.3 Expanding a data DB area (adding a data DB area file).

If you cannot expand the data DB area, secure free space in the data DB area, and then re-execute the command. For details about how to secure free space in a data DB area, see 11.10.6 Securing free space in a data DB area.

If you cannot also secure free space in the data DB area, use the following procedure to release the base table from the non-updatable status. Note that in the following procedure, before releasing the base table from the non-updatable status, you temporarily delete indexes to secure the amount of free space in the data DB area. Then, after releasing the base table from the non-updatable status, you redefine and rebuild the indexes. Therefore, free space for re-creating the indexes is required in the data DB area. Because free space is required to rebuild the indexes, check whether you can secure sufficient free space for rebuilding the indexes beforehand. After that, release the base table from the non-

updatable status. For details about how to secure free space in a data DB area, see 11.10.6 Securing free space in a data DB area. If you cannot secure the free space necessary to rebuild the indexes, try expanding the data DB area again.

Procedure

1. Determine the indexes to be deleted.

   Of the indexes stored in the data DB area that has insufficient free space (and which are defined for the base table you want to release from non-updatable status), determine the ones to delete temporarily.

2. Obtain the definition information of the indexes.

   Because you must be able to rebuild indexes with the same definition information as the indexes you will temporarily delete (indexes determined in step 1), obtain the necessary definition information beforehand.

   For details about how to obtain index definition information, see (29) Finding out index definition information in B.22 Searching a dictionary table.

3. Delete the indexes.

   Temporarily delete the indexes determined in step 1.

   For details about how to delete indexes, see 11.3.9 Deleting an index.

4. Re-execute the command that was interrupted previously.

   Re-execute the previously interrupted command for the base table to be released from the non-updatable status. If the command terminates normally, the base table is released from the non-updatable status.

   For details about how to check the interrupted commands, see (1) Checking whether a base table is non-updatable in 10.9.2 Checking the status and usage of a base table.

   If an error occurs during command execution, proceed to step 5. If the command terminates normally, proceed to step 6.

5. Take measures for the command that failed in step 4.

   - **If the cause of the error is insufficient free space in the data DB area**

     You need also to delete other indexes to secure free space in the data DB area. Repeat the procedure from step 1 for other indexes.

   - **If the cause of the error is not insufficient free space in the data DB area**

     Check the error message that has been output, and eliminate the cause of the error. Then, re-execute the command.

     If the command terminates normally, the base table is released from the non-updatable status. If the command terminates normally, proceed to step 6.

6. Secure free space in the data DB area so that indexes can be rebuilt.

   Before you can rebuild indexes, secure free space in the data DB area that stored the indexes that were deleted in step 3.

   For details about how to secure free space, see 11.10.6 Securing free space in a data DB area.

7. Redefine indexes.

   Redefine the deleted indexes based on the definition information that was obtained in step 2.

8. Rebuild indexes.

   Rebuild the indexes that were redefined in step 7. For details about how to rebuild indexes, see one of the following subsections:

   - 11.3.2 Rebuilding B-tree indexes
   - 11.3.3 Rebuilding text indexes
   - 11.3.5 Rebuilding range indexes

(3) When a non-updatable base table is searched

You can use the SELECT statement to search a non-updatable base table. Note, however, that the search result differs depending on the interrupted command.

- **In the case of the adbimport command**

  The search result depends on whether the -d option was specified during execution of the adbimport command.

  If the -d option is omitted

  The same results are returned as when the search is performed on the base table prior to execution of the adbimport command.

  If the -d option is specified

  The retrieval results are 0 for both a SELECT statement that does not use indexes and a SELECT statement that uses range indexes.

  A SELECT statement that uses B-tree indexes or text indexes results in an error.

- **In the case of the adbidxrebuild command**

  If you execute a SELECT statement that uses B-tree indexes, text indexes, or range indexes whose rebuild processing has not been completed, an error occurs because the indexes are in unfinished status.

- **In the case of the adbunarchivechunk command**

  The same results are returned as when the search is performed on the base table prior to execution of the adbunarchivechunk command.

15.8.2 When a column cannot be added to a base table

This subsection explains the steps to take when a column cannot be added to a base table by using the ALTER TABLE definition SQL statement.

In the following situations, you cannot add a column to a base table:

- The target base table is non-updatable.
- The target base table is a FIX table, and the segments for storing rows have been assigned to the table.
- Segments for storing rows are assigned to the target base table for which the CREATE TABLE statement was executed with BRANCH ALL specified.
- For a base table to which segments for storing rows are assigned, NOT NULL is specified in the column definition of the ALTER TABLE statement.

If a base table is non-updatable, release it from non-updatable status based on the explanation in 15.8.1 Steps to take when a base table becomes non-updatable. Then, re-execute the ALTER TABLE statement.

In other cases, add a column by performing the steps described in the following procedure.

■ Steps to take when a column cannot be added to a base table

1. Output data in the base table to a file.

   Use the adbexport command to output all data from the base table to a file. An output data file is created. The output data file created here becomes the input data file used in steps 4 and 5.

2. Delete all row data from the base table.

   Execute the TRUNCATE TABLE statement to delete all row data from the base table.

3. Add a column to the base table.

   Add a column to the base table by re-executing the `ALTER TABLE` statement.

4. Edit the input data file.

   Add the data to be stored in the column added in step 3 to the input data file created in step 1.

5. Store data in the base table.

   Execute the `adbimport` command to store in the base table the data in the input data file that was edited in step 4.

Note

For details about the status in which segments for storing rows are assigned, see 5.3.1 Notes on defining B-tree indexes (unfinished status of B-tree indexes).

15.9 Problems related to B-tree indexes

This section explains the steps to take when a problem occurs in a B-tree index.

15.9.1 Steps to take when unfinished status is applied to a B-tree index

If you define a B-tree index for a base table to which segments for storing rows are assigned, that B-tree index is placed in unfinished status (a status in which no B-tree index data can be created). For details about the status in which segments for storing rows are assigned, see 5.3.1 Notes on defining B-tree indexes (unfinished status of B-tree indexes).

When a B-tree index is placed in unfinished status, the following operations end in an error.

■ Operations that end in an error

- Executing a `SELECT` statement that uses an unfinished B-tree index
- Executing the `INSERT`, `UPDATE`, or `DELETE` statement on a table for which an unfinished B-tree index is defined
- Executing the `adbimport` command without the `-d` option specified on a base table for which an unfinished B-tree index is defined
- Executing the `adbmergechunk` command on a base table for which an unfinished B-tree index is defined
- Executing the `adbunarchivechunk` command on an archivable multi-chunk table for which a B-tree index in unfinished status is defined

> **Note**
>
> The `TRUNCATE TABLE` statement can be executed. If the `TRUNCATE TABLE` statement is used to delete all rows from a base table, B-tree index data is also deleted, and therefore, the B-tree index is released from unfinished status.

A B-tree index is placed in unfinished status for the following reasons:

- You define a B-tree index for a base table to which segments for storing rows are assigned.
- The `adbimport` or `adbidxrebuild` command terminated abnormally.

The way you release a B-tree index from unfinished status depends on the cause of the unfinished status.

(1) If you define a B-tree index for a base table to which segments for storing rows are assigned

1. Check whether the B-tree index is in unfinished status.

   You can use the `adbdbstatus` command to check whether a B-tree index is in unfinished status. See 10.9.3 Checking the status and usage of a B-tree index.

   If the B-tree index is in unfinished status, proceed to step 2.

2. Release the B-tree index from unfinished status.

   Execute the `adbidxrebuild` command on the base table for which the unfinished B-tree index is defined.

   Rebuilding a B-tree index by executing the `adbidxrebuild` command releases the B-tree index from unfinished status.

If the CREATE INDEX statement is executed after the adbimport command has terminated abnormally, B-tree indexes might be placed in unfinished status. In this case, execute the adbidxrebuild command with the --force option specified. If the adbidxrebuild command is executed without the --force option specified, an error will result.

For details about the adbidxrebuild command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

(2) When the adbimport or adbidxrebuild command has terminated abnormally

1. Check whether the B-tree index is in unfinished status.

   You can use the adbdbstatus command to check whether a B-tree index is in unfinished status. See 10.9.3 Checking the status and usage of a B-tree index.

   If the B-tree index is in unfinished status, proceed to step 2.

2. Release the B-tree index from unfinished status.

   If the adbimport or adbidxrebuild command terminates abnormally, re-execute the respective command on the base table for which the unfinished B-tree index is defined.

   When the re-executed adbimport or adbidxrebuild command terminates normally, the B-tree index is released from unfinished status.

15.9.2 Steps to take when the uniqueness constraint is violated (when the KFAA61205-W message is output)

When the adbimport or adbidxrebuild command is executed on a table for which a unique index is defined, processing of each command continues even if there are duplicate key values and the uniqueness constraint is violated. When the adbimport or adbidxrebuild command terminates, the KFAA61205-W message is output, indicating violation of the uniqueness constraint.

A unique index that violates the uniqueness constraint is treated as a B-tree index that does not have the uniqueness constraint. This status is called *uniqueness constraint violation*. When uniqueness constraint violation occurs, the search efficiency might be inferior to a case in which the uniqueness constraint is satisfied.

(1) Checking whether a uniqueness constraint violation has occurred

Besides checking the KFAA61205-W message, you can execute the adbdbstatus command to check whether a unique index is in uniqueness constraint violation. For details, see 10.9.3 Checking the status and usage of a B-tree index.

(2) Steps to take when the uniqueness constraint is violated

To release a unique index from uniqueness constraint violation, take the steps described in the following procedure:

1. Search for duplicate key values

   Search for duplicate key values by executing the SELECT statement on the table for which the target unique index is defined. The following shows a specification example of the SELECT statement.

   ■ SELECT statement specification example

   This example searches for key values that violate the uniqueness constraint when unique index IX1 is defined, with columns C1 and C2 of table T1 as indexed columns.

```
SELECT "C1","C2" FROM "T1"
   WHERE "C1" IS NOT NULL AND "C2" IS NOT NULL GROUP BY "C1","C2"
     HAVING COUNT(*) >= 2
```

2. Delete rows that have duplicate key values.

   Use the `DELETE` statement to delete rows with key values that violate the uniqueness constraint.

   For details about the `DELETE` statement, see *DELETE (delete rows)* in *Data Manipulation SQL* in the manual *HADB SQL Reference*.

3. Rebuild a B-tree index.

   Execute the `adbidxrebuild` command to rebuild a B-tree index. When the `adbidxrebuild` command finishes, you can cancel uniqueness constraint violation.

   For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

Note

Uniqueness constraint violation is also canceled when either of the following operations is used to delete all data from a table for which a unique index is defined that violates the uniqueness constraint:

- Executing the `adbimport` command with the `-d` option (creation mode) specified
- Executing the `TRUNCATE TABLE` statement

15.10 Problems related to text indexes

This section explains the steps to take when a problem occurs in text indexes.

15.10.1 Steps to take when unfinished status is applied to a text index

If you define text index for a base table to which segments for storing rows are assigned, that text index is placed in unfinished status (a status in which no text index data can be created). For details about the status in which segments for storing rows are assigned, see 5.4.3 Notes on defining text indexes (unfinished status of text indexes).

When text indexes have been placed in unfinished status, the following operations will terminate in an error.

■ Operations that terminate in an error

- Executing a `SELECT` statement that uses an unfinished text index
- Executing an `INSERT`, `UPDATE`, or `DELETE` statement on a base table for which an unfinished text index is defined
- Executing the `adbimport` command without the `-d` option specified on a base table for which an unfinished text index is defined
- Executing the `adbmergechunk` command on a base table for which an unfinished text index is defined
- Executing the `adbunarchivechunk` command on an archivable multi-chunk table for which a text index in unfinished status is defined

> **Note**
>
> The `TRUNCATE TABLE` statement can be executed. If the `TRUNCATE TABLE` statement is used to delete all rows from a base table, text index data is also deleted, and therefore, the text index is released from unfinished status.

A text index is placed in unfinished status for the following reasons:

- You define a text index for a base table to which segments for storing rows are assigned.
- The `adbimport` or `adbidxrebuild` command has terminated abnormally.

How you release a text index from unfinished status depends on the cause of the unfinished status.

(1) If you define a text index for a base table to which segments for storing rows are assigned

1. Check whether the text index is in unfinished status.
   You can use the `adbdbstatus` command to check whether a text index is in unfinished status. See 10.9.4 Checking the status and usage of a text index.
   If the text index is in unfinished status, go to step 2.
2. Release the text index from unfinished status.
   Execute the `adbidxrebuild` command on the base table for which the unfinished text index was defined.
   Rebuilding a text index by executing the `adbidxrebuild` command releases the text index from unfinished status.

If the `CREATE INDEX` statement is executed after the `adbimport` command has terminated abnormally, the text index might be placed in unfinished status. In this case, execute the `adbidxrebuild` command with the `--force` option specified. If the `adbidxrebuild` command is executed without the `--force` option specified, an error will result.

For details about the `adbidxrebuild` command, see *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

(2) When the adbimport or adbidxrebuild command has terminated abnormally

1. Check whether the text index is in unfinished status.

   You can use the `adbdbstatus` command to check whether a text index is in unfinished status. See 10.9.4 Checking the status and usage of a text index.

   If the text index is in unfinished status, go to step 2.

2. Release the text index from unfinished status.

   Re-execute the command that terminated abnormally (`adbimport` or `adbidxrebuild`) on the base table for which the unfinished text index was defined.

   When the re-executed command terminates normally, the text index will have been released from unfinished status.

15.11 Problems related to range indexes

This section explains the steps to take when a problem occurs in a range index.

15.11.1 Steps to take when unfinished status is applied to a range index

If you define a range index for a base table to which segments for storing rows are assigned, that range index is placed in unfinished status (a status in which no range index data can be created). For details about the status in which segments for storing rows are assigned, see 5.5.4 Notes on defining range indexes (unfinished status of range indexes).

When a range index is placed in unfinished status, the following operations end in an error:

■ Operations that end in an error

- Executing the `SELECT` statement using an unfinished range index
- Executing the `INSERT` or `UPDATE` statement on a table for which an unfinished range index is defined
- Executing the `adbimport` command without the `-d` option specified on a base table for which an unfinished range index is defined
- Executing the `adbmergechunk` command on a base table for which an unfinished range index is defined
- Executing the `adbunarchivechunk` command on an archivable multi-chunk table for which a range index in unfinished status is defined

> **Note**
>
> The `TRUNCATE TABLE` statement can be executed. If the `TRUNCATE TABLE` statement is used to delete all rows from a base table, range index data is also deleted, and therefore, the range index is released from unfinished status.

You can use the `adbdbstatus` command to check whether a range index is in unfinished status. See 10.9.5 Checking the status and usage of range indexes.

A range index goes into unfinished status for one of the following reasons:

- You define a range index for a base table to which segments for storing rows are assigned.
- The `adbidxrebuild` command terminated abnormally

To release the range index from unfinished status, execute the `adbidxrebuild` command on the base table for which the unfinished range index is defined.

Rebuilding a range index by executing the `adbidxrebuild` command releases the range index from unfinished status.

If the `CREATE INDEX` statement is executed after the `adbimport` command has terminated abnormally, the range index might be placed in unfinished status. In this case, execute the `adbidxrebuild` command with the `--force` option specified. If the `adbidxrebuild` command is executed without the `--force` option specified, an error will result.

15.12 Problems related to the background-import facility

This section explains the steps to take when the KFAA51246-E message is issued.

The KFAA51246-E message is output when the number of chunks that are created during background import exceeds the maximum number of chunks allowed (the value specified for CHUNK in the CREATE TABLE statement).

Therefore, first reduce the number of created chunks based on the explanation in 15.12.1 Reducing the number of chunks that were created during background import.

To increase the maximum number of chunks that can be created, see 15.12.2 Increasing the maximum number of chunks that can be created during background import.

If you cannot change the number of created chunks, see 15.12.3 Steps to take when the number of chunks that are created during background import cannot be changed.

15.12.1 Reducing the number of chunks that were created during background import

If the KFAA51246-E message is output during execution of the adbimport command using the background-import facility, execute the adbmergechunk command.

By using the adbmergechunk command to merge multiple created chunks into a single chunk, you can reduce the number of chunks that were created. Once you have reduced the number of chunks, you can perform background import.

■ **Steps to take**

Execute the adbmergechunk command based on the explanation in 11.4.9 Merging chunks (to reduce the number of chunks).

15.12.2 Increasing the maximum number of chunks that can be created during background import

If the KFAA51246-E message is output during execution of the adbimport command using the background-import facility, and you want to reduce the number of created chunks and increase the maximum number of chunks that can be created, execute the ALTER TABLE statement.

By executing the ALTER TABLE statement, you can change the value of CHUNK that was specified when the base table was defined, and increase the maximum number of chunks that can be created.

■ **Steps to take**

Execute the ALTER TABLE statement based on the explanation in 11.4.10 Changing the maximum number of chunks.

15.12.3 Steps to take when the number of chunks that are created during background import cannot be changed

This subsection explains the steps to take when the message KFAA51246-E is output during execution of the adbimport command using the background-import facility, and neither of the procedures described in the following subsections can be performed:

- 15.12.1 Reducing the number of chunks that were created during background import
- 15.12.2 Increasing the maximum number of chunks that can be created during background import

■ Steps to take

First, output data that was stored using background import based on the explanation in 11.4.5 Exporting data in units of chunks.

Then, take one of the following steps:

- **Delete the unnecessary chunks.**
  Execute the PURGE CHUNK statement based on the explanation in 11.4.6 Deleting data in units of chunks. Once you have reduced the number of created chunks by deleting unnecessary chunks, you can perform background import.
- **Import data in the creation mode.**
  Re-import the data that you output before by executing the adbimport command with the -d option (creation mode) specified. Executing the adbimport command in the creation mode deletes all existing data (chunks) from a table, and imports data. Because the number of created chunks is 1, you can perform background import.
  For details about the adbimport command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

15.13 Problems related to archive directories

This section explains the steps to take when a problem occurs in an archive directory.

15.13.1 Steps to take when a failure occurs in an archive file

If a failure occurs in an archive file, one of the following messages is output.

- KFAA41205-E
- KFAA50225-E
- KFAA51271-E
- KFAA51272-E

Take the corrective action described in the Action section for the output message. For details about the Action section for messages, see the manual *HADB Messages*.

If you cannot correct the failure by taking the corrective action described in the Action section for the message, or if you cannot take that action, you need to recover the database from the backup. If a full backup of a database that was collected according to 10.3.1 Backup acquisition method is available, recover the database from the latest full backup by following 10.3.2 Recovering the database from the backup.

15.13.2 Steps to take if free space in the archive directory is insufficient

If the adbarchivechunk command terminated abnormally due to insufficient free space in the archive directory, change the archive directory to another disk having a larger capacity. The following describes the procedure.

Procedure:

1. Terminate the HADB server normally.
   Execute the adbstop command to terminate the HADB server normally.
2. Copy the files under the archive directory.
   Execute the OS's cp command to copy all the files under the archive directory to a disk having a larger capacity.
3. Mount the disk in step 2 to the archive directory.
   Execute the OS's mount command to mount the disk in step 2 to the archive directory.
4. Start the HADB server normally.
   Execute the adbstart command to start the HADB server normally.

15.14 Problems related to synonym dictionary files

This section explains the steps to take when a problem occurs in a synonym dictionary file.

15.14.1 Steps to take when a failure occurs in a synonym dictionary file

If a failure occurs in a synonym dictionary file, take one of the following corrective actions. First, see Action 1.

- **Action 1**

  If a failure occurs in a synonym dictionary file, the following messages are output:

  - KFAA30959-E
  - KFAA34008-E

  In this case, follow the corrective action described in the Action section for the output message. If you cannot correct the problem by taking the corrective action described in the Action section for the message, or if you cannot take that action, take the action in Action 2.

- **Action 2**

  Recover the synonym dictionary file by using the backup of the synonym dictionary file. If you do no have a backup of the synonym dictionary file, take the action in Action 3.

- **Action 3**

  Re-create the synonym dictionary file in which a failure occurred by using the adbsyndict command. Execute the adbsyndict command by using the saved dictionary creation file as the input information.

  If you cannot re-create the synonym dictionary file because, for example, you do not have the dictionary creation file, take the action in Action 4.

- **Action 4**

  Restore the database from the latest full backup of the database.

> **Note**
>
> The synonym dictionary file is stored under the directory specified in the adb_syndict_storage_path operand in the server definition.

15.14.2 Steps to take if free space in the directory for storing synonym dictionary files is insufficient

If the adbsyndict command terminated abnormally due to insufficient free space in the storage directory for synonym dictionary files, the KFAA51514-E message is output. In the KFAA51514-E message, if the operation target information is synonym-dictionary-file or temporary-work-file and the error number is 28 (ENOSPC), free space in the storage directory for synonym dictionary files might be insufficient.

In this case, perform Action 1. If you cannot perform Action 1 or if there is not enough free space even after Action 1 is performed, perform Action 2.

- **Action 1**

  If multiple synonym dictionaries are updated all at once when the adbsyndict command is executed, update the synonym dictionaries one by one. If you reduce the number of synonym dictionaries to be updated at one time, you

might be able to execute the `adbsyndict` command because less free space is required in the storage directory for synonym dictionary files.

- **Action 2**
  Create the storage directory for synonym dictionary files again on another directory having a larger capacity. For details about the procedure, see 11.16.13 Changing the directory for storing synonym dictionary files.

15.14.3 Steps to take if synonym dictionary files or their storage directory is accidentally deleted

If you have accidentally deleted synonym dictionary files or the storage directory for synonym dictionary files, you need to re-create the synonym dictionary files.

> **Important**
>
> You cannot recover synonym dictionary files or their storage directory from the backups.

(1) Steps to take if a synonym dictionary file is accidentally deleted

If you have accidentally deleted synonym dictionary files, you need to re-create them. The following describes the procedure.

Procedure:

1. Prepare the synonym list definition files you have stored.
   You need the synonym list definition files corresponding to the synonym dictionary files that were deleted accidentally.
2. Prepare the dictionary creation files that you have stored.
   You need the dictionary creation files corresponding to the synonym dictionary files that were deleted accidentally.
3. Execute the `adbsyndict` command to re-register the synonym dictionaries.
   At this time, although the `KFAA51504-W` message is output, no action is required.

> **Note**
>
> - When registration of the synonym dictionaries is completed in step 3, the `KFAA51509-I` message is output (this message is output as many times as the number of synonym dictionaries that have been registered). Check the synonym dictionary name in the message to confirm that the synonym dictionary has been registered.
> - If you have lost a synonym list definition file, rebuild the synonym list definition file based on the explanation in 11.16.14 Rebuilding a synonym list definition file (when a synonym list definition file is lost).

(2) Steps to take if the storage directory for synonym dictionary files is accidentally deleted

If you have accidentally deleted the storage directory for synonym dictionary files, you need to re-create all the synonym dictionary files. The following describes the procedure.

Procedure:

1. Re-create the storage directory for synonym dictionary files.
   Re-create the storage directory for synonym dictionary files that was deleted accidentally.
2. Assign the permissions to the storage directory for synonym dictionary files.
   For details about the permissions to be granted, see (5) Assigning permissions to the directory for storing synonym dictionary files in 11.16.1 Preparing for synonym search operations.
3. Prepare the synonym list definition files you have stored.
   You need the synonym list definition files for all the synonym dictionaries.
4. Prepare the dictionary creation files that you have stored.
   You need the dictionary creation files for all the synonym dictionaries.
5. Execute the `adbsyndict` command to re-register all the synonym dictionaries.
   At this time, although the `KFAA51504-W` message is output, no action is required.

Note

- When registration of the synonym dictionaries is completed in step 5, the `KFAA51509-I` message is output (this message is output as many times as the number of synonym dictionaries that have been registered). Check the synonym dictionary name in the message to confirm that the synonym dictionary has been registered.
- If you have lost a synonym list definition file, rebuild the synonym list definition file based on the explanation in 11.16.14 Rebuilding a synonym list definition file (when a synonym list definition file is lost).

15.15 Problems related to unload files

This section explains the steps to take when a problem occurs in an unload file.

15.15.1 Steps to take in the event of a shortage of disk space for storing unload files

Unload files are created in the directory specified in the `-f` option while the `adbreorgsystemdata` command is being executed.

If one of the following messages is output during execution of the `adbreorgsystemdata` command, a shortage might have occurred in the disk space for storing unload files:

- `KFAA30959-E`
- `KFAA40204-E`
- `KFAA40205-E`
- `KFAA40214-E`
- `KFAA41205-E`
- `KFAA41206-I`

You can determine whether a shortage has occurred in the disk space for storing unload files from the information provided in the message. For details, see (1) How to determine whether a shortage has occurred in the disk space for storing unload files.

The following are possible causes of a shortage of disk space for storing unload files. Take the corrective action appropriate to the cause.

- There are unneeded unload files on the disk.
  See (2) If there are unneeded unload files on the disk.
- The disk capacity for storing unload files is too small to store unload files that will be created.
  See (3) If the disk capacity for storing unload files is too small .

(1) How to determine whether a shortage has occurred in the disk space for storing unload files

To determine whether a shortage has occurred in the disk space for storing unload files, check the output message. If the output message is any of those listed in the following table, a shortage has occurred in the disk space for storing unload files.

Table 15-2: Messages that are output when a shortage has occurred in the disk space for storing unload files and the information provided in the messages

No.	Message ID	Information provided in the message	
		File name	Error number
1	`KFAA30959-E`	File name[#1]	`28 (ENOSPC)`
2	`KFAA40204-E`	File path name[#2]	--

No.	Message ID	Information provided in the message	
		File name	Error number
3	KFAA40205-E	--	28(ENOSPC)
4	KFAA40214-E	--	28(ENOSPC)
5	KFAA41205-E	File path name#2	28(ENOSPC)
6	KFAA41206-I	File path name#2	28(ENOSPC)

Legend:

--: This information is not output in the corresponding message.

#1

Check the directory that contains the unload files indicated in the message. The following steps explain how to check the directory that contains a specific file.

1. Use the OS's `find` command to locate the file whose file name is displayed in the message.

   The following shows an example of the OS's `find` command.

   ▪ **Execution example of the OS's `find` command**

   ```
   find ~/ -name "xxxxx"
   ```

   Explanation

   Execute the `find` command by specifying the file name (*xxxxx*) displayed in the message.

2. Check the `adbreorgsystemdata` command's execution results to determine whether the directory storing the unload file matches the directory specified in the `-f` option.

   If the directories match, a shortage has occurred in the disk space for storing unload files.

#2

Check the directory that contains the unload file based on the path name displayed in the message.

Check whether the directory contained in the path name matches the directory specified in the `-f`option of the `adbreorgsystemdata` command.

If the directories match, a shortage has occurred in the disk space for storing unload files.

(2) If there are unneeded unload files on the disk

Unload files remaining on the disk are all unneeded. Delete the unneeded unload files by executing, for example, the OS's `rm` command.

Then, re-execute the `adbreorgsystemdata` command.

(3) If the disk capacity for storing unload files is too small

If the disk capacity for storing unload files is too small, you need to change the storage location for unload files to another disk with a greater capacity. Take the steps described in the following procedure.

Procedure:

1. Re-estimate the size of an unload file.

   Estimate the size of an unload file that is created when the `adbreorgsystemdata` command is executed. For details about how to estimate the size of an unload file, see 6.20 Estimating the size of an unload file.

Then, store the unload file in a disk whose capacity is larger than the value you estimated. To change the directory specified in the -f option of the adbreorgsystemdata command, go to step 2. If you do not change the directory specified in the -f option, go to step 3.

2. Change the directory specified in the -f option.

   In the directory path file specified in the -f option of the adbreorgsystemdata command, change the directory specifying the storage location for unload files to a directory on another disk with a greater capacity.

   The following shows examples of directory file path specifications before and after the directory specified in the directory path file is changed.

   ▪ Example of directory path file specifications before change

   ```
   /mnt/disk1/xxxxx
   ```

   Explanation

   In the directory path file before change, a directory on a disk having insufficient capacity ('/mnt/disk1') is specified as the storage location for unload files.

   ▪ Specification example of the directory file path (after change)

   ```
   /mnt/largedisk1/xxxxx
   ```

   Explanation

   This example changes the storage location for unload files by specifying a different disk with a greater capacity (/mnt/largedisk1) in place of /mnt/disk1.

   Then re-execute the adbreorgsystemdata command.

3. Create a symbolic link to the directory on the disk with the greater capacity for the directory specified in the -f option.

   Execute the OS's ln command for the *directory specifying the storage location for unload files* in the directory path file specified in the -f option of the adbreorgsystemdata command. Use the ln command to create a symbolic link to a directory on another disk with greater capacity.

   In this case, specify, as the path name of the symbolic link, the *directory specifying the storage location for unload files* in the directory path file specified in the -f option. The following shows an example of the OS's ln command for creating a symbolic link.

   ▪ Example of the OS's ln command

   ```
   ln  -s /mnt/largedisk1/xxxxx  /mnt/disk1/xxxxx
   ```

   Explanation

   This example specifies a directory ('/mnt/largedisk1/xxxxx) on a disk with a greater capacity as the link target, and creates a symbolic link (/mnt/disk1/xxxxx) with the same name as the directory (/mnt/disk1/xxxxx) specified in the -f option.

   Then, re-execute the adbreorgsystemdata command.

15.16 Problems related to the message log file

This section describes the steps to take when problems related to the message log file occur.

15.16.1 Releasing the message log file from fall-back mode

This subsection describes how to release the message log file from fall-back mode. For details about fall-back mode of the message log file, see 10.4.4 Fall-back mode of the message log file.

> **Note**
>
> The message log file is automatically released from fall-back mode in the following condition. An SQL statement or command is executed when the free space of the disk that stores the message log file is larger than or equal to the following value: the value specified (in megabytes) for environment variable ADBMSGLOGSIZE × 2.

■ Releasing the message log file on the HADB server from fall-back mode

Perform the following procedure.

Procedure:

1. Execute the following command to check the amount of free space on the disk storing the message log file:

```
df $ADBDIR/spool
```

2. Reserve free space on the disk, by, for example, deleting unnecessary files. At least the following amount of free space is required:

```
Value specified for environment variable ADBMSGLOGSIZE (megabytes) x 2
```

■ Releasing the message log file on an HADB client from fall-back mode

Perform the following procedure.

Procedure (for an HADB client (for Windows)):

1. From the command prompt, execute the following command to check the amount of free space on the disk drive storing the message log file:

```
echo %ADBCLTDIR%
```

2. Check the amount of free space on the disk drive storing the message log file.
3. Reserve free space on the disk drive by, for example, deleting unnecessary files. At least the following amount of free space is required:

```
Value specified for environment variable ADBMSGLOGSIZE (megabytes) x 2
```

Procedure (for an HADB client (for Linux)):

1. Execute the following command to check the amount of free space on the disk storing the message log file:

```
df $ADBCLTDIR/spool
```

2. Reserve free space on the disk, by, for example, deleting unnecessary files. At least the following amount of free space is required:

```
Value specified for environment variable ADBMSGLOGSIZE (megabytes) x 2
```

15.17 Problems related to restart of the HADB server

This section describes the steps to take when problems related to restart of the HADB server occur.

15.17.1 Steps to take when the processing time for restarting the HADB server takes too long

This subsection describes what to do when the processing time for restarting the HADB server takes too long.

■ **What to do while the HADB server is restarting**

After the `adbstart` command is executed, if the processing for restarting the HADB server takes more time than you expected, check whether the `KFAA81215-I` message has been output. In the `KFAA81215-I` message, you can check the progress of database recovery processing.

Based on the information in the `KFAA81215-I` message, estimate the time required for database recovery processing to complete and subsequent processing to resume.

Note that the `KFAA81215-I` message is output at regular intervals, according to the value specified for the `adb_log_rec_msg_interval` operand in the server definition.

■ **What to do after the processing for restarting the HADB server is complete**

After the processing for restarting the HADB server is complete, check whether the `KFAA81211-I` message has been output. In the `KFAA81211-I` message, you can check the contents of transaction recovery processing.

Check the information output to the `KFAA81211-I` message and consider reducing the number of updates per transaction. For example, consider increasing the number of commits.

Note that the `KFAA81211-I` message is output if there is a transaction that took longer to recover than the value specified for the `adb_log_rec_msg_interval` operand in the server definition.

Based on the transaction ID that is output to the `KFAA81211-I` message, you can identify the SQL statement that was executed for the corresponding transaction. Use the transaction ID as the key, and check the value for `tran_id` that was output to the "SQL statement execution information" in the SQL trace information. The SQL statement whose `tran_id` matches the transaction ID is the SQL statement executed for the corresponding transaction. For details about the SQL statement execution information, see (2) SQL statement execution information in 10.11.2 Information that is output as SQL trace information.

For details about the `adb_log_rec_msg_interval` operand in the server definition, see the description of the adb_log_rec_msg_interval operand in 7.2.3 Operands related to system logs (set format).

16 Operations When Using the Multi-Node Function

This chapter explains how to build and perform operations in a system that uses the multi-node function.

For an overview of the multi-node function, see 2.19 Multi-node function.

16.1 Notes on reading this chapter

The explanation of the multi-node function in this chapter assumes that the reader has a basic knowledge of HA Monitor. For details about HA Monitor, see the manual *HA Monitor for Linux(R) (x86)*.

When reading the manual *HA Monitor for Linux(R) (x86)*, note the following points:

- The HADB server operates in monitor mode.
- The configuration inherits the alias IP address.
- The HADB server runs as a program that does not have an interface to HA Monitor.
- Server failures are detected without issuing an API.
- HA Monitor Extension cannot be used.
- The following table shows the correspondence between the terminology used in HA Monitor manuals and the terminology used in HADB manuals.

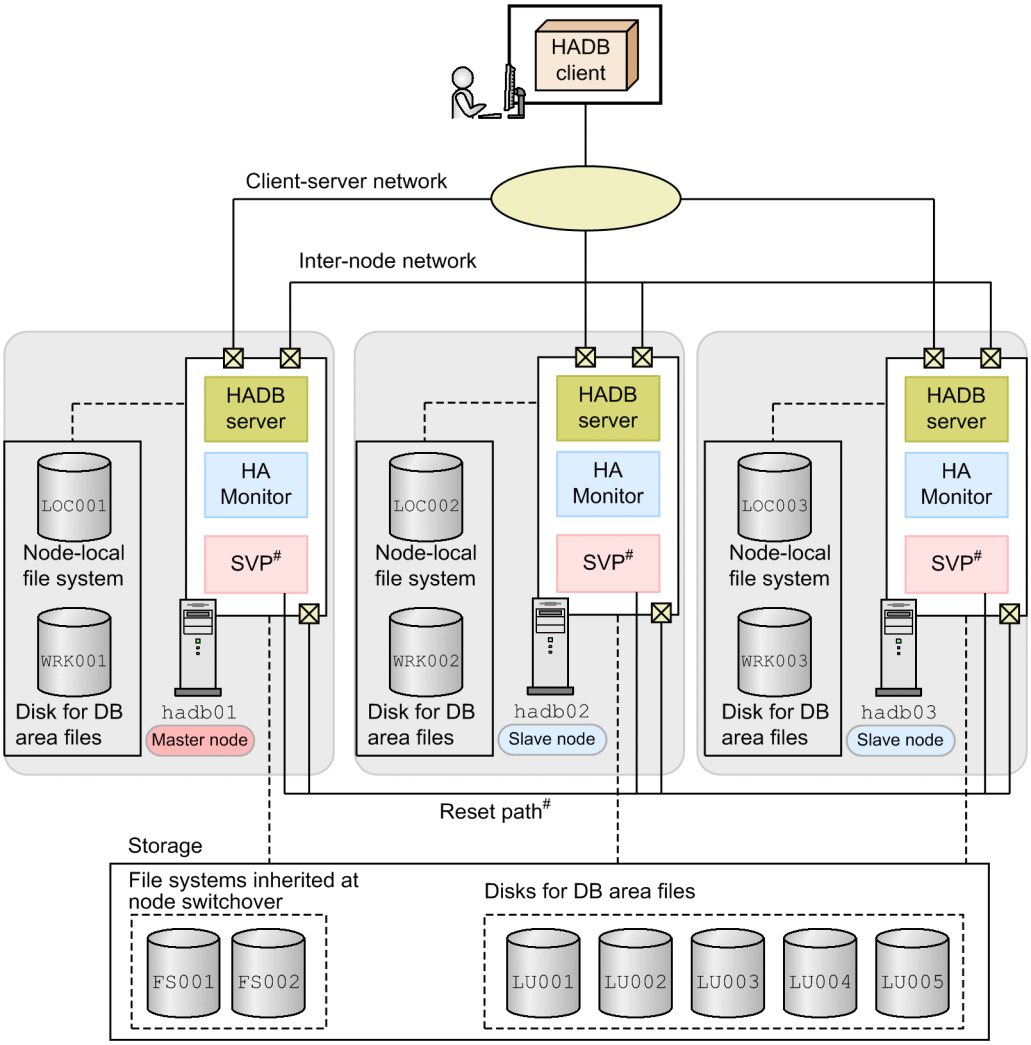
Table 16-1: Correspondence between the terminology used in HA Monitor manuals and the terminology used in HADB manuals

No.	Terminology used in HA Monitor manuals	Terminology used in HADB manuals
1	Host	Node
2	Active system	Master node
3	Standby system	Slave node
4	Active server	HADB server on the master node
5	Standby server	HADB server on a slave node
6	Host failure	Node failure
7	Server failure	

16.2 System configuration example that uses the multi-node function

The following figure shows an example of a system configuration using the multi-node function.

Figure 16-1: Example of a system configuration using the multi-node function



#

When using the SCSI reservation for shared disk method, neither an SVP nor a reset path are required.

Important

You can select either of the following as the shared disk data protection method that uses HA Monitor. We recommend you use host reset.

- Host reset
- SCSI reservation for shared disk

16.2.1 Prerequisite software program

Use of the multi-node function requires HA Monitor.

HADB uses the HA Monitor's monitor mode function to monitor node failures. If a failure occurs in the master node, the HA Monitor function is used to switch over one of the slave nodes to the master node, in order to allow operations to continue. If a node failure occurs for a slave node, the HA Monitor function is used to separate the slave node in which the problem occurred from the multi-node configuration in order to allow operations to continue by using the remaining nodes.

For details on the separation of a slave node in the event of a node failure, see 16.15 Operations when a node failure occurs.

Note that depending on the version of Linux you are using, the prerequisite version of HA Monitor differs as follows:

- Red Hat Enterprise Linux Server 6 (64-bit x86_64): 01-56-01 or later
- Red Hat Enterprise Linux Server 7 (64-bit x86_64): 01-61 or later

Important

Make sure the version of HA Monitor is the same on all nodes.

16.2.2 Server configuration

The example configuration shown in Figure 16-1: Example of a system configuration using the multi-node function consists of three nodes (`hadb01`, `hadb02`, and `hadb03`. The master node is `hadb01`, and `hadb02` and `hadb03` are slave nodes.

When you use the multi-node function, the server machines on the individual nodes do not need to have the same performance (in CPU and memory size, for example). However, if performance varies among the individual server machines, the processing performance for SQL statements might also vary depending on the connected node. Therefore, we recommend that you keep the performance of the server machines on all nodes as close to the same as possible.

Note that the multi-node function uses the hot standby function of HA Monitor. Select one of the following as the shared disk data protection method used when switching hosts. We recommend that you select host reset.

- Host reset
- SCSI reservation for shared disk

Use the recommended host reset method when there are failure management processors (the SVPs shown in Figure 16-1: Example of a system configuration using the multi-node function) deployed on the server machines in all nodes.

For details about host reset, failure management processors, and SCSI reservation for shared disk, see the following sections in the manual *HA Monitor for Linux(R) (x86)*:

- Host reset: *Host reset*
- Failure management processor: *Required hardware*
- SCSI reservation for shared disk: *SCSI reservation for shared disk*

16.2.3 Network configuration

The multi-node function uses the three networks listed below. Keep these networks physically separated.

- Client-server network
- Inter-node network
- Reset path

(1) Client-server network

This network is used for communication between HADB clients and HADB servers.

An HADB client uses an alias IP address to connect to the HADB server. Therefore, set the alias IP addresses based on the explanation in *Inheriting a LAN* in the manual *HA Monitor for Linux(R) (x86)*.

The following table shows an example of IP address and port number settings for the system configuration shown in Figure 16-1: Example of a system configuration using the multi-node function:

Table 16-2: Setting examples for IP addresses and port numbers in a client-server network

No.	Setting location	IP address	Port number
1	Client machine	`10.196.108.111`	Setting not required
2	Server machine `hadb01` (master node)	`10.196.108.11`	`23650`
3	Server machine `hadb02` (slave node)	`10.196.108.12`	`23650`
4	Server machine `hadb03` (slave node)	`10.196.108.13`	`23650`
5	Alias IP address	`10.196.108.143`	`23650`

(2) Inter-node network

This network is used for communication between HADB servers, and as HA Monitor's monitoring path.

When you build a network, we recommend that you achieve NIC redundancy using Linux's bonding function. Furthermore, when you use the bonding function, have multiple LAN cards available.

For details about the bonding function, see the documentation for the operating system you are using. For details about how to set HA Monitor's monitoring path, see *Configuring the monitoring paths* in the manual *HA Monitor for Linux(R) (x86)*.

The following table shows an example of IP address and port number settings for the system configuration shown in Figure 16-1: Example of a system configuration using the multi-node function:

Table 16-3: Setting examples for IP addresses and port numbers in an inter-node network

No.	Setting location	IP address	Port number
1	HADB inter-server communication for server machine `hadb01` (master node)	`172.16.0.11`	`23651`
2	HA Monitor's monitoring path for server machine `hadb01` (master node)	`172.16.0.11`	`7777`
3	HADB inter-server communication for server machine `hadb02` (slave node)	`172.16.0.12`	`23651`
4	HA Monitor's monitoring path for server machine `hadb02` (slave node)	`172.16.0.12`	`7777`

No.	Setting location	IP address	Port number
5	HADB inter-server communication for server machine `hadb03` (slave node)	`172.16.0.13`	`23651`
6	HA Monitor's monitoring path for server machine `hadb03` (slave node)	`172.16.0.13`	`7777`

Normally, the network used for communication between HADB servers is separated from the network used as HA Monitor's monitoring path. However, in such a case, HA Monitor cannot detect failures in the network used for communication between HADB servers. When the same network is used for communication between HADB servers and for HA Monitor's monitoring path as indicated in the table above, HA Monitor can detect network failures. Therefore, we recommend that you configure the inter-node network as shown above.

However, because the target network becomes a single point of failure, you need to enhance the network's resiliency by providing a redundant LAN adapter. You can do this by using a program such as the bonding package in Linux, which is a kernel extension package.

(3) Reset path

When using host reset, a reset path is required. SCSI reservation for shared disk does not require a reset path.

A reset path is a network used, when HA Monitor detects a failure, to reset the input/output of the node on which the failure occurred. This network needs to be dedicated to the reset path. For details, see *Reset path configuration* in the manual *HA Monitor for Linux(R) (x86)*.

The following table shows an example of IP address and port number settings for the system configuration shown in Figure 16-1: Example of a system configuration using the multi-node function:

Table 16-4: Setting examples for IP addresses and port numbers in a reset path

No.	Setting location	IP address	Port number
1	Reset path of server machine `hadb01` (master node)	`192.168.0.11`	`11111`
2	SVP of server machine `hadb01` (master node)	`192.168.0.21`	`22222`
3	Reset path of server machine `hadb02` (slave node)	`192.168.0.12`	`11111`
4	SVP of server machine `hadb02` (slave node)	`192.168.0.22`	`22222`
5	Reset path of server machine `hadb03` (slave node)	`192.168.0.13`	`11111`
6	SVP of server machine `hadb03` (slave node)	`192.168.0.23`	`22222`

As the shared disk data protection method used by HA Monitor, we recommend that you use host reset. In this case, a reset path is required. The server machine on which the HADB server operates must also be equipped with a failure management processor such as an SVP. For details, see *Required hardware* in the manual *HA Monitor for Linux(R) (x86)*.

16.2.4 Storage configuration

When you use the multi-node function, provide the following file systems and disks.

- **Node-local file system**

  In Figure 16-1: Example of a system configuration using the multi-node function, `LOC001` to `LOC003` are node-local file systems.

- **File system inherited at node switchover**

  In Figure 16-1: Example of a system configuration using the multi-node function, `FS001` and `FS002` are file systems inherited at node switchover.

- **Disks for DB area files**

  In Figure 16-1: Example of a system configuration using the multi-node function, `LU001` to `LU005` and `WRK001` to `WRK003` are disks for DB area files.

Note that some file systems and disks require device names that allow the disk to be identified on each node. If the path between the server and storage is a single-path configuration, we recommend that you use WWNs. When implementing redundant paths between server and storage, the device name will differ according to the redundancy method being used. Examples of device names in Figure 16-1: Example of a system configuration using the multi-node function are as follows:

- Examples of device names in single-path configuration

  `FS001: /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b53e0a`
  `FS002: /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b6c44d`
  `LU001: /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b828e9`
  `LU002: /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b7d9fd`
  `LU003: /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b87793`
  `LU004: /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b8c6d3`
  `LU005: /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b9160f`

- Examples of device names in redundant configuration using multipath software (DMMP)

  `FS001: /dev/mapper/mpath1`
  `FS002: /dev/mapper/mpath2`
  `LU001: /dev/mapper/mpath11`
  `LU002: /dev/mapper/mpath12`
  `LU003: /dev/mapper/mpath13`
  `LU004: /dev/mapper/mpath14`
  `LU005: /dev/mapper/mpath15`

(1) Node-local file system

A node-local file system stores the following directories:

- Server directory

  To create a server directory, use the `adbinstall` command.

- DB directory

  For details about how to create the DB directory, see 16.3.8 Creating a database.

- Multi-node synonym dictionary storage directory

  For details about the multi-node synonym dictionary storage directory, see (6) Creating the multi-node synonym dictionary storage directory in 16.24.1 Preparing for synonym search operations.

- Audit trail directory

  For details about the audit trail directory, see 2.18.5 Output destination of audit trails (audit trail file).

- Output-directory for common format audit trails

  For details about the output-directory for common format audit trails, see 2.18.9 Conversion of audit trail information (linkage with JP1/Audit Management - Manager).

In Figure 16-1: Example of a system configuration using the multi-node function, LOC001 to LOC003 are node-local file systems.

- LOC001
  Local file system of server machine hadb01 (master node)
- LOC002
  Local file system of server machine hadb02 (slave node)
- LOC003
  Local file system of server machine hadb03 (slave node)

(2) File systems inherited at node switchover

Configure the following file systems as file systems inherited at node switchover:

- File system for the system directory which stores the content of ADBSYS under the DB directory
- File system that stores synonym dictionary files

Create these file systems inherited at node switchover on an LV in a VG that consists of disks (shared disks) that can be referenced from all nodes. For details about shared disks, see *Shared disk configuration* in the manual *HA Monitor for Linux(R) (x86)*.

In Figure 16-1: Example of a system configuration using the multi-node function, FS001 and FS002 are the file systems inherited at node switchover.

- FS001
  File system for the system directory
  Here, the name of the VG is vg_hadb, and the name of the LV is hadb_sys.
- FS002
  File system that stores synonym dictionary files
  In this example, the name of the VG is vg_hadb02, and the name of the LV is hadb_syndict. If you do not intend to conduct synonym searches, the file system that stores synonym dictionary files is not required.

Important

When using SCSI reservation for shared disk, configure the disks where the file systems inherited at node switchover are created as subject to SCSI reservation. On disks subject to SCSI reservation, do not create any areas other than file systems inherited at node switchover.

(3) Disks for DB area files

Provide a disk for each DB area file.

For the DB area files listed below, allocate a block special file. Also, make sure that disks that correspond to block special files can be referenced from all nodes.

- Master directory DB area file
- Dictionary DB area file
- System-table DB area file

- Data DB area file

Important

When using SCSI reservation for shared disk, configure the disks that store the preceding DB area files as subject to SCSI reservation.

Provide the disk that is used by the following DB area file on each node:

- Work table DB area file

Important

- If a disk that needs to be prepared for each node (disk to store the work table DB area files) is shared and used by multiple nodes, operations are not guaranteed. Incorrect retrieval results might be displayed, or the HADB server might terminate abnormally.
- When using SCSI reservation for shared disk, do not configure the disks on which work table DB area files are stored as subject to SCSI reservation.

In Figure 16-1: Example of a system configuration using the multi-node function, `LU001` to `LU005` and `WRK001` to `WRK003` are the disks for DB area files.

- `LU001`
  DB area disk that comprises a dictionary DB area
- `LU002`
  DB area disk that comprises a master directory DB area
- `LU003`
  DB area disk that comprises a system-table DB area
- `LU004`
  DB area disk that comprises a data DB area (`ADBUTBL01`)
- `LU005`
  DB area disk that comprises a data DB area (`ADBUIDX01`)
- `WRK001`
  DB area disk that comprises the work table DB area of the HADB server on server machine `hadb01` (master node)
  This is a node-local disk. The name of the corresponding block special file is set to `/dev/mapper/WRK001`.
- `WRK002`
  DB area disk that comprises the work table DB area of the HADB server on server machine `hadb02` (slave node)
  This is a node-local disk. The name of the corresponding block special file is set to `/dev/mapper/WRK002`.
- `WRK003`
  DB area disk that comprises the work table DB area of the HADB server on server machine `hadb03` (slave node)
  This is a node-local disk. The name of the corresponding block special file is set to `/dev/mapper/WRK003`.

16.3 Building a system that uses the multi-node function

This section explains how to build a system that uses the multi-node function.

The explanation in this section assumes a new installation of HADB. If HADB is already installed and you want to use the multi-node function, see 16.19 Migrating to a system that uses the multi-node function.

16.3.1 Installing HA Monitor

Install HA Monitor on the server machines of all nodes. For details about how to install HA Monitor, see *System Configuration* in the manual *HA Monitor for Linux(R) (x86)*.

16.3.2 Installing HADB server and setting up an environment

Install HADB server on the server machines of all nodes and set up their environments. For details about how to install the HADB server and set up the environment, see 8. Building a System.

The following notes apply to HADB server installation and environment setup:

- Use the same version and revision for the HADB servers on all nodes.
- You need to create either a directory for storing communication-information files, or a settings file, on all nodes. For details about how to create a directory for storing communication-information files, or how to create a settings file, see (7) Creating the directories for storing communication-information files in 8.2.1 Tasks that must be performed before installation.
- For details about how to set the environment variables, see 8.4 Setting environment variables. Note that you need to specify the same value on each node for the following environment variables:
  - `ADBLANG`
  - `TZ`

> **Note**
>
> The path name of the HADB server's server directory (during installation and during operation) does not need to be the same on all nodes.

16.3.3 Installing HADB client and setting up the environment

For details about installing and setting up the environment for the HADB client, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

16.3.4 Setting up an HA Monitor environment

Set up an HA Monitor environment on all nodes. For details about how to set up an HA Monitor environment, see *System Configuration* in the manual *HA Monitor for Linux(R) (x86)*.

This subsection provides notes on setting up an HA Monitor environment.

(1) Setting environment variables

Set the following HADB administrator environment variable:

- `PATH`

  Add the following path so that HA Monitor commands can be executed:
  `/opt/hitachi/HAmon/bin`

(2) Specifying settings in the sysdef file

Among the operands that you need to specify for HA Monitor's `sysdef` file, the values to specify are predetermined for the following operands:

- `address`

  Specifies the order of priority for resetting nodes. The smaller the specification value, the higher the priority of resetting that node. Specify different values for the individual nodes as follows:

  - The master node needs to be the highest priority, so specify the smallest value of all the nodes.
  - Specify a different value for each slave node. The higher the reset priority of the slave node, the smaller the value you must specify. Specify higher node reset priorities for slave nodes you want to prioritize when switching over the master node.

- `monbegin_restart`

  Specify `nouse`.

- `multistandby`

  Specify `use` because HA Monitor's multi-standby function is used as the multi-node function.

- `usrcommand`

  Specify the absolute path for the user command that describes the settings for HADB.

  Create the user command based on the appropriate template. For details about how to create a user command, see (b) Creating a user command in (4) Creating environment variable definitions for commands, and creating commands, and *Creating user commands* in the manual *HA Monitor for Linux(R) (x86)*.

- `termcmd_at_abort`

  Specify `nouse`.

- `fence_reset`

  When using host reset, either omit this operand or specify the default value (`use`).

  When using SCSI reservation for shared disk, specify `nouse`.

- `fence_scsi`

  When using host reset, either omit this operand or specify the default value (`nouse`).

  When using SCSI reservation for shared disk, specify `use`.

- `fence_lan`

  Either omit this operand or specify the default value (`nouse`).

For details about the above operands, see *HA Monitor environment settings (sysdef)* in the manual *HA Monitor for Linux(R) (x86)*.

(3) Specifying the servers file

Among the operands that you need to specify for HA Monitor's `servers` file, the values to specify are predetermined for the following operands:

- `name`

  Specify the absolute path for the server directory.

  Note that you cannot specify a character string that includes spaces for the `name` operand as a path name. Therefore, if the multi-node function is being used, we recommend that you use as the server directory a directory whose path name does not include spaces.

- `acttype`

  Specify `monitor` because the multi-node function runs in HA Monitor's monitor mode.

- `termcommand`

  Specify the absolute path for the server termination command that describes the settings for HADB.

  Create the server termination command based on the appropriate template. For details about how to create a server termination command, see (d) Creating a server termination command in (4) Creating environment variable definitions for commands, and creating commands, and *Creating a server termination command* in the manual *HA Monitor for Linux(R) (x86)*.

- `initial`

  Specify `online` for the master node, and specify `standby` for a slave node.

- `disk`

  Specify the absolute path for the VG that includes the file system for the system directory and the file system that stores the synonym dictionary files. In the case of Figure 16-1: Example of a system configuration using the multi-node function, specify `/dev/vg_hadb` and `/dev/vg_hadb02`.

  If you do not intend to conduct synonym searches, you do not need to specify the absolute path of the VG that contains the file system where the synonym dictionary files are stored.

- `fs_name`

  Specify the absolute path for the logical volume that corresponds to the file system for the system directory and the file system that stores the synonym dictionary files. In the case of Figure 16-1: Example of a system configuration using the multi-node function, specify `/dev/vg_hadb/sys_hadb` and `/dev/vg_hadb02/hadb_syndict`.

  If you do not intend to conduct synonym searches, you do not need to specify the absolute path of the logical volume that corresponds to the file system where the synonym dictionary files are stored.

- `fs_mount_dir`

  Specify the path of the system directory (`$DBDIR/ADBSYS`) and the path of the directory where the file system that stores the synonym dictionary files is to be mounted.

  As the path of the directory where the file system that stores the synonym dictionary files is to be mounted, specify the same path on each node.

  If you do not intend to conduct synonym searches, you do not need to specify the path of the directory where the file system that stores the synonym dictionary files is to be mounted.

- `fs_mount_opt`

  Specify the mount option for mounting the system directory file system and the file system that stores the synonym dictionary files.

  If you do not intend to conduct synonym searches, you do not need to specify the mount option for mounting the file system that stores the synonym dictionary files.

- `lan_updown`

Specify `use` because the LAN status settings files are used.

Configure an alias IP address in the LAN status settings files. For details, see *Specifying LAN status settings files* in the manual *HA Monitor for Linux(R) (x86)*.

- `patrolcommand`

  Specify the absolute path for the server-monitoring command that describes the settings for HADB.

  Create the server-monitoring command based on the appropriate template. For details about how to create a server-monitoring command, see (e) Creating a server-monitoring command in (4) Creating environment variable definitions for commands, and creating commands, and *Creating a server monitoring command* in the manual *HA Monitor for Linux(R) (x86)*.

- `servexec_retry`

  Specify `0`.

- `waitserv_exec`

  Specify `yes`.

- `ip_neck`

  Specify `use`.

- `uoc_neck`

  Specify `nouse`.

- `vg_neck`

  Specify `use`.

  If you will be performing synonym searches, also specify `use` for the file system that stores synonym dictionary files.

- `fs_neck`

  Specify `use`.

  If you will be performing synonym searches, also specify `use` for the file system that stores synonym dictionary files.

- `actcommand`

  Specify the absolute path for the server startup command that describes the settings for HADB.

  Create the server startup command based on the appropriate template. For details about how to create a server startup command, see (c) Creating a server startup command in (4) Creating environment variable definitions for commands, and creating commands, and *Creating a server start command* in the manual *HA Monitor for Linux(R) (x86)*.

- `standbypri`

  Specify the node priority for the node to be made the master node in the case of a switchover. The smaller the specification value, the higher the priority. The higher the priority of the slave node, the smaller the value you must specify.

  Specify a different value for each slave node.

  Specify this operand in each slave node's HA Monitor (when `standby` is specified for the `initial` operand).

- `scsi_device`

  In this operand, specify the absolute paths of the device names that are to be subject to SCSI reservation.

  When using host reset, you do not need to specify this operand.

  Specify this operand when you are using SCSI reservation for shared disk and the shared disk meets any of the following conditions:

  - The shared disk is in a single-path configuration.

- The shared disk is in a VMware ESXi virtual environment (excluding situations where DMMP is used).
- The shared disk is in a redundant configuration realized using multipath software (HDLM).

When specifying this operand, specify the device names of the following file systems and disks by absolute path:

- File systems inherited at node switchover

  In Figure 16-1: Example of a system configuration using the multi-node function, `FS001` and `FS002` are file systems inherited at node switchover.
- Disks for DB area files (excluding disks for work table DB areas)

  In Figure 16-1: Example of a system configuration using the multi-node function, `LU001` to `LU005` are disks for DB area files.

If you do not intend to perform synonym searches, you do not need to specify the device name of the file system where the synonym dictionary files are stored.

Important

Do not specify the device names of the following file systems and disks in this operand:

- Node-local file system

  In Figure 16-1: Example of a system configuration using the multi-node function, `LOC001` to `LOC003` are node-local file systems.
- Disk where work table DB area is stored

  In Figure 16-1: Example of a system configuration using the multi-node function, `WRK001` to `WRK003` are disks where work table DB areas are stored.

When specifying this operand, see *Determining the operand value needed for SCSI reservation for shared disk* in the manual *HA Monitor for Linux(R) (x86)*.

- `dmmp_device`

In this operand, specify the absolute paths of the logical DMMP devices that are to be subject to SCSI reservation.

When using host reset, you do not need to specify this operand.

Specify this operand when you are using SCSI reservation for shared disk, and the shared disk is in a redundant configuration based on multipath software (DMMP).

When specifying this operand, specify the device names of the following file systems and disks by absolute path:

- File systems inherited at node switchover

  In Figure 16-1: Example of a system configuration using the multi-node function, `FS001` and `FS002` are file systems inherited at node switchover.
- DB area file disks that are accessible from all nodes

  In Figure 16-1: Example of a system configuration using the multi-node function, `LU001` to `LU005` are DB area file disks that are accessible from all nodes.

If you do not intend to perform synonym searches, you do not need to specify the device name of the file system where the synonym dictionary files are stored.

Important

Do not specify the device names of the following file systems and disks in this operand:

- Node-local file system

In Figure 16-1: Example of a system configuration using the multi-node function, LOC001 to LOC003 are node-local file systems.

- Disk where work table DB area is stored

  In Figure 16-1: Example of a system configuration using the multi-node function, WRK001 to WRK003 are disks where work table DB areas are stored.

When specifying this operand, see *Determining the operand value needed for SCSI reservation for shared disk* in the manual *HA Monitor for Linux(R) (x86)*.

For details about the preceding operands specified in the servers file, see *Server environment definition (servers)* in the manual *HA Monitor for Linux(R) (x86)*.

(4) Creating environment variable definitions for commands, and creating commands

To link to HA Monitor, create environment variable definitions for commands, and create the following commands:

- User command
- Server startup command
- Server termination command
- Server-monitoring command

Templates for environment variable definitions for commands, as well as the above commands, are located under $ADBDIR/sample/scripts. Create the commands based on the templates.

(a) Creating environment variable definitions for commands

$ADBDIR/sample/scripts/multinode.env is the template. Copy this template and correct the following locations:

- Environment variable ADBMGR

  Specify the user name of the HADB administrator (OS user).
- Environment variable ADBALIAS

  Specify the value specified for the alias operand in the servers file of HA Monitor.

(b) Creating a user command

$ADBDIR/sample/scripts/multinode_user.sh is the template. Copy this template and correct the following location:

- source statement

  Specify for the argument of the source statement the absolute path for the file that specifies the environment variable definitions for commands.

(c) Creating a server startup command

$ADBDIR/sample/scripts/multinode_act.sh is the template. Copy this template and correct the following location:

- source statement

Specify for the argument of the `source` statement the absolute path for the file that specifies the environment variable definitions for commands.

(d) Creating a server termination command

The template for the server termination command differs according to whether you are using host reset or SCSI reservation for shared disk. When using host reset, the template is `$ADBDIR/sample/scripts/multinode_term.sh`. When using SCSI reservation for shared disk, the template is `$ADBDIR/sample/scripts/multinode_term_scsi.sh`.

Copy the applicable template and amend the following part of the file:

- `source` statement

  Specify for the argument of the `source` statement the absolute path for the file that specifies the environment variable definitions for commands.

(e) Creating a server-monitoring command

`$ADBDIR/sample/scripts/multinode_patrol.sh` is the template. Copy this template and correct the following location:

- `source` statement

  Specify for the argument of the `source` statement the absolute path for the file that specifies the environment variable definitions for commands.

(5) File specification examples (when using host reset)

This subsection provides specification examples for the following files. The following are specification examples when you are using host reset.

- `sysdef` file
- `servers` file
- File for environment variable definitions for commands (`/HADB/scripts/multinode.env` in this example)
- User command file (`/HADB/scripts/multinode_user.sh` in this example)
- Server startup command file (`/HADB/scripts/multinode_act.sh` in this example)
- Server termination command file (`/HADB/scripts/multinode_term.sh` in this example)
- Server-monitoring command file (`/HADB/scripts/multinode_patrol.sh` in this example)

The specification examples explained here assume the system configuration shown in Figure 16-1: Example of a system configuration using the multi-node function.

(a) sysdef file specification examples

■ Specification example of the `sysdef` file for master node `hadb01`

```
environment  name hadb01,
             address 1,           ...1
             patrol 30,           ...2
             lan path11,          ...3
             lanport HAmon1;      ...4
function     pathpatrol 240,
             connect_retry 5:200,
```

```
            monbegin_restart nouse,
            multistandby use,
            usrcommand /HADB/scripts/multinode_user.sh,  ...5
            termcmd_at_abort nouse;
```

Explanation

1. Specifies the order of priority for resetting this node. Because this node is the master node, its reset priority needs to be the highest, so `1` is specified.
2. Specify a value of 30 or less for the `patrol` operand.
3. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path11`, which corresponds to IP address `172.16.0.11`.
4. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
5. Specify the absolute path for the user command to be used by the multi-node function.

■ **Specification example of the `sysdef` file for slave node `hadb02`**

```
environment  name hadb02,
             address 2,          ...1
             patrol 30,          ...2
             lan path21,         ...3
             lanport HAmon1;     ...4
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             multistandby use,
             usrcommand /HADB/scripts/multinode_user.sh,  ...5
             termcmd_at_abort nouse;
```

Explanation

1. Specifies the order of priority for resetting this node. Because this node needs to be the next highest priority after the master node, `2` is specified.
2. Specify a value of 30 or less for the `patrol` operand.
3. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path21`, which corresponds to IP address `172.16.0.12`.
4. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
5. Specify the absolute path for the user command to be used by the multi-node function.

■ **Specification example of the `sysdef` file for slave node `hadb03`**

```
environment  name  hadb03,
             address 3,          ...1
             patrol 30,          ...2
             lan path31,         ...3
             lanport HAmon1;     ...4
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             multistandby use,
             usrcommand /HADB/scripts/multinode_user.sh,  ...5
             term_at_abort  nouse;
```

Explanation

1. Specifies the order of priority for resetting this node. Because this node needs to be the next highest priority after the slave node `hadb02`, `3` is specified.
2. Specify a value of 30 or less for the `patrol` operand.
3. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path31`, which corresponds to IP address `172.16.0.13`.
4. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
5. Specify the absolute path for the user command to be used by the multi-node function.

(b) servers file specification examples

The examples described here assume that you are using ext4 as the file system. If you are using a file system other than ext4, the OS `mount` command options will differ from those in the examples shown here.

■ Specification example of the `servers` file for master node `hadb01`

```
server name /HADB/server,                        ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb:                        ...2
            /dev/vg_hadb02,
       lan_updown use,
       fs_name /dev/vg_hadb/sys_hadb:            ...3
               /dev/vg_hadb02/hadb_syndict,
       fs_mount_dir /HADB/db/ADBSYS:             ...4
                    /mnt/syndict,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":  ...5
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/multinode_act.sh",         ...6
       termcommand "/HADB/scripts/multinode_term.sh",       ...7
       patrolcommand "/HADB/scripts/multinode_patrol.sh",   ...8
       servexec_retry 0,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:use,    ...9
       fs_neck use:use,    ...10
       initial online;     ...11
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.

9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify `online` to make this node the master node.

■ **Specification example of the `servers` file for slave node `hadb02`**

```
server name /HADB/server,                       ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb:                       ...2
            /dev/vg_hadb02,
       lan_updown use,
       fs_name /dev/vg_hadb/sys_hadb:           ...3
               /dev/vg_hadb02/hadb_syndict,
       fs_mount_dir /HADB/db/ADBSYS:            ...4
                    /mnt/syndict,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":  ...5
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/multinode_act.sh",         ...6
       termcommand "/HADB/scripts/multinode_term.sh",       ...7
       patrolcommand "/HADB/scripts/multinode_patrol.sh",   ...8
       servexec_retry 0,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:use,    ...9
       fs_neck use:use,    ...10
       initial standby,    ...11
       standbypri 1;       ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify `standby` to make this node a slave node.
12. Specify the node priority for the node to be made the master node in the case of a switchover. Because this node needs to be the highest priority, `1` is specified.

■ **Specification example of the `servers` file for slave node `hadb03`**

```
server name /HADB/server,                       ...1
       alias HADB,
```

```
        acttype monitor,
        disk /dev/vg_hadb:                      ...2
             /dev/vg_hadb02,
        lan_updown use,
        fs_name /dev/vg_hadb/sys_hadb:          ...3
                /dev/vg_hadb02/hadb_syndict,
        fs_mount_dir /HADB/db/ADBSYS:           ...4
                     /mnt/syndict,
        fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":  ...5
                     "-t ext4 -o defaults,noatime,_netdev",
        actcommand "/HADB/scripts/multinode_act.sh",         ...6
        termcommand "/HADB/scripts/multinode_term.sh",       ...7
        patrolcommand "/HADB/scripts/multinode_patrol.sh",   ...8
        servexec_retry 0,
        waitserv_exec yes,
        ip_neck use,
        uoc_neck nouse,
        vg_neck use:use,    ...9
        fs_neck use:use,    ...10
        initial standby,    ...11
        standbypri 2;       ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify `standby` to make this node a slave node.
12. Specify the node priority for the node to be made the master node in the case of a switchover. Because this node needs to be the next highest priority after the node `hadb02`, `2` is specified.

(c) Specification example of a file for environment variable definitions for commands

In this example, environment variable definitions for commands are stored in `/HADB/scripts/multinode.env`. Use the same specification content for all nodes.

■ Specification example of the file for environment variable definitions for commands

```
# The environment variables for HADB
export ADBMGR=adbmanager              ...1
export ADBALIAS=HADB                  ...2
export ADBDIR=/HADB/server            ...3
```

Explanation

1. Specify user name `adbmanager` for the HADB administrator (OS user).
2. Specify `HADB`, which is the value specified for the `alias` operand in the `servers` file of HA Monitor.
3. Specify the absolute path for the server directory.

(d) Specification example of the user command file

In this example, the user command is stored in `/HADB/scripts/multinode_user.sh`. Use the same specification content for all nodes.

■ Specification example of the user command file

```
#!/bin/sh

# Sample of the usrcommand for HADB

# Setting environment variables for HADB
source /HADB/scripts/multinode.env       ...1

###############################################################################
#  Main
###############################################################################

KIND_ONLINE="online"
KIND_STANDBY="standby"
SERV_START="-s"
SERV_END="-e"
SERV_PLANEND="-p"
SERV_ABORT="-a"
SERV_ABORT_NS="-o"
SERV_FAULT="-f"
SERV_HOSTDOWN="-h"
SERV_PLANSWAP="-w"
STATUS_START="start"
STATUS_END="end"
STATUS_SBYEND="sbyend"


patrol_sby_exe()
{
  $ADBDIR/bin/patrol_sby_exe       ...2
}

patrol_sby_term()
{
  $ADBDIR/bin/patrol_sby_term      ...3
}

stop_sby()
{
  $ADBDIR/bin/stop_sby             ...4
}

# Processing of the server
if [ "$2" = "$ADBALIAS" ]
then
  if [ "$4" = "$KIND_ONLINE" ]
  then
    case "$5" in
    "$SERV_START" )
      ;;
```

```
    "$SERV_END" )
      ;;
    "$SERV_PLANEND" )
      ;;
    "$SERV_ABORT" )
      ;;
    "$SERV_ABORT_NS" )
      ;;
    "$SERV_PLANSWAP" )
      ;;
    esac
  else
    case "$5" in
    "$SERV_START" )
      [ "$6" = "$STATUS_START" ] && patrol_sby_exe &
      ;;
    "$SERV_END" )
      [ "$6" = "$STATUS_START" -o "$6" = "$STATUS_SBYEND" ] && stop_sby
      ;;
    "$SERV_PLANEND" )
      [ "$6" = "$STATUS_START" ] && patrol_sby_term
      ;;
    "$SERV_ABORT" )
      [ "$6" = "$STATUS_START" ] && patrol_sby_term
      [ "$6" = "$STATUS_SBYEND" ] && stop_sby
      ;;
    "$SERV_FAULT" )
      [ "$6" = "$STATUS_START" ] && stop_sby
      ;;
    "$SERV_HOSTDOWN" )
      [ "$6" = "$STATUS_START" ] && patrol_sby_term
      ;;
    "$SERV_PLANSWAP" )
      [ "$6" = "$STATUS_START" ] && patrol_sby_term
      ;;
    esac
  fi
fi

exit 0
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/multinode.env`).
2. Specify the command for starting the monitoring of the slave node.
3. Specify the command for ending the monitoring of the slave node.
4. Specify the command for terminating the slave node.

(e) Specification example of the server startup command file

In this example, the server startup command is stored in `/HADB/scripts/multinode_act.sh`. Use the same specification content for all nodes.

■ Specification example of the server startup command file

```
#!/bin/sh

# Sample of the actcommand for HADB

SU=/bin/su
```

```
# Setting environment variables for HADB
source /HADB/scripts/multinode.env      ...1

# Execute adbchgnodetype command
$SU - $ADBMGR -c "$ADBDIR/bin/adbchgnodetype --master -n $ADBALIAS"
CHGNODERES=$?
if [ "$CHGNODERES" = "0" -o "$CHGNODERES" = "4" ]
then
  exit 0
else
  exit $CHGNODERES
fi
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/multinode.env`).

(f) Specification example of the server termination command file

In this example, the server termination command is stored in `/HADB/scripts/multinode_term.sh`. Use the same specification content for all nodes.

■ Specification example of the server termination command file

```
#!/bin/sh

# Sample of the termcommand for HADB

STS_ACTIVE="ACTIVE"
STS_STOP="STOP"
STS_STARTING="STARTING"
STS_STOPPING="STOPPING"
STS_ABORT="ABORT"
STS_QUIESCE="QUIESCE"
STS_OFFLINE="OFFLINE"
STS_CHGMODE="CHGMODE"
STS_STOPWAIT="STOPWAIT"
STS_COREDUMP="COREDUMP"
STS_FORCE="FORCE"
STS_MAINTNCE="MAINTNCE"

AWK=/bin/awk
ECHO=/bin/echo
GREP=/bin/grep
SU=/bin/su

# Setting environment variables for HADB
source /HADB/scripts/multinode.env      ...1

# Execute adbstop command
while :
do

  GET_STS=`$SU - $ADBMGR -c "$ADBDIR/bin/adbls -d srv 2>/dev/null" | $GREP -v SVID
 | $AWK '{ if ($1 ~ /[0-9]+/) { print $2 } else { print $1 } }'`

  case "$1" in
  "-e" )
    # Normal stop operations of master node.
    # (when the monend command is executed.)
    if [ "$GET_STS" = "$STS_ACTIVE"   -o \
```

```
          "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
          "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
          "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_MAINTNCE" ]
    then
      $SU - $ADBMGR -c "$ADBDIR/bin/adbstop --cancel"
      STOPRES=$?
      if [ "$STOPRES" = "0" -o "$STOPRES" = "4" ]
      then
        break
      fi
    elif [ "$GET_STS" = "$STS_STARTING" ]
    then
      break
    else
      break
    fi

    ;;
  "-w" )
    # Plan stop operations of master node.
    # (when the monswap command is executed.)
    if [ "$GET_STS" = "$STS_ACTIVE"   -o "$GET_STS" = "$STS_STARTING" -o \
         "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
         "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
         "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_MAINTNCE" ]
    then
      $SU - $ADBMGR -c "$ADBDIR/bin/adbstop --cancel --node"
      STOPRES=$?
      if [ "$STOPRES" = "0" -o "$STOPRES" = "4" ]
      then
        break
      fi
    else
      break
    fi

    ;;
  "-c" )
    break

    ;;
  esac

done

# stop HADB
if [ "$GET_STS" = "$STS_STARTING" ]
then
  ADB_ID=`$SU - $ADBMGR -c "ps x" | $GREP adbsrvd | $GREP -v $GREP | awk '{print $
1}'`
  $SU - $ADBMGR -c "kill $ADB_ID"

  exit 0
fi

# Wait for end of HADB
while [ "$GET_STS" = "$STS_ACTIVE"   -o "$GET_STS" = "$STS_STARTING" -o \
        "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
        "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
        "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_COREDUMP" -o \
        "$GET_STS" = "$STS_MAINTNCE" ]
do
  GET_STS=`$SU - $ADBMGR -c "$ADBDIR/bin/adbls -d srv 2>/dev/null" | $GREP -v SVID
 | $AWK '{ if ($1 ~ /[0-9]+/) { print $2 } else { print $1 } }'`
```

```
  sleep 1
done

exit 0
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/multinode.env`).

(g) Specification example of the server-monitoring command file

In this example, the server-monitoring command is stored in `/HADB/scripts/multinode_patrol.sh`. Use the same specification content for all nodes.

■ Specification example of the server-monitoring command file

```
#!/bin/sh

# Sample of the patrolcommand for HADB

AWK=/bin/awk
GREP=/bin/grep
PS=/bin/ps
PGREP=/usr/bin/pgrep
PKILL=/usr/bin/pkill
SU=/bin/su

# Setting environment variables for HADB
source /HADB/scripts/multinode.env      ...1

# Execute adbmonitor command for master node
$SU - $ADBMGR -c "$ADBDIR/bin/adbmonitor -n" &

## Get su command process id
PID=$!

# Trap SIGTERM and terminate adbmonitor
trap "$PKILL -P $PID" 15


## Wait process terminated
CPID=""
CHKPID=`$PS aux | $AWK -v PID=$PID '{ if ($2 == PID) { print $2 } }'`
while [ "$CHKPID" != "" ]
do

  ## Get adbmonitor process id
  if [ "$CPID" = "" ]
  then
    CPID=`$PGREP -P $PID`
  fi

  ## Wait
  if [ "$CPID" != "" ]
  then
    CHKCPID=`$PS aux | $AWK -v CPID=$CPID '{ if ($2 == CPID) { print $2 } }'`
    while [ "$CHKCPID" != "" ]
    do
      sleep 1
      CHKCPID=`$PS aux | $AWK -v CPID=$CPID '{ if ($2 == CPID) { print $2 } }'`
    done
  fi
```

```
  sleep 1
  CHKPID=`$PS aux | $AWK -v PID=$PID '{ if ($2 == PID) { print $2 } }'`
done

exit 0
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/multinode.env`).

(6) File specification examples (when using SCSI reservation for shared disk)

This subsection provides specification examples for the following files. The following are specification examples when you are using SCSI reservation for shared disk.

- `sysdef` file
- `servers` file
- File for environment variable definitions for commands (`/HADB/scripts/multinode.env` in this example)
- User command file (`/HADB/scripts/multinode_user.sh` in this example)
- Server startup command file (`/HADB/scripts/multinode_act.sh` in this example)
- Server termination command file (`/HADB/scripts/multinode_term_scsi.sh` in this example)
- Server-monitoring command file (`/HADB/scripts/multinode_patrol.sh` in this example)

The specification examples explained here assume the system configuration shown in Figure 16-1: Example of a system configuration using the multi-node function.

(a) sysdef file specification examples

■ Specification example of the `sysdef` file for master node `hadb01`

```
environment  name hadb01,
             address 1,         ...1
             patrol 30,         ...2
             lan path11,        ...3
             lanport HAmon1;    ...4
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             multistandby use,
             usrcommand /HADB/scripts/multinode_user.sh,              ...5
             termcmd_at_abort nouse,
             fence_reset nouse,                                       ...6
             fence_scsi use,                                          ...7
             fence_lan nouse;
```

Explanation

1. Specifies the order of priority for resetting this node. Because this node is the master node, its reset priority needs to be the highest, so `1` is specified.
2. Specify a value of 30 or less for the `patrol` operand.

3. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path11`, which corresponds to IP address `172.16.0.11`.
4. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
5. Specify the absolute path for the user command to be used by the multi-node function.
6. Because host reset will not be used, specify `nouse`.
7. Because SCSI reservation for shared disk will be used, specify `use`.

■ **Specification example of the `sysdef` file for slave node `hadb02`**

```
environment  name hadb02,
             address 2,         ...1
             patrol 30,         ...2
             lan path21,        ...3
             lanport HAmon1;    ...4
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             multistandby use,
             usrcommand /HADB/scripts/multinode_user.sh,              ...5
             termcmd_at_abort nouse,
             fence_reset nouse,                                       ...6
             fence_scsi use,                                          ...7
             fence_lan nouse;
```

Explanation

1. Specifies the order of priority for resetting this node. Because this node needs to be the next highest priority after the master node, `2` is specified.
2. Specify a value of 30 or less for the `patrol` operand.
3. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path21`, which corresponds to IP address `172.16.0.12`.
4. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
5. Specify the absolute path for the user command to be used by the multi-node function.
6. Because host reset will not be used, specify `nouse`.
7. Because SCSI reservation for shared disk will be used, specify `use`.

■ **Specification example of the `sysdef` file for slave node `hadb03`**

```
environment  name hadb03,
             address 3,         ...1
             patrol 30,         ...2
             lan path31,        ...3
             lanport HAmon1;    ...4
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             multistandby use,
             usrcommand /HADB/scripts/multinode_user.sh,              ...5
             termcmd_at_abort nouse,
             fence_reset nouse,                                       ...6
             fence_scsi use,                                          ...7
             fence_lan nouse;
```

Explanation

1. Specifies the order of priority for resetting this node. Because this node needs to be the next highest priority after the slave node `hadb02`, `3` is specified.
2. Specify a value of 30 or less for the `patrol` operand.
3. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path31`, which corresponds to IP address `172.16.0.13`.
4. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
5. Specify the absolute path for the user command to be used by the multi-node function.
6. Because host reset will not be used, specify `nouse`.
7. Because SCSI reservation for shared disk will be used, specify `use`.

(b) Specification example of a servers file (when using a single-path configuration)

The following describes examples of specifying a `servers` file in a single-path configuration.

The examples described here assume that you are using ext4 as the file system. If you are using a file system other than ext4, the OS `mount` command options will differ from those in the examples shown here.

■ Specification example of the `servers` file for master node `hadb01`

```
server name /HADB/server,                          ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb:                          ...2
            /dev/vg_hadb02,
       lan_updown use,
       fs_name /dev/vg_hadb/sys_hadb:              ...3
               /dev/vg_hadb02/hadb_syndict,
       fs_mount_dir /HADB/db/ADBSYS:               ...4
                    /mnt/syndict,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":       ...5
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/multinode_act.sh",              ...6
       termcommand "/HADB/scripts/multinode_term_scsi.sh",       ...7
       patrolcommand "/HADB/scripts/multinode_patrol.sh",        ...8
       servexec_retry 0,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:use,                                                    ...9
       fs_neck use:use,                                                    ...10
       scsi_device /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b53e0a:  ...11
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b6c44d:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b828e9:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b7d9fd:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b87793:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b8c6d3:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b9160f,
       initial online;                                                     ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory

- The directory storing synonym dictionary files (a directory created in order to perform synonym searches)

3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify the absolute paths of the disks that store the following files and file systems:
    - File system where the system directory is created
    - File system where the directory that stores synonym dictionary files is created
    - DB area files (excluding work table DB area files)

    Specify the absolute paths in the same order on all nodes.
12. Specify `online` to make this node the master node.

■ **Specification example of the `servers` file for slave node `hadb02`**

```
server name /HADB/server,                              ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb:                              ...2
            /dev/vg_hadb02,
       lan_updown use,
       fs_name /dev/vg_hadb/sys_hadb:                  ...3
               /dev/vg_hadb02/hadb_syndict,
       fs_mount_dir /HADB/db/ADBSYS:                   ...4
                    /mnt/syndict,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":       ...5
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/multinode_act.sh",              ...6
       termcommand "/HADB/scripts/multinode_term_scsi.sh",       ...7
       patrolcommand "/HADB/scripts/multinode_patrol.sh",        ...8
       servexec_retry 0,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:use,                                                    ...9
       fs_neck use:use,                                                    ...10
       scsi_device /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b53e0a:  ...11
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b6c44d:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b828e9:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b7d9fd:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b87793:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b8c6d3:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b9160f,
       initial standby,                                                    ...12
       standbypri 1;                                                       ...13
```

Explanation

1. Specify the absolute path for the server directory.

2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify the absolute paths of the disks that store the following files and file systems:
    - File system where the system directory is created
    - File system where the directory that stores synonym dictionary files is created
    - DB area files (excluding work table DB area files)

    Specify the absolute paths in the same order on all nodes.
12. Specify `standby` to make this node a slave node.
13. Specify the node priority for the node to be made the master node in the case of a switchover. Because this node needs to be the highest priority, `1` is specified.

■ **Specification example of the `servers` file for slave node `hadb03`**

```
server name /HADB/server,                                ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb:                                ...2
            /dev/vg_hadb02,
       lan_updown use,
       fs_name /dev/vg_hadb/sys_hadb:                    ...3
               /dev/vg_hadb02/hadb_syndict,
       fs_mount_dir /HADB/db/ADBSYS:                     ...4
                    /mnt/syndict,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":       ...5
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/multinode_act.sh",              ...6
       termcommand "/HADB/scripts/multinode_term_scsi.sh",       ...7
       patrolcommand "/HADB/scripts/multinode_patrol.sh",        ...8
       servexec_retry 0,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:use,                                                    ...9
       fs_neck use:use,                                                    ...10
       scsi_device /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b53e0a:  ...11
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b6c44d:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b828e9:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b7d9fd:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b87793:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b8c6d3:
                   /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b9160f,
```

```
       initial standby,                                                    ...12
       standbypri 2;                                                       ...13
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify the absolute paths of the disks that store the following files and file systems:
    - File system where the system directory is created
    - File system where the directory that stores synonym dictionary files is created
    - DB area files (excluding work table DB area files)

    Specify the absolute paths in the same order on all nodes.
12. Specify `standby` to make this node a slave node.
13. Specify the node priority for the node to be made the master node in the case of a switchover. Because this node needs to be the next highest priority after the node `hadb02`, `2` is specified.

(c) Specification example of a servers file (when using a redundant configuration realized by multipath software)

The following describes examples of specifying a `servers` file in a redundant configuration realized by multipath software.

The examples described here assume that you are using ext4 as the file system. If you are using a file system other than ext4, the OS `mount` command options will differ from those in the examples shown here.

■ Specification example of the `servers` file for master node `hadb01`

```
server name /HADB/server,                          ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb:                          ...2
            /dev/vg_hadb02,
       lan_updown use,
       fs_name /dev/vg_hadb/sys_hadb:              ...3
               /dev/vg_hadb02/hadb_syndict,
       fs_mount_dir /HADB/db/ADBSYS:               ...4
                    /mnt/syndict,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":   ...5
```

```
                "-t ext4 -o defaults,noatime,_netdev",
      actcommand "/HADB/scripts/multinode_act.sh",          ...6
      termcommand "/HADB/scripts/multinode_term_scsi.sh",   ...7
      patrolcommand "/HADB/scripts/multinode_patrol.sh",    ...8
      servexec_retry 0,
      waitserv_exec yes,
      ip_neck use,
      uoc_neck nouse,
      vg_neck use:use,                    ...9
      fs_neck use:use,                    ...10
      dmmp_device /dev/mapper/mpath1:     ...11
                  /dev/mapper/mpath2:
                  /dev/mapper/mpath11:
                  /dev/mapper/mpath12:
                  /dev/mapper/mpath13:
                  /dev/mapper/mpath14:
                  /dev/mapper/mpath15,
      initial online;                     ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify the absolute paths of the disks that store the following files and file systems:
    - File system where the system directory is created
    - File system where the directory that stores synonym dictionary files is created
    - DB area files (excluding work table DB area files)

    Specify the absolute paths in the same order on all nodes.
12. Specify `online` to make this node the master node.

■ **Specification example of the `servers` file for slave node `hadb02`**

```
server name /HADB/server,                         ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb:                         ...2
            /dev/vg_hadb02,
       lan_updown use,
       fs_name /dev/vg_hadb/sys_hadb:             ...3
               /dev/vg_hadb02/hadb_syndict,
       fs_mount_dir /HADB/db/ADBSYS:              ...4
```

```
                /mnt/syndict,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":   ...5
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/multinode_act.sh",          ...6
       termcommand "/HADB/scripts/multinode_term_scsi.sh",   ...7
       patrolcommand "/HADB/scripts/multinode_patrol.sh",    ...8
       servexec_retry 0,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:use,                    ...9
       fs_neck use:use,                    ...10
       dmmp_device /dev/mapper/mpath1:     ...11
                   /dev/mapper/mpath2:
                   /dev/mapper/mpath11:
                   /dev/mapper/mpath12:
                   /dev/mapper/mpath13:
                   /dev/mapper/mpath14:
                   /dev/mapper/mpath15,
       initial standby,                    ...12
       standbypri 1;                       ...13
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify the absolute paths of the disks that store the following files and file systems:
    - File system where the system directory is created
    - File system where the directory that stores synonym dictionary files is created
    - DB area files (excluding work table DB area files)

    Specify the absolute paths in the same order on all nodes.
12. Specify `standby` to make this node a slave node.
13. Specify the node priority for the node to be made the master node in the case of a switchover. Because this node needs to be the highest priority, `1` is specified.

■ **Specification example of the `servers` file for slave node `hadb03`**

```
server name /HADB/server,                          ...1
       alias HADB,
       acttype monitor,
```

```
disk /dev/vg_hadb:                          ...2
     /dev/vg_hadb02,
lan_updown use,
fs_name /dev/vg_hadb/sys_hadb:              ...3
        /dev/vg_hadb02/hadb_syndict,
fs_mount_dir /HADB/db/ADBSYS:               ...4
             /mnt/syndict,
fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":   ...5
             "-t ext4 -o defaults,noatime,_netdev",
actcommand "/HADB/scripts/multinode_act.sh",          ...6
termcommand "/HADB/scripts/multinode_term_scsi.sh",   ...7
patrolcommand "/HADB/scripts/multinode_patrol.sh",    ...8
servexec_retry 0,
waitserv_exec yes,
ip_neck use,
uoc_neck nouse,
vg_neck use:use,                    ...9
fs_neck use:use,                    ...10
dmmp_device /dev/mapper/mpath1:     ...11
            /dev/mapper/mpath2:
            /dev/mapper/mpath11:
            /dev/mapper/mpath12:
            /dev/mapper/mpath13:
            /dev/mapper/mpath14:
            /dev/mapper/mpath15,
initial standby,                    ...12
standbypri 2;                       ...13
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - System directory
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used by the multi-node function.
7. Specify the absolute path for the server termination command to be used by the multi-node function.
8. Specify the absolute path for the server-monitoring command to be used by the multi-node function.
9. For the VGs that contain the file systems of the directories in 2, specify `use`.
10. For the file systems of the directories in 2, specify `use`.
11. Specify the absolute paths of the disks that store the following files and file systems:
    - File system where the system directory is created
    - File system where the directory that stores synonym dictionary files is created
    - DB area files (excluding work table DB area files)

    Specify the absolute paths in the same order on all nodes.
12. Specify `standby` to make this node a slave node.
13. Specify the node priority for the node to be made the master node in the case of a switchover. Because this node needs to be the next highest priority after the node `hadb02`, `2` is specified.

(d) Specification example of a file for environment variable definitions for commands

For a specification example of a file for environment variable definitions for commands, see (c) Specification example of a file for environment variable definitions for commands in (5) File specification examples (when using host reset).

(e) Specification example of the user command file

For an example of specifying a user command file, see (d) Specification example of the user command file in (5) File specification examples (when using host reset).

(f) Specification example of the server startup command file

For a specification example of the server startup command file, see (e) Specification example of the server startup command file in (5) File specification examples (when using host reset).

(g) Specification example of the server termination command file

In this example, the server termination command is stored in `/HADB/scripts/multinode_term_scsi.sh`. Use the same specification content for all nodes.

■ Specification example of the server termination command file

```
#!/bin/sh

# Sample of the termcommand for HADB

STS_ACTIVE="ACTIVE"
STS_STOP="STOP"
STS_STARTING="STARTING"
STS_STOPPING="STOPPING"
STS_ABORT="ABORT"
STS_QUIESCE="QUIESCE"
STS_OFFLINE="OFFLINE"
STS_CHGMODE="CHGMODE"
STS_STOPWAIT="STOPWAIT"
STS_COREDUMP="COREDUMP"
STS_FORCE="FORCE"
STS_MAINTNCE="MAINTNCE"

AWK=/bin/awk
ECHO=/bin/echo
GREP=/bin/grep
SU=/bin/su

# Setting environment variables for HADB
source /HADB/scripts/multinode.env      ...1

# Execute adbstop command
while :
do

  GET_STS=`$SU - $ADBMGR -c "$ADBDIR/bin/adbls -d srv 2>/dev/null" | $GREP -v SVID
 | $AWK '{ if ($1 ~ /[0-9]+/) { print $2 } else { print $1 } }'`

  case "$1" in
  "-e" )
    # Normal stop operations of master node.
    # (when the monend command is executed.)
    if [ "$GET_STS" = "$STS_ACTIVE"   -o \
         "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
         "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
```

```
          "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_MAINTNCE" ]
    then
      $SU - $ADBMGR -c "$ADBDIR/bin/adbstop --cancel"
      STOPRES=$?
      if [ "$STOPRES" = "0" -o "$STOPRES" = "4" ]
      then
        break
      fi
    elif [ "$GET_STS" = "$STS_STARTING" ]
    then
      break
    else
      break
    fi

    ;;
  "-w" )
    # Plan stop operations of master node.
    # (When the monswap command is executed or a monitoring path failure occurs.)
    if [ "$GET_STS" = "$STS_ACTIVE"   -o "$GET_STS" = "$STS_STARTING" -o \
         "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
         "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
         "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_MAINTNCE" ]
    then
      $SU - $ADBMGR -c "$ECHO y|$ADBDIR/bin/adbstop --force"
      STOPRES=$?
      if [ "$STOPRES" = "0" -o "$STOPRES" = "4" ]
      then
        break
      fi
    else
      break
    fi

    ;;
  "-c" )
    break

    ;;
  esac

done

# stop HADB
if [ "$GET_STS" = "$STS_STARTING" ]
then
  ADB_ID=`$SU - $ADBMGR -c "ps x" | $GREP adbsrvd | $GREP -v $GREP | awk '{print $
1}'`
  $SU - $ADBMGR -c "kill $ADB_ID"

  exit 0
fi

# Wait for end of HADB
while [ "$GET_STS" = "$STS_ACTIVE"   -o "$GET_STS" = "$STS_STARTING" -o \
        "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
        "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
        "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_COREDUMP" -o \
        "$GET_STS" = "$STS_MAINTNCE" ]
do
  GET_STS=`$SU - $ADBMGR -c "$ADBDIR/bin/adbls -d srv 2>/dev/null" | $GREP -v SVID
 | $AWK '{ if ($1 ~ /[0-9]+/) { print $2 } else { print $1 } }'`
  sleep 1
done
```

```
exit 0
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/multinode.env`).

(h) Specification example of the server-monitoring command file

For a specification example of the server-monitoring command file, see (g) Specification example of the server-monitoring command file in (5) File specification examples (when using host reset).

(7) HA Monitor startup setting

Set up HA Monitor to start automatically when the operating system starts. For details about how to configure this setting, see *Automating the operation from system start through server start* in the manual *HA Monitor for Linux(R) (x86)*.

16.3.5 Creating server definitions on all nodes

When you use the multi-node function, you need to create a server definition on each node. When doing so, you must specify the `adb_sys_multi_node_info` operand.

For details about how to create a server definition, see 8.5 Creating and modifying a server definition. For details about the operands in the server definition, see 7. Designing the Server Definition.

You must specify the same value on each node for the following operands. If you omit an operand, omit it from all nodes. If the specification values are different, HADB servers in the multi-node configuration will not be able to start.

- `adb_dbarea_wrk_page_size`
- `adb_sys_multi_node_info` (The values also need to be specified in the same order.)
- `adb_sys_max_users`
- `adb_sys_max_users_wrn_pnt`
- `adb_sys_trn_iso_lv`
- `adb_sql_order_mode`
- `adb_sql_prep_delrsvd_words` (The values also need to be specified in the same order.)
- `adb_rpc_port`
- `adb_sql_prep_dec_div_rs_prior`
- `adb_sql_default_dbarea_shared`
- `adb_syndict_storage_path`
- `adbcltgrp` (`-g`, `-m`, `-u`, and `-w` options)

You must specify the following operands on all nodes when using the audit trail facility. The values specified for the operands do not need to be the same on all nodes. If any of these operands are specified on some nodes but not others, HADB servers in the multi-node configuration will not be able to start.

- `adb_audit_log_path`

■ **Server definition specification example**

A server definition specification example is provided below. In this example, the specification content is identical on all nodes. However, except for the operands that need to have the same specification values, you can change the specification values for each node.

```
set adb_db_path = /HADB/db
set adb_rpc_port = 23650
set adb_sys_max_users = 10
set adb_sys_rthd_num = 40
set adb_sys_uthd_num = 128
set adb_sql_exe_max_rthd_num = 4
set adb_sys_rthd_area_max = 1024
set adb_sys_proc_area_max = 8192
set adb_sys_multi_node_info =172.16.0.11:23651,\    ...1
                             172.16.0.12:23651,\    ...2
                             172.16.0.13:23651      ...3

adbbuff -g TBLBUF01 \
        -n ADBUTBL01 \
        -p 1000000 \
        -v 1024
adbbuff -g IDXBUF01 \
        -n ADBUIDX01 \
        -p 2500000
```

Explanation

1. Specify IP address `172.16.0.11` and port number `23651` for communication between HADB servers on master node `hadb01`.
2. Specify IP address `172.16.0.12` and port number `23651` for communication between HADB servers on slave node `hadb02`.
3. Specify IP address `172.16.0.13` and port number `23651` for communication between HADB servers on slave node `hadb03`.

16.3.6 Creating client definitions

Create separate client definitions for application programs that execute only retrieval SQL statements, and application programs that also execute update SQL statements, by specifying different transaction access modes and transaction isolation levels. For example, if you are using the JDBC driver, change the specification of the transaction access mode and transaction isolation level in properties such as the system properties.

For details about how to create a client definition, see *Creating a client definition* in *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*. For details about the operands in a client definition, see *Contents of operands in the client definition* under *Designing Client Definitions* in the *HADB Application Development Guide*.

■ **Client definition specification example (for an application that executes only retrieval SQL statements)**

```
set adb_clt_rpc_srv_host = 10.196.108.143     ...1
set adb_clt_rpc_srv_port = 23650              ...2
set adb_clt_ap_name = "AP01"                  ...3
set adb_clt_trn_access_mode = READ_ONLY       ...4
set adb_clt_trn_iso_lv = READ_COMMITTED       ...5
```

Explanation

1. For the connection-destination host name, specify alias IP address `10.196.108.143`, which is used for client-server communication.
2. For the connection-destination port number, specify `23650`, which is the value specified for the `adb_rpc_port` operand in the server definition.
3. Specify an application identifier.
4. Because this is an application that executes only retrieval SQL statements, specify `READ_ONLY` to set the transaction access mode to read-only.
5. Specify `READ_COMMITTED` as the transaction isolation level.

As a result of the specifications in 4 and 5, the HADB server on the master node selects a node with a light load, and retrieval SQL statements are executed on that node.

■ **Client definition specification example (for an application that also executes update SQL statements)**

```
set adb_clt_rpc_srv_host = 10.196.108.143     ...1
set adb_clt_rpc_srv_port = 23650              ...2
set adb_clt_ap_name = "AP02"                  ...3
set adb_clt_trn_access_mode = READ_WRITE      ...4
```

Explanation

1. For the connection-destination host name, specify alias IP address `10.196.108.143`, which is used for client-server communication.
2. For the connection-destination port number, specify `23650`, which is the value specified for the `adb_rpc_port` operand in the server definition.
3. Specify an application identifier.
4. Because this is an application that also executes update SQL statements, specify `READ_WRITE` to set the transaction access mode to read/write. As a result, the SQL statements to be executed by this application are executed only on the master node.

16.3.7 Environment settings when using the function for centrally managing client definitions

When using the function for centrally managing client definitions, create the client management definition file and the client definition file used with this function, on all nodes (create the same files on all nodes). If the file content is inconsistent among the nodes, the HADB server cannot be started in a multi-node configuration.

> **Note**
>
> If a master node switchover occurs, you need to create the same files on all nodes in order to continue using the function for centrally managing client definitions in the same state as before the switchover occurred.

In addition, if you changed the content of the client management definition file on the master node or the client definition file used with this function, please also change the content of the files on all slave nodes. If the file content is inconsistent among the nodes, executing the `adbclientdefmang` command with the `--update` option specified results in an error.

For details about the function for centrally managing client definitions, see 2.13 Centralized management of client definitions.

16.3.8 Creating a database

This subsection explains the procedure for creating a database when the multi-node function is used.

(1) General procedure for creating a database

The following shows the general procedure for creating a database:

1. Initialize the file systems to be inherited at node switchover.
2. Create a database on the master node.
3. Create a DB directory on each slave node.

(2) Initializing the file systems subject to node switchover

Use commands provided by the operating system to initialize the file system for the system directory and the file system that stores the synonym dictionary files.

Initializing the file system for the system directory

The following example initializes `/dev/vg_hadb/hadb_sys`, which has been created as an LV for the system directory, as an ext4 file system.

Command execution example

```
mkfs -t ext4 /dev/vg_hadb/hadb_sys
```

Initializing the file system that stores synonym dictionary files

The following execution example initializes `/dev/vg_hadb02/hadb_syndict`, which was created as the LV for storing synonym dictionary files, as an ext4 file system.

Command execution example

```
mkfs -t ext4 /dev/vg_hadb02/hadb_syndict
```

If you will not conduct synonym searches, you do not need to initialize the file system that stores synonym dictionary files.

(3) Creating a database on the master node

The following explains how to create a database on the master node.

First, execute the OS `mount` command on the master node, and on `$DBDIR/ADBSYS` mount the file system used for the system directory.

Next, execute the `adbinit` command on the master node, and create the database. The following shows an example of specifying the initialization option of the `adbinit` command.

■ Initialization option specification example (master node)

- Example for single-path configuration

```
set adb_init_dbarea_initialize = Y
set adb_init_wrk_blk_path = /dev/mapper/WRK001
set adb_init_mst_blk_path = /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b7d9fd
set adb_init_dic_blk_path = /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b828e9
set adb_init_stbl_blk_path=/dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b87793
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b8c6d3
```

```
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b9160f
```

- Example for redundant configuration using multipath software (DMMP)

```
set adb_init_dbarea_initialize = Y
set adb_init_wrk_blk_path = /dev/mapper/WRK001
set adb_init_mst_blk_path = /dev/mapper/mpath12
set adb_init_dic_blk_path = /dev/mapper/mpath11
set adb_init_stbl_blk_path = /dev/mapper/mpath13
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/mapper/mpath14
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/mapper/mpath15
```

Finally, after execution of the `adbinit` command ends, from `$DBDIR/ADBSYS` unmount the file system used for the system directory.

(4) Creating a DB directory on each slave node

Execute the `adbinit` command on each of the slave nodes. The initialization option to be specified here is different from that specified for the master node in the following ways:

- `N` is specified for `adb_init_dbarea_initialize` to create only a framework for a DB directory.
- Specify for `adb_init_wrk_blk_path` the block special file name of the work table DB area file to be allocated to each node.

■ Initialization option specification example (slave node `hadb02`)

- Example for single-path configuration

```
# Create only a framework for the DB directory.
set adb_init_dbarea_initialize = N
# WRK002 : DB area disk that constitutes work table DB area for hadb02
set adb_init_wrk_blk_path = /dev/mapper/WRK002
set adb_init_mst_blk_path = /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b7d9fd
set adb_init_dic_blk_path = /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b828e9
set adb_init_stbl_blk_path=/dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b87793
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b8c6d3
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b9160f
```

- Example for redundant configuration using multipath software (DMMP)

```
# Create only a framework for the DB directory.
set adb_init_dbarea_initialize = N
# WRK002 : DB area disk that constitutes work table DB area for hadb02
set adb_init_wrk_blk_path = /dev/mapper/WRK002
set adb_init_mst_blk_path = /dev/mapper/mpath12
set adb_init_dic_blk_path = /dev/mapper/mpath11
set adb_init_stbl_blk_path = /dev/mapper/mpath13
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/mapper/mpath14
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/mapper/mpath15
```

■ Initialization option specification example (slave node `hadb03`)

- Example for single-path configuration

```
# Create only a framework for the DB directory.
set adb_init_dbarea_initialize = N
# WRK003 : DB area disk that constitutes work table DB area for hadb03
set adb_init_wrk_blk_path = /dev/mapper/WRK003
set adb_init_mst_blk_path = /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b7d9fd
set adb_init_dic_blk_path = /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b828e9
set adb_init_stbl_blk_path=/dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b87793
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b8c6d3
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/disk/by-id/wwn-0x600605b0041db9c016ea34c3b1b9160f
```

- Example for redundant configuration using multipath software (DMMP)

```
# Create only a framework for the DB directory.
set adb_init_dbarea_initialize = N
# WRK003 : DB area disk that constitutes work table DB area for hadb03
set adb_init_wrk_blk_path = /dev/mapper/WRK003
set adb_init_mst_blk_path = /dev/mapper/mpath12
set adb_init_dic_blk_path = /dev/mapper/mpath11
set adb_init_stbl_blk_path = /dev/mapper/mpath13
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/mapper/mpath14
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/mapper/mpath15
```

16.4 Starting and terminating HADB servers in the multi-node configuration

This section explains how to start and terminate HADB servers in the multi-node configuration.

Also see 10.2 Starting and terminating the HADB server and its operation modes.

16.4.1 Starting the HADB servers in the multi-node configuration

(1) Startup procedure for the HADB servers in a multi-node configuration

The procedure for starting the HADB servers in the multi-node configuration is given below.

Important

After you have confirmed that monitoring by HA Monitor has stopped, start the HADB servers in the multi-node configuration. If you accidentally executed HA Monitor's `monbegin` command before executing the `adbstart` command, execute the `monend` command to stop monitoring by HA Monitor and the HADB servers that are running. Then, follow the correct procedure to start the HADB servers in the multi-node configuration.

Procedure:

1. Confirm that HA Monitor is active on all nodes.
   For the confirmation method, see *System Operation* in the manual *HA Monitor for Linux(R) (x86)*.
2. Execute the `adbstart` command on all nodes.
   When the `adbstart` command is executed, the message `KFAA91109-I` is output asking for execution of HA Monitor's `monbegin` command.
3. Execute HA Monitor's `monbegin` command on all nodes.
   After confirming that the message `KFAA91109-I` is output, execute the `monbegin` command from a different terminal (command input screen).

   Important

   Execute the `monbegin` command before the `adbstart` command terminates.

   If the `adbstart` command terminates normally on all nodes, the starting process for the HADB servers in the multi-node configuration is complete.

(2) Startup modes of HADB servers in the multi-node configuration

The following table shows the startup modes of HADB servers in the multi-node configuration.

Table 16-5: Startup modes of HADB servers in the multi-node configuration

No.	Startup mode	Command to be executed	Explanation	Last termination mode
1	Normal start	`adbstart`	This is the normal startup mode. In a normal start, database recovery processing does not take place.	When the master node HADB server terminated normally
2	Restart		If the previous termination mode is as described on the right, the server will automatically restart. During the restart, system logs of the HADB server on the master node are used to perform database recovery processing.	When only the master node remains, and that master node's HADB server terminated abnormally or was forced to terminate

(3) HADB server operation modes in the multi-node configuration

In the case of an HADB server on the master node, by specifying an option for the `adbstart` command, you can specify the HADB server operation mode to be used after the HADB server has started. In the case of an HADB server on a slave node, its HADB server operation mode is always the quiescence mode, and therefore you cannot change the HADB server operation mode by specifying an option for the `adbstart` command.

(4) Checking whether the startup process for HADB servers in the multi-node configuration is complete

Confirm that the return code of the `adbstart` command is `0` or `4` on all nodes.

(5) Notes on starting HADB servers in the multi-node configuration

Startup processing of the HADB servers in the multi-node configuration is not complete until startup processing of each HADB server on each node is complete. If startup processing of the HADB servers in the multi-node configuration is not complete, check whether an error or failure occurred on one of the nodes.

16.4.2 Terminating HADB servers in the multi-node configuration

(1) Termination procedures for HADB servers in a multi-node configuration

(a) Normally terminating HADB servers in the multi-node configuration

The following shows the procedure for normally terminating the HADB servers in the multi-node configuration.

Procedure:

1. Execute the `adbstop` command on the master node.

   When the `adbstop` command is executed on the master node, the HADB servers on all nodes terminate normally.
2. Execute HA Monitor's `monend` command on the master node.

   After confirming that the HADB servers on all nodes have terminated normally, execute HA Monitor's `monend` command on the master node.

(b) Forcibly terminating HADB servers in the multi-node configuration

The following shows the procedure for forcibly terminating the HADB servers in a multi-node configuration.

Procedure:

1. Execute the `adbstop --force` command on each slave node.
   Execute the `adbstop --force` command on all slave nodes.
   In addition, confirm that the `adbstop --force` command terminated on all slave nodes.
2. Execute the `adbstop --force` command on the master node.
   Confirm that the `adbstop --force` command terminated.
3. Execute HA Monitor's `monend` command on the master node.

(2) Terminating an HADB server on a specific node

(a) Normally terminating the HADB server on the master node only

The procedure for normally terminating the HADB server only on the master node differs depending on whether you are using host reset or SCSI reservation for shared disk.

- **When using host reset**
  If you execute the HA Monitor `monswap` command on the master node, only the HADB server on the master node ends normally. At this time, a master node switchover occurs, and the slave node with the highest priority becomes the master node.
  Furthermore, if you execute the `monswap` command, all running transactions on the master node are canceled, and all the connections running those transactions are terminated. All connections that have not yet run any transactions are also terminated. After the connection termination processing is complete, the master node's HADB server begins its termination processing.

  > **Important**
  >
  > If you accidentally terminate the master node's HADB server by using the `adbstop --node` command, execute the `monswap` command on that node.

- **When using SCSI reservation for shared disk**
  Procedure:
  1. Execute the `adbstop --cancel --node` command on the master node.
  2. Execute the HA Monitor command `monswap` on the master node.

  This procedure normally terminates the HADB server only on the master node. At this time, a master node switchover occurs, and the slave node with the highest priority becomes the master node.

(b) Forcibly terminating the HADB server on the master node only

If you execute the `adbstop --force` command on the master node, only the master node's HADB server is forcibly terminated. At this time, a master node switchover occurs, and the slave node with the highest priority becomes the master node.

> **Note**
>
> Recovery processing (rollback) performed on update processing that had been running on the master node you forcibly terminated, is executed on the slave node that becomes the new master node.

(c) Normally terminating the HADB server on a slave node only

Execute the `adbstop --node` command on a slave node to normally terminate the HADB server of the slave node on which the command is executed.

(d) Forcibly terminating the HADB server on a slave node only

Execute the `adbstop --force` command on a slave node to forcibly terminate the HADB server of the slave node on which the command is executed.

(3) Termination modes for HADB servers in a multi-node configuration

The following shows the termination modes for the HADB servers in a multi-node configuration.

Table 16-6: Termination modes for HADB servers in a multi-node configuration

No.	Termination mode	Command to be executed	Explanation
1	Normal termination of HADB servers in a multi-node configuration	`adbstop`	Last termination mode Executing the `adbstop` command on the master node terminates normally the HADB servers in the multi-node configuration. There are four types of normal termination, and the termination process differs depending on the option specified. For details, see (2) Option specification and normal termination types in 10.2.2 Terminating the HADB server.
2	Normal termination of the HADB server of a specific node	• In the case of a master node: `monswap`# • In the case of a slave node: `adbstop --node`	Only the HADB server of the slave node on which the command was executed is terminated normally. If the master node is terminated normally, one of the slave node switches over to become the master node.
3	Forced termination of the HADB server of a specific node	`adbstop --force`	In this case, the HADB server of the node on which you executed the `adbstop` command is immediately terminated without waiting for the completion of ongoing transactions. If the master node is terminated forcibly, one of the slave node switches over to become the master node. At this time, recovery processing (rollback) performed on update processing that had been running on the master node, is executed on the slave node that becomes the new master node. When only the master node remains, and the master node's HADB server is terminated forcibly, recovery processing (rollback) is performed on update processing when the HADB servers are restarted in the multi-node configuration.
4	Abnormal termination of the HADB server of a specific node	--	• When the master node HADB server is terminated abnormally If the master node terminates abnormally, it is separated from the multi-node configuration, and one of the slave node switches over to become the master node. At this time, recovery processing (rollback) performed on update processing that had been running on the master node, is executed on the slave node that becomes the new master node. When only the master node remains, and the master node's HADB server terminates abnormally, recovery processing

No.	Termination mode	Command to be executed	Explanation
			(rollback) is performed on update processing when the HADB servers are restarted in the multi-node configuration. • When a slave node's HADB server terminates abnormally The slave node that terminated abnormally is separated from the multi-node configuration. At this time, transactions are rolled back on the master node.

Legend:

--: Not applicable.

#

Execute this command when using host reset.

When using SCSI reservation for shared disk, execute the `adbstop --cancel --node` command before executing the `monswap` command.

16.4.3 HADB server operation modes when the multi-node function is used

On the master node, you can use the `adbchgsrvmode` command to change the HADB server operation mode.

In the case of a slave node, its HADB server operation mode is always the quiescence mode. Therefore, you cannot execute the `adbchgsrvmode` command.

In addition, to check the HADB server operation mode, execute the `adbls -d srv` command on the node running the HADB server you want to check. For details, see 10.2.3 HADB server operation modes.

Note that if a master node switchover occurs, the HADB server operation mode is carried over to the new master node.

16.5 Application operations (when the multi-node function is being used)

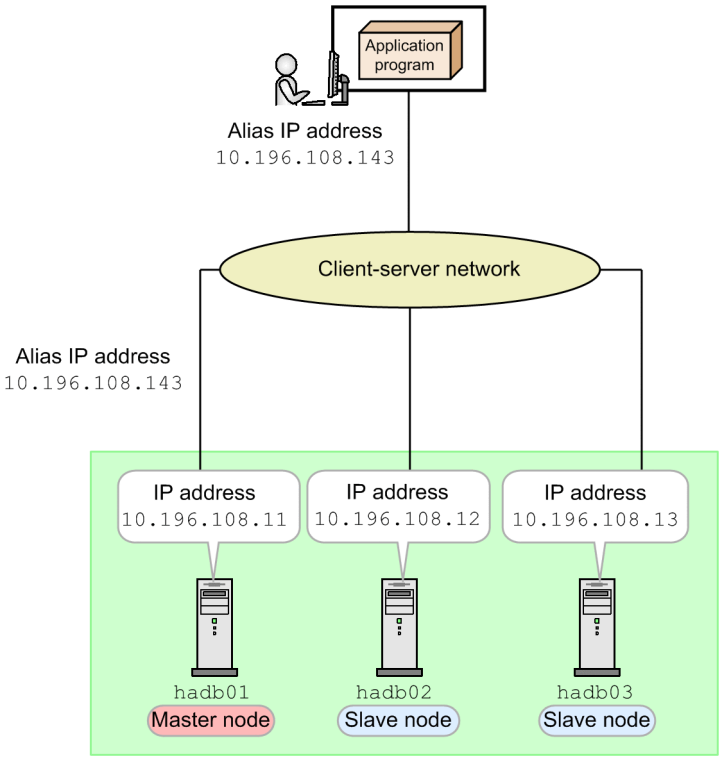
This section provides notes related to application operations when the multi-node function is used. For details about how to create an application and other information, see the *HADB Application Development Guide*.

16.5.1 Connecting to the HADB server

When the multi-node function is used, applications must use an alias IP address to connect to the HADB server on the master node. For details about how to set an alias IP address, see the manual *HA Monitor for Linux(R) (x86)*.

The following figures show HADB servers in the multi-node configuration and application configuration examples.

Figure 16-2: HADB servers in the multi-node configuration and application configuration example



16.5.2 Maximum number of concurrent connections when the multi-node function is used

The maximum number of concurrent connections when the multi-node function is used is the value specified in the `adb_sys_max_users` operand in the server definition (the value specified for the `adb_sys_max_users` operand needs to be the same for each HADB server on each node).

Applications cannot connect to the HADB servers in the multi-node configuration beyond the maximum number of concurrent connections (an error occurs).

For details about the `adb_sys_max_users` operand in the server definition, see the description of the adb_sys_max_users operand in 7.2.1 Operands related to system configuration (set format).

16.5.3 Node that executes transactions

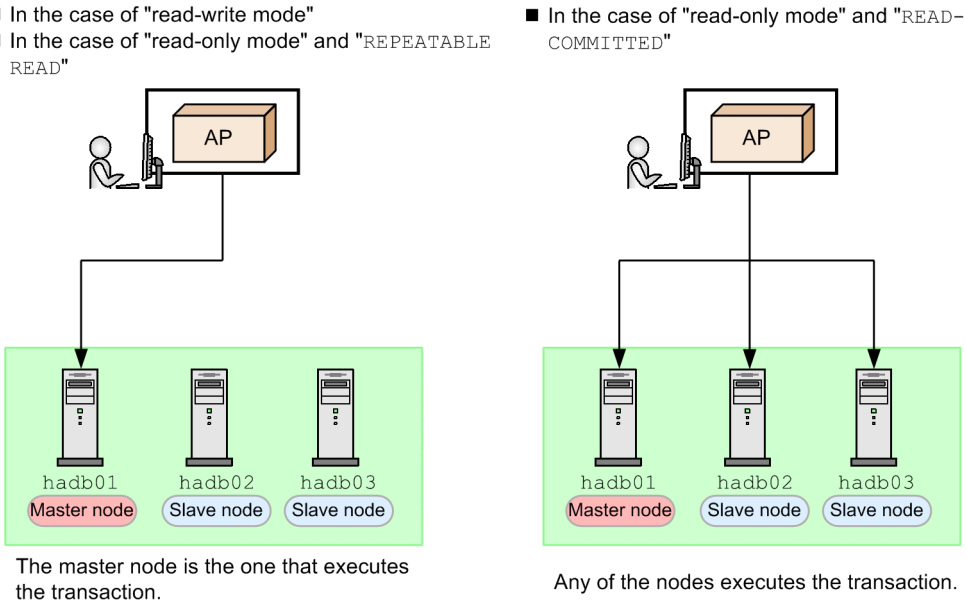
The node that executes transactions differs depending on the specified transaction access mode and transaction isolation level. The following table shows the relationship between a given transaction access mode and transaction isolation level, and the node that executes transactions.

Table 16-7: Relationship between the transaction access mode and transaction isolation level, and the node that executes transactions

No.	Transaction access mode specification	Transaction isolation level specification	Node that executes transactions
1	Read/write mode	`READ COMMITTED`	Master node
2		`REPEATABLE READ`	
3	Read-only mode	`READ COMMITTED`	Master node or slave node
4		`REPEATABLE READ`	Master node

The node that executes transactions is shown in the following figure.

Figure 16-3: Node that executes transactions



In the case of an application program that executes only retrieval SQL statements, specifying the following settings can help effectively utilize the resources of slave nodes:

- Setting the transaction access mode to read-only mode
- Setting the transaction isolation level to `READ COMMITTED`

The following table shows how to specify the transaction access mode and transaction isolation level.

Table 16-8: Specifying the transaction access mode and transaction isolation level

No.	Application type	Transaction access mode specification method	Transaction isolation level specification method
1	Application that uses the JDBC driver	• `adb_clt_trn_access_mode` in the system properties	• `adb_clt_trn_iso_lv` in the system properties

No.	Application type	Transaction access mode specification method	Transaction isolation level specification method
		• adb_clt_trn_access_mode in the user properties • adb_clt_trn_access_mode for the connection URL • setReadOnly method of the Connection interface	• adb_clt_trn_iso_lv in the user properties • adb_clt_trn_iso_lv for the connection URL • setTransactionIsolation method of the Connection interface
2	Application that uses the ODBC driver	• The adb_clt_trn_access_mode operand in the client definition • SQLSetConnectAttr() or SQLSetConnectAttrW()	• The adb_clt_trn_iso_lv operand in the client definition • SQLSetConnectAttr() or SQLSetConnectAttrW()
3	Application that uses CLI functions	• The adb_clt_trn_access_mode operand in the client definition • a_rdb_SQLSetConnectAttr()	• The adb_clt_trn_iso_lv operand in the client definition • a_rdb_SQLSetConnectAttr()

For details about the specification method, see the *HADB Application Development Guide*.

Also note that the nodes executing transactions are automatically determined by the master node's HADB server based on the following factors:

- Transaction access mode
- Transaction isolation level
- Number of free threads

Consequently, the executing node might change for every transaction. However, if the cursor is a holdable cursor, the executing node does not change while the cursor is open.

Important

When transactions in read/write mode, or some commands are currently being executed, transactions in read-only mode are also executed on the master node (slave nodes are not used). Therefore, to effectively utilize the resources of slave nodes, as far as possible make sure to finish transactions in read/write mode frequently on an individual basis, rather than in bulk.

For what is meant by "some commands", see (4) Restrictions on simultaneously executing commands with transactions in 2.19.2 Nodes on which transactions and commands are executed.

16.5.4 When a node failure occurs

When a node failure occurs, the transactions that were executing on the node where the failure occurred are rolled back.

Recovery processing (rollback) performed on update processing that had been running on the master node you forcibly terminated, is executed on the slave node that becomes the new master node. During this recovery processing, if an application program running on a slave node accesses a target page of the recovery processing, that application program's processing might result in an error. At this time, the KFAA31733-E message is output. In this case, follow the steps given in the message.

Furthermore, application programs that were executing transactions on the node where the failure occurred are disconnected from the HADB servers in the multi-node configuration. To re-execute the transactions, connect again to the HADB servers in the multi-node configuration.

If a node failure occurs on the master node, application programs that were not executing transactions on any node at the time the failure occurred, are disconnected from the HADB servers in the multi-node configuration.

16.6 Locking operations (when the multi-node function is being used)

In a multi-node configuration, the HADB servers on all nodes share the locked resources described in 2.10.2 Locked resources. Consequently, if a transaction that is being executed by an HADB server on some node reserves a specific locked resource in the exclusive mode, no other transaction on an HADB server on another node can reserve that locked resource.

For further details about locking, see 2.10 Locking.

16.7 Switching over the master node by using a command

You can switch over the master node by executing a command. The procedure differs depending on whether you are using host reset or SCSI reservation for shared disk.

- When using host reset
  You can switch over the master node by executing the HA Monitor `monswap` command on the master node.
- When using SCSI reservation for shared disk
  You can use the following procedure to switch over the master node:
  1. Execute the `adbstop --cancel --node` command on the master node.
  2. Execute the HA Monitor `monswap` command on the master node.

The slave node that becomes the new master node is decided based on the specifications in the `initial` and `standbypri` operands in the `servers` file of HA Monitor. For details about the `initial` and `standbypri` operands, see (3) Specifying the servers file in 16.3.4 Setting up an HA Monitor environment, and *Server environment definition (servers)* in the manual *HA Monitor for Linux(R) (x86)*.

The node where the command was executed is separated from the multi-node configuration.

Note

- For details about the `monswap` command, see *Commands* in the manual *HA Monitor for Linux(R) (x86)*.
- Switching over the master node through commands uses the HA Monitor planned hot standby function.

16.8 Backing up a database (when the multi-node function is being used)

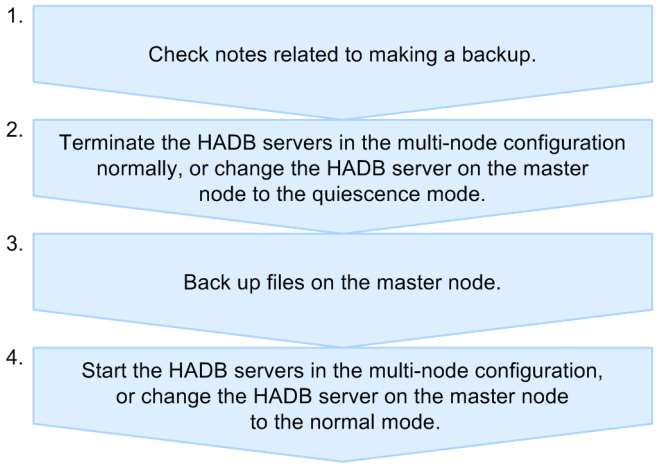
This section explains how to make a backup and recover a database from a backup when the multi-node function is being used.

For details about how to make a backup and recover a database from a backup when you are not using the multi-node function, see 10.3 Backing up a database.

16.8.1 Backup acquisition method

The following figure shows the general procedure for backing up a database.

Figure 16-4: General procedure for backing up a database (when the multi-node function is being used)



The following subsections explain these steps in detail.

(1) Check notes related to making a backup

Check the notes related to making a backup with reference to 10.3.1 Backup acquisition method.

(2) Terminate the HADB servers in the multi-node configuration normally, or change the HADB server on the master node to the quiescence mode

Perform either of the following operations:

- Execute the `adbstop` command and HA Monitor's `monend` command to terminate the HADB servers in the multi-node configuration normally.
- Execute the `adbchgsrvmode` command on the master node to change the HADB server on the master node to the quiescence mode.

If you do not perform either of the above operations, database consistency cannot be achieved when the database is restored from a backup.

To check the status of an HADB server, execute the `adbls` command on the desired node.

(3) Back up files on the master node

The table below lists the files that need to be backed up on the master node.

If the file to be backed up is a symbolic link file, you need a copy of the linked file or block special file.

If the HADB servers in a multi-node configuration are normally terminated, make sure that the system directory is mounted to `$DBDIR/ADBSYS` before copying. If the directory is not mounted, execute the OS's `mount` command to mount the system directory to `$DBDIR/ADBSYS`.

Do not use the backed-up files as sparse files.

Table 16-9: List of files and directories that need to be backed up

No.	File that needs to be backed up	File storage location	File type
1	Command status file	`$DBDIR/ADBSYS/ADBUTL`	Regular file
2	System log file	`$DBDIR/ADBSYS/ADBSLG`	Regular file
3	Status file	`$DBDIR/ADBSYS/ADBSTS`	Regular file
4	Master directory DB area file	`$DBDIR/ADBMST`	Block special file
5	Dictionary DB area file	`$DBDIR/ADBDIC`	Block special file
6	System-table DB area file	`$DBDIR/ADBSTBL`	Block special file
7	All files comprising the data DB area	`$DBDIR/DBAREA`[#1]	Block special file
8	Archive file	Archive directory[#2]	Regular file
9	Synonym dictionary file	Directory for storing synonym dictionary files[#3]	Regular file

#1

This is the name specified by the user in the `adbinit` or `adbmodarea` command.

#2

Refers to the directory specified in `ARCHIVEDIR`, which is in the chunk archive specification of the `CREATE TABLE` statement.

#3

Refers to the directory specified in the `adb_syndict_storage_path` operand in the server definition.

Note

The following files do not need to be backed up:

- Work table DB area file (`$DBDIR/ADBWRK`)
- Files under the work directory (`$DBDIR/ADBWORK`)
- Files under the output directory for error information (core files) (`$DBDIR/SPOOL`)

  This is the directory specified in the `adb_core_path` operand, if that operand is specified in the server definition.

Note that there is no problem if you back up these files along with the files listed in the table above, and recover them from a backup.

Note

There is no need to back up the audit trail files in the audit trail directory (the directory specified for the `adb_audit_log_path` operand in the server definition). Instead of backing up the audit trail files in the audit trail directory, you can move them to the audit trail storage directory. For details about moving audit trail files, see 12.3.1 Moving audit trail files (to audit trail storage directory).

(4) Start the HADB servers in the multi-node configuration, or change the HADB server on the master node to the normal mode

Perform either of the following operations:

- If the `adbstop` command was used to terminate the HADB servers in the multi-node configuration normally before making a backup:

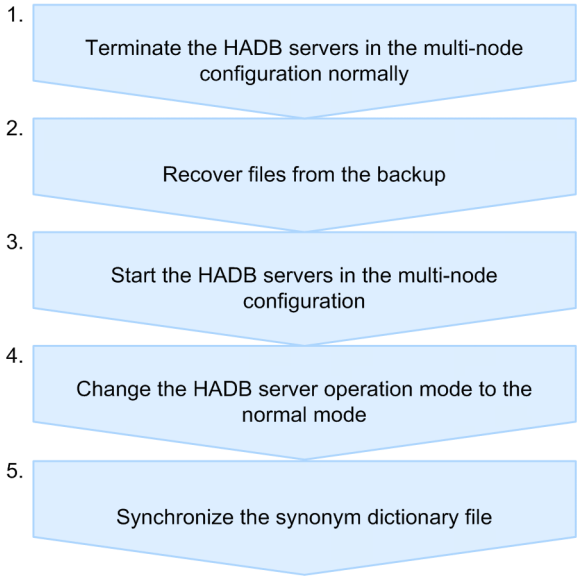
  Execute the `adbstart` command and HA Monitor's `monbegin` command to start the HADB servers in the multi-node configuration.
- If the `adbchgsrvmode` command was used to change the HADB server on the master node to the quiescence mode before making a backup:

  Execute the `adbchgsrvmode` command on the master node to change the HADB server on the master node to the normal mode.

16.8.2 Recovering a database from a backup

The following figure shows the general procedure for recovering a database from a backup.

Figure 16-5: General procedure for recovering a database from a backup (when the multi-node function is being used)



When recovering a database from a backup, also see 10.3.2 Recovering the database from the backup.

The following subsections explain the steps shown in the above figure.

(1) Terminate the HADB servers in the multi-node configuration normally

Execute the `adbstop` command and HA Monitor's `monend` command to terminate the HADB servers in the multi-node configuration normally.

If you perform database recovery without first terminating normally the HADB servers in the multi-node configuration, database compatibility cannot be achieved.

(2) Recover files from the backup

Recover the files from the backup you acquired in (3) Back up files on the master node under 16.8.1 Backup acquisition method.

Furthermore, before recovering the files from backup, make sure that the system directory is mounted to `$DBDIR/ADBSYS`. If the directory is not mounted, execute the OS's `mount` command to mount the system directory to `$DBDIR/ADBSYS`.

(3) Start the HADB servers in the multi-node configuration

Execute the `adbstart` command and HA Monitor's `monbegin` command to start the HADB servers in the multi-node configuration.

(4) Change the HADB server operation mode to the normal mode

If an HADB server is started after the database has been recovered from a backup that was acquired in the quiescence mode, the HADB server remains in the quiescence mode. In this case, execute the `adbchgsrvmode` command on the master node to change the HADB server operation mode of the master node to the normal mode.

(5) Synchronize the synonym dictionary file

Execute the `adbsyndict` command to synchronize the synonym dictionary file. For details about how to synchronize the synonym dictionary file, see 16.24.5 Synchronizing synonym dictionary files.

16.8.3 Backup operation example (using OS commands)

This subsection provides an example of a backup operation using OS commands (an operation example for acquiring a full backup).

(1) System configuration examples

In the multi-node configuration consisting of three nodes (`hadb01`, `hadb02`, and `hadb03`), the master node is set as `hadb01`. The DB directory, the archive directory, the directory for storing synonym dictionary files, and the audit trail directory have the following directory structures:

■ Directory structure of `hadb01`

- DB directory configuration

```
/HADB/db
 ├ADBDIC ... Block special file (10 MB in size)
 ├ADBMST ... Block special file (512 MB in size)
 ├ADBSTBL ... Block special file (512 MB in size)
```

```
 ├ADBWORK ... Directory on the local file system
 ├ADBWRK ... Block special file (2 GB in size)
 ├ADBUTBL01 ... Block special file (4 GB in size)
 ├ADBUIDX01 ... Block special file (2 GB in size)
 ├SPOOL ... Directory on the local file system
 └ADBSYS ... System directory
```

- Archive directory structure

```
/HADB/archive
```

The archive directory is created when defining an archivable multi-chunk table.

- Synonym dictionary file storage directory structure

```
/mnt/syndict
```

The synonym dictionary file storage directory is created in environments where synonym searches will be performed.

- Audit trail directory structure

```
/HADB/audit
```

The audit trail directory is created in environments where the audit trail facility is used.

■ Directory structure of `hadb02`

- DB directory configuration

```
/HADB/db
 ├ADBDIC ... Block special file
 ├ADBMST ... Block special file
 ├ADBSTBL ... Block special file
 ├ADBWORK ... Directory on the local file system
 ├ADBWRK ... Block special file
 ├ADBUTBL01 ... Block special file
 ├ADBUIDX01 ... Block special file
 ├SPOOL ... Directory on the local file system
 └ADBSYS ... Mount point to the file system of the system directory
```

- Archive directory structure

```
/HADB/archive
```

The archive directory is created when defining an archivable multi-chunk table.

- File system mount point for synonym dictionary file storage directory

```
/mnt/syndict
```

The synonym dictionary file storage directory is created in environments where synonym searches will be performed.

- Audit trail directory structure

```
/HADB/audit
```

The audit trail directory is created in environments where the audit trail facility is used.

■ Directory structure of `hadb03`

- DB directory configuration

```
/HADB/db
 ├ADBDIC ... Block special file
 ├ADBMST ... Block special file
```

```
 ├ADBSTBL ... Block special file
 ├ADBWORK ... Directory on the local file system
 ├ADBWRK ... Block special file
 ├ADBUTBL01 ... Block special file
 ├ADBUIDX01 ... Block special file
 ├SPOOL ... Directory on the local file system
 └ADBSYS ... Mount point to the file system of the system directory
```

- Archive directory structure

```
/HADB/archive
```

The archive directory is created when defining an archivable multi-chunk table.

- File system mount point for synonym dictionary file storage directory

```
/mnt/syndict
```

The synonym dictionary file storage directory is created in environments where synonym searches will be performed.

- Audit trail directory structure

```
/HADB/audit
```

The audit trail directory is created in environments where the audit trail facility is used.

(2) Making a backup

The following shows the procedure for using OS commands to back up a database.

Procedure:

1. Terminate the HADB servers in the multi-node configuration.[#1]

```
adbstop
KFAA90000-I adbstop processing started.
KFAA91154-I The HADB system was terminated normally.
KFAA90001-I adbstop processing ended. (return code = 0)

monend
```

After normally terminating the HADB servers on all nodes by executing the `adbstop` command on master node `hadb01`, execute HA Monitor's `monend` command.

2. Use OS commands to make a backup.

Back up the following directories by executing the OS's `cp` and `dd` commands on the master node `hadb01`:

- Backing up the DB directory

```
cp -r /HADB/db/ADBSYS /HADB_bkup/db/ADBSYS
dd if=/HADB/db/ADBMST of=/HADB_bkup/db/ADBMST bs=524288
dd if=/HADB/db/ADBDIC of=/HADB_bkup/db/ADBDIC bs=524288
dd if=/HADB/db/ADBSTBL of=/HADB_bkup/db/ADBSTBL bs=524288
dd if=/HADB/db/ADBUTBL01 of=/HADB_bkup/db/ADBUTBL01 bs=524288
dd if=/HADB/db/ADBUIDX01 of=/HADB_bkup/db/ADBUIDX01 bs=524288
```

The backup data is stored in the `/HADB_bkup/db` directory.

If the system directory is not mounted in `$DBDIR/ADBSYS`, mount it by executing the operating system's `mount` command.

When data in a block special file is backed up, the size of the backup will be larger than the actual size of the data being used because the entire volume is copied.

- Backing up the archive directory

```
cp -r /HADB/archive /HADB_bkup/archive
```

If an archivable multi-chunk table is defined, you need to back up the archive directory.

If the archive directory was created on an NFS server, you do not need to back up the archive directory on slave nodes.

- Backing up the directory for storing synonym dictionary files

```
cp -r /mnt/syndict /HADB_bkup/syndict
```

In environments where synonym searches are used, you need to back up the directory for storing synonym dictionary files.

> **Note**
>
> Because the audit trail files in the audit trail directory are regularly moved to the audit trail storage directory, you do not need to back up the audit trail directory.

3. Start the HADB servers in the multi-node configuration.#2

```
adbstart
```

On all nodes from `hadb01` to `hadb03`, execute the `adbstart` command and the HA Monitor `monbegin` command. After executing the `adbstart` command, execute the HA Monitor `monbegin` command from another terminal (command input screen).

> **Important**
>
> Execute the `monbegin` command before the `adbstart` command terminates.

```
monbegin
```

#1

Instead of step 1, you can perform the following process:

Execute the `adbchgsrvmode` command on master node `hadb01` to change the HADB server operation mode of the master node to the quiescence mode.

```
adbchgsrvmode --quiescence
```

#2

If the process described in footnote 1 is performed in step 1, execute the `adbchgsrvmode` command on master node `hadb01` to change the HADB server operation mode of the master node to the normal mode.

```
adbchgsrvmode --normal
```

(3) Recovering the database from a backup

The following shows the procedure for recovering the database from a backup.

Procedure:

1. Terminate the HADB servers in the multi-node configuration.

```
adbstop
KFAA90000-I adbstop processing started.
KFAA91154-I The HADB system was terminated normally.
KFAA90001-I adbstop processing ended. (return code = 0)

monend
```

After normally terminating the HADB servers on all nodes by executing the `adbstop` command on master node `hadb01`, execute HA Monitor's `monend` command.

2. Use OS commands to recover the database.

   Recover the following directories from the backup by executing the OS's `cp` and `dd` commands on the master node `hadb01`:

   - Recovering the DB directory

     ```
     cp -r /HADB_bkup/db/ADBSYS /HADB/db/ADBSYS
     dd if=/HADB_bkup/db/ADBMST of=/HADB/db/ADBMST bs=524288
     dd if=/HADB_bkup/db/ADBDIC of=/HADB/db/ADBDIC bs=524288
     dd if=/HADB_bkup/db/ADBSTBL of=/HADB/db/ADBSTBL bs=524288
     dd if=/HADB_bkup/db/ADBUTBL01 of=/HADB/db/ADBUTBL01 bs=524288
     dd if=/HADB_bkup/db/ADBUIDX01 of=/HADB/db/ADBUIDX01 bs=524288
     ```

     The backup file is stored in the directory `/HADB_bkup/db`.

     If the system directory is not mounted in `$DBDIR/ADBSYS`, mount it by executing the operating system's `mount` command.

   - Recovering the archive directory

     ```
     rm -r /HADB/archive/*
     cp -r /HADB_bkup/archive/* /HADB/archive
     ```

     If an archivable multi-chunk table is defined, you need to recover the archive directory.

     If the archive directory was created on an NFS server, you do not need to recover the archive directory on slave nodes.

   - Recovering the directory for storing synonym dictionary files

     ```
     rm -r /mnt/syndict/*
     cp -r /HADB_bkup/syndict/* /mnt/syndict
     ```

     In environments where synonym searches are used, you need to recover the directory for storing synonym dictionary files.

3. Start the HADB servers in the multi-node configuration.

```
adbstart
```

On all nodes from `hadb01` to `hadb03`, execute the `adbstart` command and the HA Monitor `monbegin` command. After executing the `adbstart` command, execute the HA Monitor `monbegin` command from another terminal (command input screen).

Important

Execute the `monbegin` command before the `adbstart` command terminates.

```
monbegin
```

After the HADB servers in the multi-node configuration start, if the HADB server operation mode of the master node is the quiescence mode, execute the `adbchgsrvmode` command on master node `hadb01` to change the HADB server operation mode of the master node to the normal mode.

```
adbchgsrvmode --normal
```

4. Synchronize the synonym dictionary files.

```
adbsyndict -s
```

Execute the `adbsyndict` command on the master node `hadb01` to synchronize the synonym dictionary files. This step is only required in environments that use synonym searches.

16.8.4 Backup operation example (using ShadowImage)

This subsection provides an example of a backup operation using ShadowImage.

(1) System configuration examples

In the multi-node configuration consisting of three nodes (`hadb01`, `hadb02`, and `hadb03`), the master node is set as `hadb01`. The DB directory, the archive directory, the directory for storing synonym dictionary files, and the audit trail directory have the following directory structures:

■ Directory structure of `hadb01`

- DB directory configuration

```
/HADB/db
 ├ADBDIC ... Block special file (using LU002)
 ├ADBMST ... Block special file (using LU001)
 ├ADBSTBL ... Block special file (using LU003)
 ├ADBWORK ... Directory on the local file system (in /dev/sdb1)
 ├ADBWRK ... Block special file (using /dev/mapper/WRK001)
 ├ADBUTBL01 ... Block special file (using LU004)
 ├ADBUIDX01 ... Block special file (using LU005)
 ├SPOOL ... Directory on the local file system (in /dev/sdb1)
 └ADBSYS ... System directory (file system on FS001)
```

- Archive directory structure

```
/HADB/archive
```

The archive directory is created when defining an archivable multi-chunk table.

- Synonym dictionary file storage directory structure (file system on `FS002`)

```
/mnt/syndict
```

The synonym dictionary file storage directory is created in environments where synonym searches will be performed.

- Audit trail directory structure

```
/HADB/audit
```

The audit trail directory is created in environments where the audit trail facility is used.

■ Directory structure of `hadb02`

- DB directory configuration

```
/HADB/db
 ├ADBDIC ... Block special file (using LU002)
 ├ADBMST ... Block special file (using LU001)
 ├ADBSTBL ... Block special file (using LU003)
 ├ADBWORK ... Directory on the local file system (in /dev/sdb1)
 ├ADBWRK ... Block special file (using /dev/mapper/WRK002)
 ├ADBUTBL01 ... Block special file (using LU004)
 ├ADBUIDX01 ... Block special file (using LU005)
 ├SPOOL ... Directory on the local file system (in /dev/sdb1)
 └ADBSYS... Mount point to the system directory
```

- Archive directory structure

```
/HADB/archive
```

  The archive directory is created when defining an archivable multi-chunk table.

- File system mount point for synonym dictionary file storage directory

```
/mnt/syndict
```

  The synonym dictionary file storage directory is created in environments where synonym searches will be performed.

- Audit trail directory structure

```
/HADB/audit
```

  The audit trail directory is created in environments where the audit trail facility is used.

■ Directory structure of `hadb03`

- DB directory configuration

```
/HADB/db
 ├ADBDIC ... Block special file (using LU002)
 ├ADBMST ... Block special file (using LU001)
 ├ADBSTBL ... Block special file (using LU003)
 ├ADBWORK ... Directory on the local file system (in /dev/sdb1)
 ├ADBWRK ... Block special file (using /dev/mapper/WRK003)
 ├ADBUTBL01 ... Block special file (using LU004)
 ├ADBUIDX01 ... Block special file (using LU005)
 ├SPOOL ... Directory on the local file system (in /dev/sdb1)
 └ADBSYS... Mount point to the system directory
```

- Archive directory structure

```
/HADB/archive
```

  The archive directory is created when defining an archivable multi-chunk table.

- File system mount point for synonym dictionary file storage directory

```
/mnt/syndict
```

  The synonym dictionary file storage directory is created in environments where synonym searches will be performed.

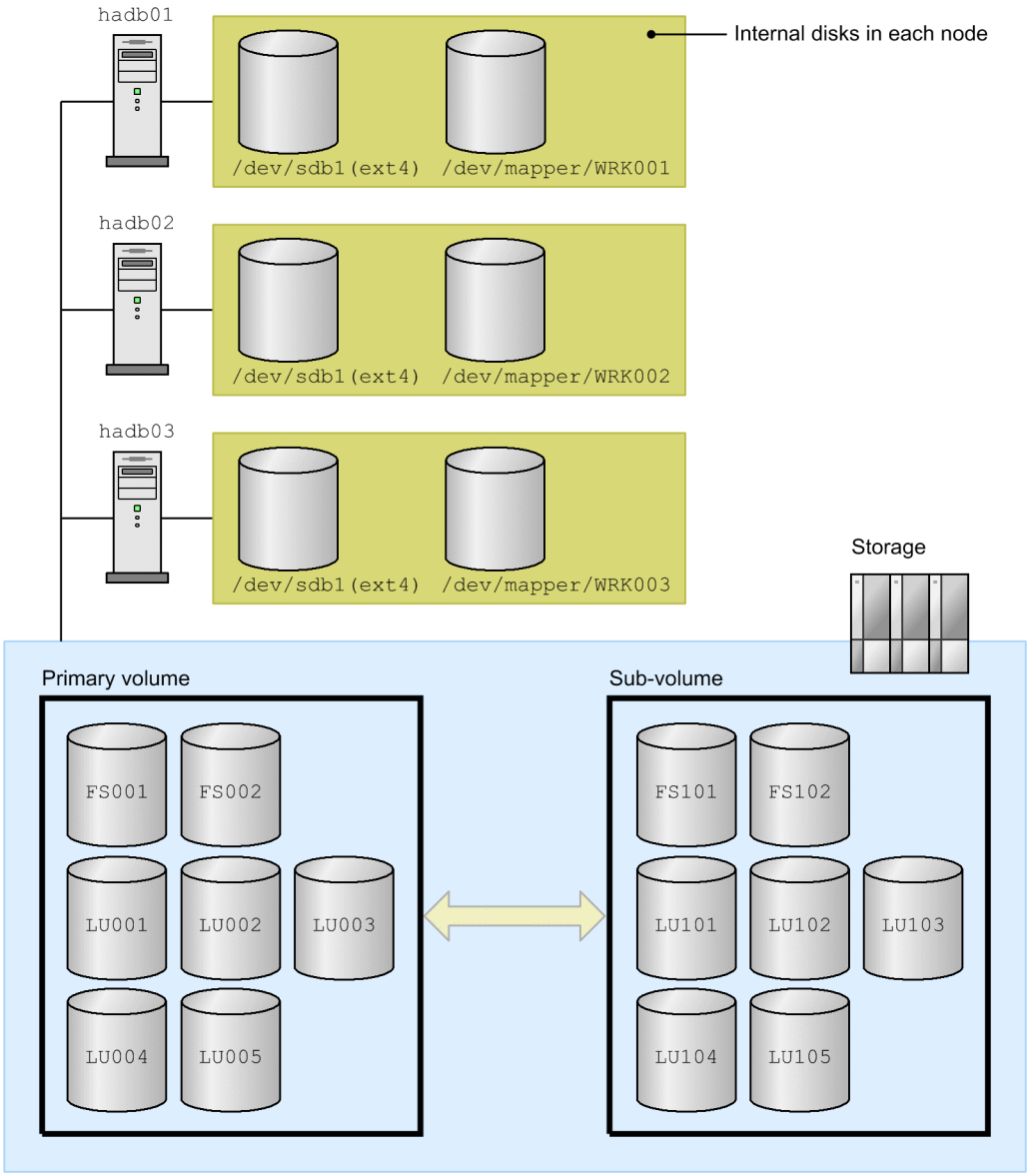
- Audit trail directory structure

```
/HADB/audit
```

  The audit trail directory is created in environments where the audit trail facility is used.

■ Logical unit configuration

The following figure shows an example of storage configuration in the multi-node configuration:



Explanation

- The file system for the system directory is created on `FS001`.
- The file system for storing synonym dictionary files is created on `FS002`.
- `LU001` to `LU005` are used for DB area files.
- The sub-volumes that correspond to the primary volumes `FS00X` and `LU00X` are `FS10X` and `LU10X`.

(2) Making a backup

Back up `FS001`, `FS002`, and all logical units `LU001` to `LU005` by following the procedure described in (3) Making a backup under 10.3.4 Backup operation example (using ShadowImage).

(3) Recovering the database from a backup

Recover `FS001`, `FS002`, and all logical units `LU001` to `LU005` by following the procedure described in (4) Recovering a database from a backup under 10.3.4 Backup operation example (using ShadowImage).

If you are using synonym searches, after recovering the database, execute the `adbsyndict` command on the master node `hadb01` to synchronize the synonym dictionary files. For details about how to synchronize the synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.

16.9 Status monitoring (when the multi-node function is being used)

This section explains the system monitoring method that is unique to cases in which the multi-node function is being used.

16.9.1 Checking information on all nodes

By executing the `adbls -d node` command on an active node, you can check the node type and status information of all nodes. For details about the information that is output when the `adbls -d node` command is executed, see *adbls -d node (Display the HADB Server Status on Each Node)* in the manual *HADB Command Reference*.

16.9.2 Checking the HADB server's status

When the multi-node function is being used, messages that are generated by the local node are output to the local node's message log file and syslog. By comparing the messages that are output to the individual nodes by their times, for example, you can check the statuses of all nodes. Therefore, use a protocol such as the NTP to synchronize the times among the individual nodes.

For details about how to check messages, see 10.4 Checking the messages.

16.9.3 When application or command execution takes a long time

If the processing time of an application or command is longer than expected, use the following procedure to identify the node that is executing the application or command.

Procedure:

1. Execute the `adbls -d cnct` command.

   Refer to the command execution result to determine the node number of the node that is executing the application or command.

2. Execute the `adbls -d node` command.

   Refer to the command execution result to identify the node that corresponds to the node number.

3. Investigate the cause.

   After identifying the node, investigate the cause of the issue based on the explanation in 10.8 Checking the transaction processing status.

For details about the information that is output when the `adbls -d cnct` command is executed, see *adbls -d cnct (Display the Connection Status)* in the manual *HADB Command Reference*.

For details about the information that is output when the `adbls -d node` command is executed, see *adbls -d node (Display the HADB Server Status on Each Node)* in the manual *HADB Command Reference*.

16.9.4 When startup or termination processing of the HADB server takes a long time

If startup or termination processing of the HADB server is taking longer than expected, investigate the cause using one of the following methods:

- Investigate the cause from the output message.

  Check the message that is output when the HADB server starts or terminates based on the content of each node's message log file.
- Investigate the cause by executing the `adbls -d srv` command.

  For details about the information that is output when the `adbls -d srv` command is executed, see *adbls -d srv (Display the HADB Server Status)* in the manual *HADB Command Reference*.

16.9.5 Checking the status of the HADB server on each node

You can use the `adbls -d srv` command to check the status of the HADB server on each node.

For details about the information that is output when the `adbls -d srv` command is executed, see *adbls -d srv (Display the HADB Server Status)* in the manual *HADB Command Reference*.

16.9.6 Checking the memory usage

To check the usage status of the memory used by HADB servers, execute the `adbstat` command. For details about how to check memory usage, see 10.6.1 Checking the usage status of all memory.

Executing the `adbstat` command outputs the maximum usage of all memory (`Total_memory_max_size`) separately for each node. Since the value of `Total_memory_max_size` varies from one node to another, check this value on all nodes.

16.10 DB area operations (when the multi-node function is being used)

This section explains how to use DB areas when the multi-node function is being used.

16.10.1 Global buffer allocation

When the multi-node function is being used, the global buffers are independent among the individual nodes. Therefore, use the `adbbuff` operand in the server definition to allocate a global buffer to the HADB server on each node.

For details about the `adbbuff` operand in the server definition, see 7.2.11 Operands and options related to global buffers (command format).

16.10.2 Checking the database status and usage

To check the database status and usage when the multi-node function is being used, execute the `adbdbstatus` command on the master node.

An error occurs if you execute the `adbdbstatus` command on a slave node (the message `KFAA31898-E` is output).

For details about how to check the database status and usage based on the output of the `adbdbstatus` command, see 10.9 Checking the database status and usage.

16.10.3 Adding, deleting, or expanding data DB areas (when the multi-node function is being used)

This section explains how to use the `adbmodarea` command to add, delete, or expand data DB areas when the multi-node function is being used.

(1) Adding data DB areas

For details about how to add data DB areas, see 11.10.1 Adding data DB areas.

The `adbmodarea` command for adding data DB areas must be executed on the master node.

> **Important**
>
> If there are any nodes that have been separated from the multi-node configuration, you cannot add data DB areas. Add data DB areas after returning the separated nodes to the multi-node configuration.

If you have added any data DB areas in the period between deleting and re-adding a node due to a server machine failure, you need to re-create the DB directory. For details about how to do so, see 16.17.1 Adding nodes.

Note

If the HADB server terminates abnormally while adding a data DB area, a symbolic link to the data DB area file you tried to add might remain in that node's DB directory. Even if a symbolic link remains, this will not cause problems in operation.

■ When using SCSI reservation for shared disk

When using SCSI reservation for shared disk, you need to change the specification of the operands in the `servers` file of HA Monitor before adding the data DB area. To add a data DB area:

Procedure

1. Terminate the HADB servers in the multi-node configuration normally.

   For details about how to terminate the HADB server, see 16.4.2 Terminating HADB servers in the multi-node configuration.

2. Change the specification of the `servers` file of HA Monitor.

   Add the device name of the disk to be added in the `scsi_device` or `dmmp_device` operand. Change the specification in the HA Monitor `servers` file on all nodes.

3. Start the HADB servers in the multi-node configuration.

   For details about how to start the HADB server, see 16.4.1 Starting the HADB servers in the multi-node configuration.

4. Execute the `adbmodarea` command to add the data DB area.

   For details about how to add data DB areas, see 11.10.1 Adding data DB areas.

(2) Deleting a data DB area

For details about how to delete data DB areas, see 11.10.2 Deleting a data DB area.

The `adbmodarea` command for deleting data DB areas must be executed on the master node.

Note

If any nodes are stopped at the time a data DB area is deleted, those nodes' DB directories might still contain symbolic links to the deleted DB area's data DB area files. Even if a symbolic link remains, this will not cause problems in operation.

■ When using SCSI reservation for shared disk

When using SCSI reservation for shared disk, you need to change the specification of the operands in the `servers` file of HA Monitor after deleting the data DB area. To delete a data DB area:

Procedure

1. Delete a data DB area by executing the `adbmodarea` command.

   For details about how to delete data DB areas, see 11.10.2 Deleting a data DB area.

2. Terminate the HADB servers in the multi-node configuration normally.

   For details about how to terminate the HADB server, see 16.4.2 Terminating HADB servers in the multi-node configuration.

3. Change the specification of the `servers` file of HA Monitor.

   Delete the device name of the deleted disk from the `scsi_device` or `dmmp_device` operand. Change the specification in the HA Monitor `servers` file on all nodes.

4. Start the HADB servers in the multi-node configuration.

   For details about how to start the HADB servers, see 16.4.1 Starting the HADB servers in the multi-node configuration.

(3) Expanding a data DB area (adding a data DB area file)

For details about how to expand data DB areas, see 11.10.3 Expanding a data DB area (adding a data DB area file).

The `adbmodarea` command for expanding data DB areas must be executed on the master node.

> **Important**
>
> If there are any nodes that have been separated from the multi-node configuration, you cannot expand data DB areas. Expand data DB areas after returning the separated nodes to the multi-node configuration.

If you have expanded any data DB areas in the period between deleting and re-adding a node due to a server machine failure, you need to re-create the DB directory. For details about how to do so, see 16.17.1 Adding nodes.

> **Note**
>
> If the HADB server terminates abnormally while expanding a data DB area, a symbolic link to the data DB area file you tried to add might remain in that node's DB directory. Even if a symbolic link remains, this will not cause problems in operation.

■ When using SCSI reservation for shared disk

When using SCSI reservation for shared disk, you need to change the specification of the operands in the `servers` file of HA Monitor before expanding a data DB area. To expand a data DB area:

Procedure

1. Terminate the HADB servers in the multi-node configuration normally.

   For details about how to terminate the HADB server, see 16.4.2 Terminating HADB servers in the multi-node configuration.

2. Change the specification of the `servers` file of HA Monitor.

   Add the device name of the disk to be added in the `scsi_device` or `dmmp_device` operand. Change the specification in the HA Monitor `servers` file on all nodes.

3. Start the HADB servers in the multi-node configuration.

   For details about how to start the HADB servers, see 16.4.1 Starting the HADB servers in the multi-node configuration.

4. Execute the `adbmodarea` command to expand the data DB area.

   For details about how to expand data DB areas, see 11.10.3 Expanding a data DB area (adding a data DB area file).

16.10.4 Changing the storage location of DB areas

This subsection explains how to change the storage location of DB area files that comprise the data DB area and work table DB area when the multi-node function is being used.

(1) Changing the storage location of the data DB area

The following shows the procedure for changing the storage location of the data DB area files (block special files) that comprise the data DB area when the multi-node function is being used.

Procedure:

1. Terminate the HADB servers in the multi-node configuration normally.
   For details about how to terminate the HADB server, see 16.4.2 Terminating HADB servers in the multi-node configuration.
2. Copy the data DB area file whose storage location you want to change.
   Copy to another area the data DB area file (symbolic link destination) whose storage location you want to change. When doing so, make sure that the copying destination can be accessed from the HADB servers on all nodes.
3. On all nodes, change the symbolic link destination to the copying-destination area.
4. Start the HADB servers in the multi-node configuration.
   For details about how to start the HADB servers, see 16.4.1 Starting the HADB servers in the multi-node configuration.

For a detailed description of the procedures for steps 2 and 3, see 11.10.5 Changing the storage location of data DB area files.

> **Important**
>
> When copying from the pre-change block special file to the post-change block special file by using the `dd` command, only execute the command on one node. In addition, changing of symbolic links must be performed on all nodes.

■ When using SCSI reservation for shared disk

When changing the storage location of a data DB area that is subject to SCSI reservation, you need to change the specification of the operands in the `servers` file of HA Monitor. Therefore, perform the following tasks before starting the HADB servers in the multi-node configuration in step 4 of the preceding procedure:

- Revise the values specified for the `scsi_device` operand or `dmmp_device` operand in the `servers` file. Delete the device name of the original disk, and add the device name of the new disk.

Change the specification in the HA Monitor `servers` file on all nodes.

(2) Changing the storage location of the work table DB area

The following shows the procedure for changing the storage location of the work table DB area files that comprise the work table DB area when the multi-node function is being used.

Procedure:

1. Terminate the HADB servers in the multi-node configuration normally.
2. In the `adb_blk_path_wrk` operand in the server definition, specify the new area to use for the work table DB area file.
3. Start the HADB servers in the multi-node configuration.

16.10.5 Operating work table DB areas (when the multi-node function is being used)

When the multi-node function is being used, the work table DB area consists of different work table DB area files on each node. Therefore, the HADB servers on multiple nodes never access the same work table DB area file.

Estimate the size of the work table DB area for each node based on the explanation in 5.9 Estimating the size of the work table DB area.

16.10.6 Operation when adding disks

If you create a DB area file on an added disk, you need to set each node's OS to recognize the additional disk in advance. To set the OSs to recognize the additional disk, either close all HADB servers in a multi-node configuration and restart all nodes' OSs, or restart each node's OS one at a time by using the following procedure.

Example:

In the case of the following multi-node configuration, add the disk, and then set each node's OS to recognize the additional disk:

- Node 1 (master node)
- Node 2 (slave node)
- Node 3 (slave node)

Procedure:

1. Normally terminate the HADB servers of node 2 and node 3 by using the `adbstop --node` command
2. Restart the OSs of node 2 and node 3
3. Return node 2 and node 3 to the multi-node configuration

   For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.
4. Switch over the master node by using the command, and switch either node 2 or node 3 over to the master node

   For details on how to switch over the master node by using a command, see 16.7 Switching over the master node by using a command.

   At this time, node 1's HADB server stops automatically.
5. Restart the OS of node 1
6. Return node 1 to the multi-node configuration

   For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

   At this time, node 1 is returned as a slave node to the multi-node configuration. To set node 1 as the master node, see (3) Procedure for switching over a returned node to the master node in 16.15.3 Returning a node to the multi-node configuration.

16.11 Schema, table, and index operations (when the multi-node function is being used)

This section explains how to use schemas, tables, and indexes when the multi-node function is being used.

16.11.1 Schema operation (when the multi-node function is being used)

This subsection explains how to use schemas when the multi-node function is being used.

Operations are essentially the same as when the multi-node function is not used. You can define and delete schemas by using the `CREATE SCHEMA` and `DROP SCHEMA` statements, respectively. See *Definition SQL* in the manual *HADB SQL Reference*.

16.11.2 Base table operations (when the multi-node function is being used)

This subsection explains how to use base tables when the multi-node function is being used.

When the multi-node function is being used, the following base table operations must be unique to the multi-node function:

- Storing data in a base table
- Method of releasing a base table from non-updatable status

All other operations are the same as when the multi-node function is not used. For details, see 11.1 Base table operations.

(1) Storing data in a base table (when the multi-node function is being used)

When the multi-node function is being used, you can execute the `adbimport` command on the master node only. You cannot execute this command on a slave node.

For details about the `adbimport` command, see *adbimport (Import Data)* in the manual *HADB Command Reference*.

To check whether it is possible to execute the `adbimport` command and a transaction simultaneously, see (4) Restrictions on simultaneously executing commands with transactions in 2.19.2 Nodes on which transactions and commands are executed.

(2) Releasing a base table from non-updatable status (when the multi-node function is being used)

- **When the `adbimport` or `adbidxrebuild` command terminates abnormally**

  If the `adbimport` or `adbidxrebuild` command terminates abnormally and makes the target base table of the processing non-updatable, re-execute the command that terminated abnormally, and release the base table from its non-updatable status.

If the master node's HADB server terminates abnormally and the master node is switched over, execute the `adbidxrebuild` command on the new master node, and release the base table from its non-updatable status. At this time, execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

For details about the `--create-temp-file` option, see *Specification format for the adbidxrebuild command* under *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

- **When the `adbunarchivechunk` command terminates abnormally**

  If the `adbunarchivechunk` command terminates abnormally and makes the target base table of the processing non-updatable, re-execute the `adbunarchivechunk` command, and release the base table from its non-updatable status.

> **Note**
>
> For details about the non-updatable status of base tables, see 15.8.1 Steps to take when a base table becomes non-updatable.

16.11.3 Viewed table operations (when the multi-node function is being used)

This subsection explains how to use viewed tables when the multi-node function is being used.

Operations are essentially the same as when the multi-node function is not used. For details, see 11.2 Viewed table operations.

16.11.4 Index operations (when the multi-node function is being used)

This subsection explains how to use B-tree indexes, text indexes, and range indexes when the multi-node function is being used.

When the multi-node function is being used, the following operations that are unique to the multi-node function are required for handling B-tree indexes, text indexes, and range indexes:

- Creating an index
- Releasing a B-tree index from unfinished status
- Releasing a text index from unfinished status
- Releasing a range index from unfinished status

All other operations are the same as when the multi-node function is not used. For details, see 11.3 Index operations.

(1) Creating an index (when the multi-node function is being used)

When the multi-node function is being used, you can execute the commands listed below on the master node only. You cannot execute these commands on a slave node.

Targeted commands

- `adbimport` command
- `adbidxrebuild` command

- `adbmergechunk` command
- `adbunarchivechunk` command

For details about these commands, see the manual *HADB Command Reference*.

(2) Releasing a B-tree index from unfinished status (when the multi-node function is being used)

If abnormal termination of the `adbimport` or `adbidxrebuild` command causes a B-tree index to be set to unfinished status, re-execute the command that terminated abnormally, and release the B-tree index from its unfinished status.

If the master node's HADB server terminates abnormally and the master node is switched over, execute the `adbidxrebuild` command on the new master node, and release the B-tree index from its unfinished status. At this time, execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

For details about the `--create-temp-file` option, see *Specification format for the adbidxrebuild command* under *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

For details about the unfinished status of B-tree indexes, see 15.9.1 Steps to take when unfinished status is applied to a B-tree index.

(3) Releasing a text index from unfinished status (when the multi-node function is being used)

If abnormal termination of the `adbimport` or `adbidxrebuild` command causes a text index to be set to unfinished status, re-execute the command that terminated abnormally, and release the text index from its unfinished status.

If the master node's HADB server terminates abnormally and the master node is switched over, execute the `adbidxrebuild` command on the new master node, and release the text index from its unfinished status. At this time, execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

For details about the `--create-temp-file` option, see *Specification format for the adbidxrebuild command* under *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

For details about the unfinished status of text indexes, see 15.10.1 Steps to take when unfinished status is applied to a text index.

(4) Releasing a range index from unfinished status (when the multi-node function is being used)

If abnormal termination of the `adbidxrebuild` command causes a range index to be set to unfinished status, re-execute the `adbidxrebuild` command, and release the range index from its unfinished status.

If the master node's HADB server terminates abnormally and the master node is switched over, execute the `adbidxrebuild` command on the new master node, and release the range index from its unfinished status. At this time, re-execute the `adbidxrebuild` command with the `--create-temp-file` option specified.

For details about the `--create-temp-file` option, see *Specification format for the adbidxrebuild command* under *adbidxrebuild (Rebuild Indexes)* in the manual *HADB Command Reference*.

For details about the unfinished status of range indexes, see 15.11.1 Steps to take when unfinished status is applied to a range index.

16.11.5 Working with multi-chunk tables (when the multi-node function is being used)

This subsection explains operations on multi-chunk tables when the multi-node function is being used.

When the multi-node function is being used, the following operations on multi-chunk tables must be unique to the multi-node function:

- Merging multiple chunks
- Changing a chunk comment
- Change chunk status

For details about operations in relation to base tables when using the multi-node function, see 16.11.2 Base table operations (when the multi-node function is being used).

All other operations are the same as when the multi-node function is not used. For details, see 11.4 Performing operations on multi-chunk tables.

(1) Merging chunks (when the multi-node function is being used)

To merge multiple chunks into a single chunk, execute the `adbmergechunk` command. For details, see 11.4.9 Merging chunks (to reduce the number of chunks).

When the multi-node function is being used, you can execute the `adbmergechunk` command on the master node only. You cannot execute this command on a slave node.

(2) Changing a chunk comment (when the multi-node function is being used)

To set a chunk comment, or to change or delete a comment that was set, execute the `adbchgchunkcomment` command. For details, see 11.4.11 Changing the comment for a chunk.

When the multi-node function is being used, you can execute the `adbchgchunkcomment` command on the master node only. You cannot execute this command on a slave node.

(3) Changing chunk status (when the multi-node function is being used)

To change the status of a chunk, execute the `adbchgchunkstatus` command. For details, see 11.4.12 Changing the chunk status.

When the multi-node function is being used, you can execute the `adbchgchunkstatus` command on the master node only. You cannot execute this command on a slave node.

16.11.6 Operating archivable multi-chunk tables (when the multi-node function is used)

This section explains how to operate archivable multi-chunk tables when using the multi-node function.

If you are using the multi-node function, you need to perform operation specific to the multi-node function for the following operations relating to archivable multi-chunk tables:

- Setting archive directories
- Archiving chunks
- Unarchiving chunks

Operations other than the preceding operations are the same as when not using the multi-node function. For details, see 11.4 Performing operations on multi-chunk tables.

(1) Setting archive directories

Make sure that the archive directory can be referenced using the same absolute path from all nodes.

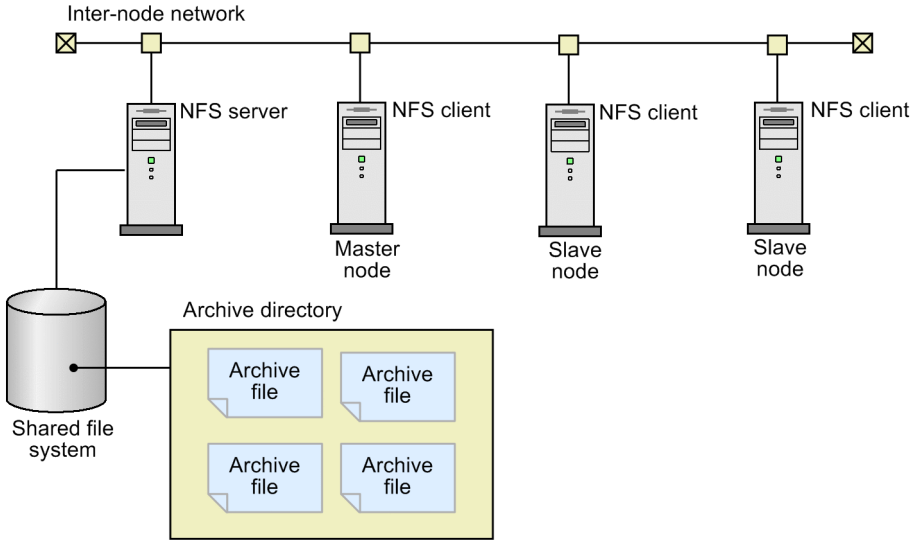
For example, when using the distributed file system NFS, use the following procedure to prepare the archive directory:

Procedure:

1. Prepare the NFS server, and then create an archive directory on the NFS server.
2. Export the archive directory you created in step 1.
   Perform this operation on the NFS server.
3. Mount the directory exported in step 2 so that the paths are the same on all nodes.
   Perform this operation on all nodes.

The following figure shows an example system configuration using NFS:

Figure 16-6: Example of a system configuration that uses NFS



(2) Archiving chunks

If you are using the multi-node function, the only node that can execute the `adbarchivechunk` command for archiving chunks is the master node. Slave nodes cannot execute the `adbarchivechunk` command.

For information about the `adbarchivechunk` command, see *adbarchivechunk (Archive Chunk)* in the manual *HADB Command Reference*.

(3) Unarchiving chunks

If you are using the multi-node function, the only node that can execute the `adbunarchivechunk` command for unarchiving chunks is the master node. Slave nodes cannot execute the `adbunarchivechunk` command.

For information about the `adbunarchivechunk` command, see *adbunarchivechunk (Unarchive Chunk)* in the manual *HADB Command Reference*.

16.11.7 Reorganizing system tables (when the multi-node function is used)

Execute the `adbreorgsystemdata` command for reorganizing system tables on the master node. Slave nodes cannot execute the `adbreorgsystemdata` command.

For details about the procedure for reorganizing system tables, see 11.17 Reorganizing system tables.

16.12 System log operations (when the multi-node function is being used)

When the multi-node function is being used, there is only one system log file for all nodes, and it is controlled by the HADB server on the master node. The HADB server on the master node outputs the operation history information of the HADB servers on all nodes, as well as the database update history information, as system logs to the system log file.

System log files are created in the file system that is mounted to the `/ADBSYS/ADBSLG` directory under the master node's DB directory.

For details about the system log, see (2) System log in 2.11.1 Recovery flow based on a restart.

- **When a master node switchover occurs**

  Carry over the system log managed by the HADB server of the old master node, to the HADB server of the new master node. The HADB server of the new master node uses the carried over system log to perform recovery processing (rollback) on update processing that had been running on the master node.

  If a master node switchover occurs, the file system storing the system log is unmounted from the pre-switchover master node. Then, the file system is mounted to the `/ADBSYS` directory under the new master node's DB directory. HA Monitor performs these unmount and mount processes automatically.

16.13 Statistical information operations (when the multi-node function is being used)

When the multi-node function is being used, statistical information is controlled separately for each node. Therefore, execute the `adbstat` command on the node whose statistical information you want to output.

Note that the system might decide which node executes transactions in some cases. Therefore, when you output connection operation information or SQL statement statistical information, information is output only for the transactions executed on the node where you executed the `adbstat` command.

For details about the nodes where transactions are executed, see 16.5.3 Node that executes transactions.

For details about how to check statistical information, see 10.10 Performing statistical analysis (checking HADB server operation information).

- **When obtaining access path statistical information**

  Access path statistical information is output in the SQL trace information. When the multi-node function is being used, SQL trace information is managed separately for each node. Therefore, when outputting statistical information for nodes, only the information for the SQL statement of the transaction executed on the local node is output. Accordingly, when obtaining statistical information for the access paths of all transactions, make sure that you obtain all nodes' SQL trace information. For details about how to obtain SQL trace information, see 10.11.5 Preparations for outputting SQL trace information.

16.14 Tuning (when the multi-node function is being used)

To tune the HADB servers when the multi-node function is being used, tune each HADB server on each node.

For details about what is involved in tuning, see 13. Tuning.

16.15 Operations when a node failure occurs

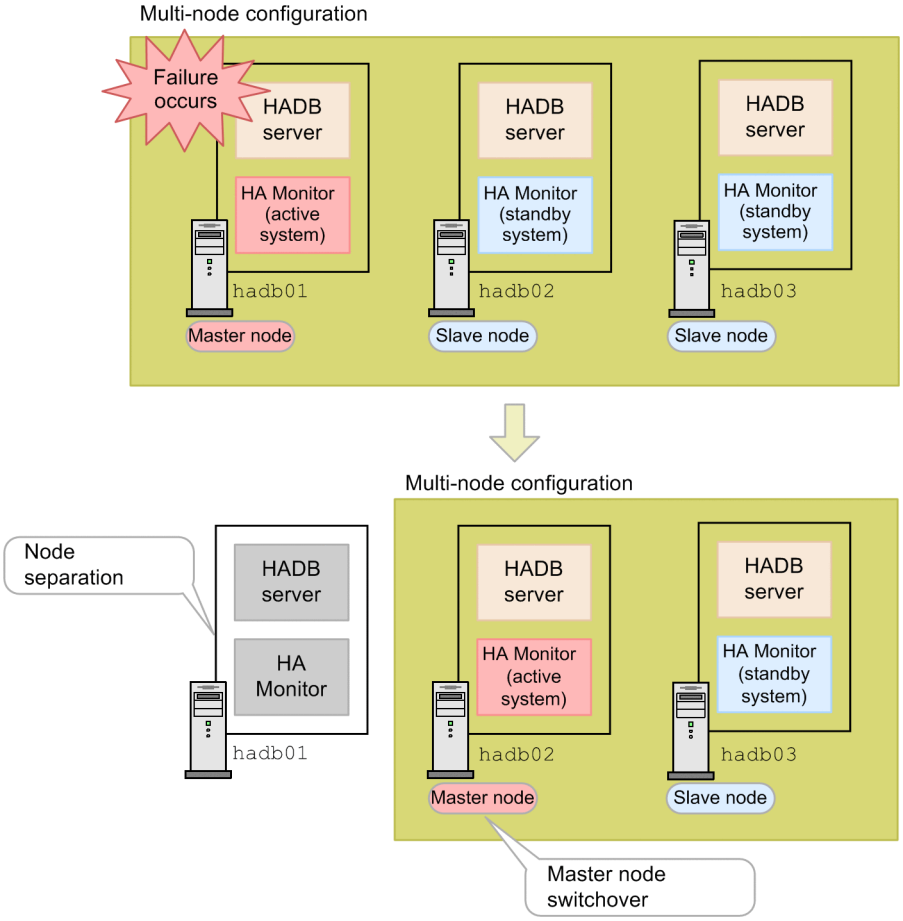
When a node failure occurs, the node in which the failure occurred is automatically separated from the multi-node configuration in order to allow operations to continue by using the remaining nodes. This section explains the operation in the event of a node failure.

16.15.1 When a node failure occurs on the master node

(1) System processing when a node failure occurs on the master node

When a node failure occurs on the master node, one of the slave nodes switches over to the master node, and the master node where the failure occurred is cut off from the multi-node configuration. The following figure shows the system processing when a node failure occurs on the master node:

Figure 16-7: System processing when a node failure occurs on the master node



Explanation

When a node failure occurs on the master node `hadb01`, one of the slave nodes switches over to the master node, and the node where the failure occurred is cut off from the multi-node configuration. Processing then continues on the master node `hadb02` and the slave node `hadb03`.

In HADB, the HA Monitor multi-standby function is used to switch over the master node. The multi-standby function refers to the function for preparing multiple standby systems (slave nodes) for one active system (master node). The order of priority for slave nodes (which slave node becomes the master node) is set by using the `initial` and

`standbypri` operands in the HA Monitor `servers` file. For details, see *Managing servers and hosts when using the multi-standby function* in the manual *HA Monitor for Linux(R) (x86)*.

(2) Steps taken by the HADB administrator when a node failure occurs

To identify the cause of the node failure, take the action as described in 14.1 Error-handling flow. Perform the relevant procedure in the preceding reference, on the node where the node failure occurred.

Alternatively, obtain the following troubleshooting information on the node where the failure occurred:

- HADB server troubleshooting information obtained by using the `adbinfoget` command
- HA Monitor troubleshooting information obtained by using the HA Monitor `monts` command

After identifying the cause of the failure, you can return any nodes that were separated from the multi-node configuration, back to the configuration. For details about the procedure for returning nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

Note that nodes are always returned to a multi-node configuration as slave nodes.

(3) Notes

When there is only one master node in a multi-node configuration, and a node failure occurs on that master node, there is no master node to switch over to, so the multi-node configuration's HADB server terminates abnormally. If this happens, execute the HA Monitor `monshow` command on the master node where the node failure occurred, and check the status of the local host server.

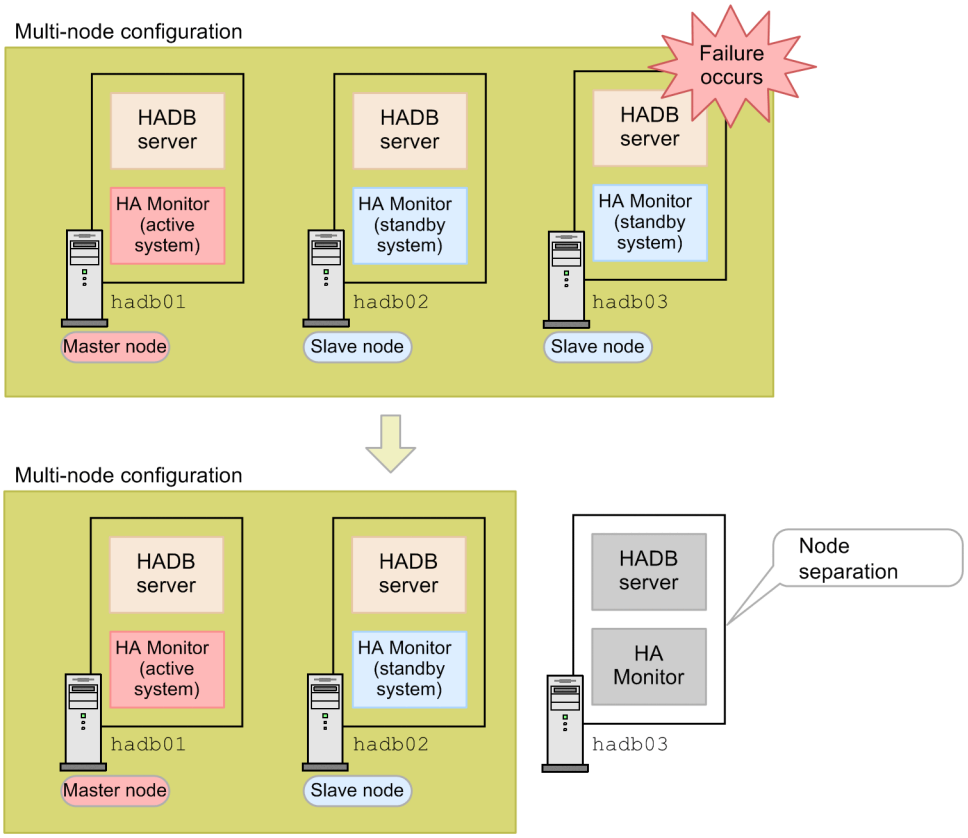
If the local host server is in the ONL status (currently executing a process), execute the HA Monitor `monend` command, and terminate HA Monitor.

16.15.2 When a node failure occurs on a slave node

(1) System processing when a node failure occurs on a slave node

When a node failure occurs on a slave node, that slave nodes is separated from the multi-node configuration. The following figure shows the system processing when a node failure occurs on a slave node:

Figure 16-8: System processing when a node failure occurs on a slave node



Explanation

When a node failure occurs on the slave node `hadb03`, that slave node is separated from the multi-node configuration. Processing then continues on the remaining nodes `hadb01` and `hadb02`.

(2) Steps taken by the HADB administrator when a node failure occurs

To identify the cause of the node failure, take the action as described in 14.1 Error-handling flow. Take the aforementioned actions on the node where the failure occurred.

Also acquire the following types of troubleshooting information from the node where the failure occurred:

- Use the `adbinfoget` command to acquire the HADB server's troubleshooting information.
- Use HA Monitor's `monts` command to acquire HA Monitor's troubleshooting information.

After resolving the cause of the failure, you can return the node that was separated from the multi-node configuration, back to the multi-node configuration. For details about the procedure for returning nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

Note that nodes are always returned to a multi-node configuration as slave nodes.

16.15.3 Returning a node to the multi-node configuration

This section explains the procedure for returning a node that was separated from a multi-node configuration, back to the multi-node configuration.

(1) Notes on returning nodes to a multi-node configuration

1. The values specified in the following server definition operands cannot be changed:

   - `adb_sys_rthd_num`

   The specified value cannot be changed even for operands that need to be specified to the same value on all nodes. For a list of operands for which the same value needs to be specified on all nodes, see 16.3.5 Creating server definitions on all nodes.

   Do not change the specified values even of operands other than the preceding operands, unless absolutely necessary.

2. Nodes cannot be returned to a multi-node configuration if at least one of the following cases applies. Before performing an operation that returns nodes to a multi-node configuration, confirm that none of the following cases applies:

   - The HADB servers that make up a multi-node configuration include ones on which termination processing is in progress.
   - A master node switchover is in progress.
   - There are HADB servers that are being returned to a multi-node configuration.
   - There are application programs or commands that are connected to an HADB server.

(2) Procedure for returning nodes to a multi-node configuration

The procedure for returning nodes to a multi-node configuration is as follows:

Procedure:

1. Execute the HA Monitor `monshow` command on the node subject to the return

   Make sure that monitoring by HA Monitor is stopped.

   > **Important**
   >
   > Perform the following procedure after making sure that monitoring by HA Monitor is stopped. If you accidentally execute the HA Monitor `monbegin` command from step 4 before executing the `adbstart` command from step 3, execute the HA Monitor `monend` command (make sure to stop monitoring by HA Monitor, and any running HADB servers). Afterwards, perform the procedure for returning nodes to a multi-node configuration from the very beginning.

2. Execute the `adbchgsrvmode --maintenance` command on the master node

   Change the HADB server operation mode on the master node to maintenance mode. Alternatively, make sure that there are no ongoing transactions.

3. Execute the `adbstart` command on the node subject to the return

   After the `adbstart` command is executed, the `KFAA91109-I` message is output asking you to execute the HA Monitor `monbegin` command.

4. Execute the HA Monitor `monbegin` command on the node subject to the return

   After confirming that the `KFAA91109-I` message has been output, execute the `monbegin` command from a different terminal (command input screen).

   > **Important**
   >
   > Before the `adbstart` command terminates, execute the `monbegin` command.

When the `adbstart` command terminates normally, returning of the node to the multi-node configuration is complete.

5. Execute the `adbchgsrvmode --normal` command on the master node

   If you changed the master node's HADB server operation mode to maintenance mode in step 2, return the operation mode to normal mode.

6. Execute the `adbls -d node` command on any node

   Refer to the `NODE_TYPE` field in the command execution result, and make sure that there is one master node. If all the nodes are slave nodes, execute the HA Monitor `monact` command on the node you want to make the master node. The node where you executed the `monact` command becomes the master node.

> **Note**
>
> In step 2, if the master node stops at the moment you execute the `monbegin` command before the `KFAA91109-I` message is output, all nodes might become slave nodes.

Note that nodes are always returned to a multi-node configuration as slave nodes. To set the node you returned as the master node, see (3) Procedure for switching over a returned node to the master node.

> **Important**
>
> In environments where synonym searches are used, you need to synchronize the synonym dictionary files after returning the nodes. For details about how to synchronize the synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.

> **Note**
>
> If the audit trail facility is enabled, audit trail output will automatically resume for the node when the process of returning it has completed. If a node is returned to a multi-node configuration where the audit trail facility is disabled, the node will be returned with the audit trail facility disabled even if the facility was enabled when the HADB server of the node was last operational.

(3) Procedure for switching over a returned node to the master node

The following explains the procedure for switching over a returned node to the master node. The procedure differs depending on whether you are using host reset or SCSI reservation for shared disk.

- **Procedure (when using host reset)**

  1. Return the node that was separated from the multi-node configuration (the former master node) back to the configuration.

     Note that nodes are returned to a multi-node configuration as slave nodes.

  2. Execute the HA Monitor `monswap` command on the current master node.

     The current master node's HADB server terminates, and the returned node in step 1 switches over to the master node.

  3. Execute the `adbls -d node` command on any node.

     Refer to the `NODE_TYPE` field in the command execution result, and make sure that the master node has switched over. You cannot perform the subsequent operations until the master node switchover is complete.

  4. Return the node you terminated in step 2, to the multi-node configuration.

For details about the procedure for returning a node to a multi-node configuration, see (2) Procedure for returning nodes to a multi-node configuration.

- **Procedure (when using SCSI reservation for shared disk)**

  1. Return the node that was separated from the multi-node configuration (the former master node) back to the configuration.

     Note that nodes are returned to a multi-node configuration as slave nodes.

  2. Execute the `adbstop --cancel --node` command on the master node.

     The master node terminates normally.

  3. Execute the HA Monitor `monswap` command.

     When you execute the preceding command on the node you terminated in step 2, the node you returned in step 1 becomes to the master node.

  4. Execute the `adbls -d node` command on any node.

     Refer to the `NODE_TYPE` field in the command execution result, and make sure that the master node has switched over. You cannot perform the subsequent operations until the master node switchover is complete.

  5. Return the node you terminated in step 2, to the multi-node configuration.

     For details about the procedure for returning a node to a multi-node configuration, see (2) Procedure for returning nodes to a multi-node configuration.

16.16 Troubleshooting (when the multi-node function is being used)

This section explains troubleshooting when the multi-node function is being used.

16.16.1 Problems related to startup or termination of the HADB servers in the multi-node configuration

This subsection explains the steps to take when a problem related to startup or termination of the HADB servers occurs in the multi-node configuration.

(1) When the HADB servers in the multi-node configuration cannot be started normally

Determine the cause by following the steps indicated in the output message. Then, start the HADB servers in the multi-node configuration normally.

If you cannot determine the cause immediately, remove the failed node from the multi-node configuration, and then normally start the HADB servers. To remove the failed node, delete the specification about it from the `adb_sys_multi_node_info` operand in the server definition of the HADB servers on all nodes.

If only one node remains on which a failure did not occur, change the `adb_multi_node_info` operand in the server definition for the HADB server on that node into a comment. Then, start the HADB server as a system that does not use the multi-node function. In this case, make sure that the system directory is mounted to `$DBDIR/ADBSYS`. If the directory is not mounted, execute the OS's `mount` command to mount the system directory to `$DBDIR/ADBSYS`.

> **Note**
>
> Adding a hash mark (#) to the beginning of the operand specification line changes the specification on the entire line into a comment.

In addition, if the node in which the failure occurred is the master node, change the master node by using either of the following methods:

- Change the master node by changing the HA Monitor settings.
- When starting the HADB server in a multi-node configuration, execute the `monact` command to change one of the slave nodes into the master node.

If there is a substitute machine, set up the substitute machine's environment so that it is the same as the machine of the node in which the failure occurred, and then start the HADB servers in the multi-node configuration.

▪ If the `KFAA41205-E` message is output

If the `KFAA41205-E` message is output containing the following information, opening of the DB area file fails:

- The name of the symbolic link to the DB area file is displayed as the file name
- `open` is displayed as the name of the system call that resulted in an error
- `ENOENT` is displayed as the error number

In this case, when you added a node, the `adbinit` command might have been executed without applying the configuration change information of the DB area to the initialization options. Therefore, you need to re-add the node.

For details about how to add nodes, see 16.17.1 Adding nodes. Pay particular attention to the explanation in 16.17.2 Notes on executing the adbinit command when adding nodes.

> **Important**
>
> When the `adbinit` command is executed on the node to be added, the initialization options used on other nodes might not be usable as is. If you have changed the DB area configuration, you need to apply those changes to the initialization options.

(2) When the HADB servers in the multi-node configuration cannot be restarted

Determine the cause by following the steps indicated in the output message. Then, restart the HADB servers in the multi-node configuration.

If you cannot determine the cause immediately, remove the failed node from the multi-node configuration and then restart the HADB servers. To remove the failed node from the multi-node configuration, delete the specification about it from the `adb_sys_multi_node_info` operand in the server definition of the HADB servers on all nodes.

In addition, if the node in which the failure occurred is the master node, change the master node by using either of the following methods:

- Change the master node by changing the HA Monitor settings
- When starting the HADB server in a multi-node configuration, execute the `monact` command to change one of the slave nodes into the master node.

▪ If the `KFAA41205-E` message is output

For details about the action to take when the `KFAA41205-E` message is output, see **▪ If the `KFAA41205-E` message is output** in (1) When the HADB servers in the multi-node configuration cannot be started normally.

(3) When an HADB server in the multi-node configuration cannot be terminated

While an application or command is connected to an HADB server in the multi-node configuration, that HADB server cannot be terminated. Wait for the application or command to finish. If there is a need to terminate the HADB servers right away, forcibly terminate the HADB servers in the multi-node configuration. For details about how to forcibly terminate the HADB server in a multi-node configuration, see (b) Forcibly terminating HADB servers in the multi-node configuration in (1) Termination procedures for HADB servers in a multi-node configuration under 16.4.2 Terminating HADB servers in the multi-node configuration.

If the HADB servers in a multi-node configuration cannot be terminated even when there is no connected application program or command, perform the following.

Procedure:

1. Execute the `adbstop --force` command on the master node.

   When you forcibly terminate the master node's HADB server, the master node is switched over.

2. Terminate the HADB servers in the multi-node configuration normally.

   For details about how to terminate the HADB server normally in a multi-node configuration, see (a) Normally terminating HADB servers in the multi-node configuration in (1) Termination procedures for HADB servers in a multi-node configuration under 16.4.2 Terminating HADB servers in the multi-node configuration.

If the HADB servers in a multi-node configuration cannot be terminated even with the above procedure, execute the `adbstop --force` command on all nodes to forcibly terminate the HADB servers in the multi-node configuration.

16.16.2 Application-related problems (when the multi-node function is being used)

If an application-related problem occurs, first take the appropriate corrective action as explained in 15.1 Application-related problems. If this does not resolve the problem, read the information provided in this subsection.

(1) Steps to take when an application cannot be executed

The following table describes the possible reasons why an application cannot be executed, and the steps that can be taken.

Table 16-10: Possible reasons when an application cannot be executed and the steps that can be taken

No.	Possible cause	Steps to take
1	The connection-destination host name might not correspond with the alias IP address used for HADB client-server communication.	Check whether a host name is specified that corresponds to the alias IP address used for communication between HADB clients and HADB servers. Check the locations where the host name of the connection destination for the application is specified, such as the client definition, and correct the host name if necessary.
2	The number of concurrent application executions might have exceeded the maximum number of concurrent connections allowed for the HADB servers in the multi-node configuration.	**Procedure:** 1. Terminate the HADB servers in the multi-node configuration. 2. Increase the value specified for the `adb_sys_max_users` operand in the server definition. 3. Start the HADB servers in the multi-node configuration.

16.16.3 Command related problems (when the multi-node function is being used)

If a command-related problem occurs, first take the appropriate corrective action as explained in 15.2 Command-related problems. If this does not resolve the problem, read the information provided in this subsection.

(1) Steps to take when a command cannot be executed (when the message KFAA31733-E is output)

If a failure occurs on the master node, recovery processing (rollback) performed on update processing that had been running on the master node, is executed on the slave node that becomes the new master node. If a command accesses any pages undergoing this recovery processing, the command results in an error. At this time, the `KFAA31733-E` message is output. In this case, follow the steps given in the message.

(2) Steps to take when a command cannot be executed (when the message KFAA50036-E is output)

An attempt was made to execute a command that cannot be executed on a slave node. Execute the command on the master node.

16.16.4 Problems related to DB areas (when the multi-node function is being used)

This subsection explains the steps to take when a problem related to a data DB area occurs.

(1) Steps to take when a data DB area becomes full

Expand the data DB area. For details about how to expand the target DB area, see (3) Expanding a data DB area (adding a data DB area file) in 16.10.3 Adding, deleting, or expanding data DB areas (when the multi-node function is being used).

16.16.5 Problems related to files in the DB directory (when the multi-node function is being used)

This subsection explains the steps to take when a problem occurs in a file under the DB directory.

(1) Steps to take when a failure occurs in a system log file

Take the steps indicated in the message that is output.

Also, check whether the file system for the system directory is mounted in `$DBDIR/ADBSYS`.

In addition, see 15.5.2 Steps to take when a problem occurs in a system log file.

(2) Steps to take when a free space shortage in a system log file causes the HADB server to terminate abnormally

Take the steps indicated in the message that is output.

Also see 15.5.2 Steps to take when a problem occurs in a system log file.

16.16.6 Problems related to files in the server directory (during installation) (when the multi-node function is being used)

Follow the explanation in the action to take section for the message that is output.

16.16.7 Problems related to files in the server directory (during operation) (when the multi-node function is being used)

This subsection explains the steps to take when a problem occurs in a file under the server directory.

(1) Steps to take when a failure occurs in the server directory

For details about the action to take when a failure occurs in relation to the server directory, see 15.4.2 Recovering the server directory.

(2) Steps to take when a failure occurs in the server definition file

For details about the steps to take when a failure occurs in the server definition file, follow the explanation in the action to take section for the message that is output.

(3) Steps to take when a failure occurs in a message log file

For details about the steps to take when a failure occurs in message log file, follow the explanation in the action to take section for the message that is output.

16.16.8 Problems related to files in the client directory (when the multi-node function is being used)

For details about how to recover the client directory, see 15.4.3 Recovering the client directory.

16.16.9 OS-related problems (when the multi-node function is being used)

If a slave node OS terminates abnormally, that slave node is separated from the multi-node configuration.

If the OS terminates abnormally on the master node, the master node is switched over. The master node whose OS terminated abnormally is separated from the multi-node configuration.

After the operating system is rebooted, eliminate the cause of the problem.

16.16.10 Hardware-related problems (when the multi-node function is being used)

This subsection explains the steps to take when a hardware-related problem occurs.

(1) Steps to take when a disk failure occurs

For details about the action to take when a disk failure occurs, see 15.4 Disk-related problems.

(2) Steps to take when the disk becomes full

(a) When the space shortage is caused by the DB area files

When the disk become full because of an increase in the size of the DB area files, take the necessary steps based on the explanations in the following subsections:

- 15.3.1 When a free space shortage is caused by an increase in the size of the DB area files
- 15.3.2 When a free space shortage is caused by failed DB area automatic extension

(b) When the space shortage is caused by a factor other than the DB area files

- If the problem is caused by the system log file or status file

  Take action as explained in (1) When the space shortage is caused by entity files under the DB directory under 15.3.3 When a free space shortage is caused by an increase in the size of files other than the DB area files.
- If the problem is caused by a file other than the system log file or status file

  Take action as explained in (2) When the space shortage is caused by symbolically linked files under 15.3.3 When a free space shortage is caused by an increase in the size of files other than the DB area files.

(3) Steps to take when a communication failure, CPU failure, or power supply failure occurs

This subsection explains the steps to take when a communication failure, CPU failure, or power supply failure occurs.

(a) When a communication failure occurs between the HADB client and the HADB server on the master node

In this case, the applicable transaction returns an error indicating that a communication failure has occurred. If this happens, perform the following procedure.

Procedure:

1. Separate the master node where the communication failure occurred, from the multi-node configuration

   Switch the master node by executing a command. The master node where the communication failure occurred, is separated from the multi-node configuration. For details about how to switch over the master node by using a command, see 16.7 Switching over the master node by using a command.
2. Investigate and address the cause of the communication failure
3. Return the node you separated in step 1, back to the multi-node configuration

   For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

(b) When a communication failure occurs between the HADB client and the HADB server on a slave node

In this case, the applicable transaction returns an error indicating that a communication failure has occurred.

Investigate the cause of the communication failure. If you cannot take a corrective action immediately, terminate the HADB server on the slave node where the failure occurred.

(c) When a communication failure occurs between nodes

The following explains the action to take when a communication failure occurs between nodes. The action to take differs depending on whether you are using host reset or SCSI reservation for shared disk.

- **When using host reset**

  If a communication error occurs between nodes, the HA Monitor detects a node failure. At this time, HA Monitor decides which nodes to separate from the multi-node configuration. If the master node is one of the nodes that are separated from the multi-node configuration, the master node is switched over.

  The server machines of nodes that are separated from the multi-node configuration are stopped by HA Monitor (the power for the server machines is shut off). If a server machine does not stop, execute the HA Monitor `monswap` command, and then stop the server machine manually.

  Then, restart the server machine, and investigate and address the cause of the communication failure.

  After taking the aforementioned steps, return the relevant nodes to the multi-node configuration as necessary. For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

- **When using SCSI reservation for shared disk**

  If a communication error occurs between nodes, the HA Monitor detects a node failure. At this time, HA Monitor decides which nodes to separate from the multi-node configuration. If the master node is one of the nodes that are separated from the multi-node configuration, the master node is switched over. The server machines of nodes that are separated from the multi-node configuration are not stopped.

  In this case, investigate and address the cause of the communication failure. After taking the aforementioned steps, return the relevant nodes to the multi-node configuration as necessary. For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

  If the failure is on the monitoring path of HA Monitor, you need to recover the path by using the `monlink` command of HA Monitor. For details, see *Hot standby operation when a failure occurs* in the manual *HA Monitor for Linux(R) (x86)*.

(d) When a CPU failure occurs

If a CPU failure occurs on a slave node, that slave node is separated from the multi-node configuration.

If a CPU failure occurs on the master node, the master node is switched over. The master node where the CPU failure occurred, is separated from the multi-node configuration.

After restarting the OS, investigate and address the cause of the problem. After addressing the problem, return the separated nodes back to the multi-node configuration as necessary. For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

(e) When a power supply failure occurs

If a power failure occurs on a slave node, that slave node is separated from the multi-node configuration.

If a power failure occurs on the master node, the master node is switched over. The master node where the power failure occurred, is separated from the multi-node configuration.

After restarting the OS, investigate and address the cause of the problem. After addressing the problem, return the separated nodes back to the multi-node configuration as necessary. For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

16.16.11 Problems related to HA Monitor (when the multi-node function is being used)

If a problem occurs in HA Monitor, take the corrective steps based on the explanations in the manual *HA Monitor for Linux(R) (x86)*.

16.16.12 Problems related to SQL statements that create work tables (when the multi-node function is being used)

If an SQL statement being executed when the HADB server terminates abnormally creates a work table, make sure that the disk to be used by work table DB area files is not shared and used by multiple nodes. If the disk to be used for the work table DB area files is shared and used by multiple nodes, operations are not guaranteed. In this case, incorrect retrieval results might be displayed, or the HADB server might terminate abnormally.

If the disk to be used for work table DB area files is shared and used by multiple nodes, perform the following.

Procedure:

1. Terminate the HADB servers in the multi-node configuration.
2. Prepare a disk for work table DB area files for each node.
3. For each node, specify the path to the disk prepared in step 2 above for the server definition `adb_blk_path_wrk` operand.
4. Start the HADB servers in the multi-node configuration again.

> **Note**
>
> You can determine whether the executed SQL statement creates a work table by referring to *Considerations when executing an SQL statement that creates work tables* in the *HADB Application Development Guide*.

16.16.13 Problems related to synonym dictionary files (when the multi-node function is being used)

This subsection explains the steps to take when a problem related to a synonym dictionary file occurs.

(1) Steps to take when a problem occurs in relation to a synonym dictionary file when the multi-node function is being used

Take the following action when a problem occurs in relation to a synonym dictionary file when the multi-node function is being used.

> **Note**
>
> The synonym dictionary files used in a multi-node configuration are stored in the directory specified for the `adb_syndict_node_storage_path` operand in the server definition.

Action 1

When an error occurs in relation to a synonym dictionary file, the following messages are output to the message log file on the slave node where the error occurred:

- KFAA30959-E
- KFAA34008-E
- KFAA51537-W

Take the action as explained in the action to take section of the message. If taking the action explained in the action to take section of the message does not resolve the problem or you are unable to take the action the message suggests, try action 2.

Action 2

Delete the synonym dictionary file that caused the error. Then, execute the adbsyndict command to synchronize the synonym dictionary files. For details about how to synchronize the synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.

(2) Steps to take when there is insufficient free space for the multi-node synonym dictionary storage directory

If the adbsyndict command terminates abnormally due to there being insufficient free space for the multi-node synonym dictionary storage directory, messages are output as follows:

- KFAA51519-E is output on the master node.
- KFAA51514-E is output on the node where the lack of free space occurred.

If the operation target information in the message KFAA51514-E is synonym-dictionary-file and the error number is 28 (ENOSPC), there might be insufficient free space for the storage directory for synonym dictionary files. In this case, first try Action 1. If you cannot perform Action 1 or there is still not enough free space after Action 1 is performed, perform Action 2.

> **Note**
>
> The multi-node synonym dictionary storage directory is the directory specified for the adb_syndict_node_storage_path operand in the server definition.

Action 1

If multiple synonym dictionaries are updated at once when the adbsyndict command is executed, try updating the synonym dictionaries one at a time. Because less free space is required when updating fewer synonym dictionaries, you might be able to execute the adbsyndict command successfully.

Action 2

Create the multi-node synonym dictionary storage directory again on another disk with more free space. The following shows the procedure.

Procedure:

1. Create another multi-node synonym dictionary storage directory on a disk with more free space. For details about how to create this directory, see (6) Creating the multi-node synonym dictionary storage directory in 16.24.1 Preparing for synonym search operations.
2. Perform a normal termination of the node where the lack of free space occurred. For details about how to terminate a node normally, see (2) Terminating an HADB server on a specific node in 16.4.2 Terminating HADB servers in the multi-node configuration.

3. After the node has terminated normally, change the value specified for the `adb_syndict_node_storage_path` operand in the server definition. Specify the directory you created in 1.
4. Start the node normally and return it to the multi-node configuration. For details about the procedure for returning a node to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.
5. Execute the `adbsyndict` command to synchronize the synonym dictionary files. For details about how to synchronize the synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.
6. Delete the old multi-node synonym dictionary storage directory.

16.17 Adding or deleting nodes

This section explains how to add nodes by adding a server machine. This section also explains how to delete nodes that are no longer necessary.

16.17.1 Adding nodes

This procedure for adding nodes by adding a server machine is as follows:

Procedure:

1. Add the server machine, install the HADB server, and configure the environment settings

   For details about how to install the HADB server and set up the environment, see 16.3.2 Installing HADB server and setting up an environment.

   > **Important**
   >
   > Before executing the `adbinit` command, read the notes provided in 16.17.2 Notes on executing the adbinit command when adding nodes.

2. Normally terminate the multi-node configuration HADB server

   For details about the procedure for terminating the HADB server normally in a multi-node configuration, see (a) Normally terminating HADB servers in the multi-node configuration in (1) Termination procedures for HADB servers in a multi-node configuration under 16.4.2 Terminating HADB servers in the multi-node configuration.

3. Correct the HA Monitor definition

   Specify information about the added node in the HA Monitor definition on all nodes.

4. Correct the server definition

   Add information about the node to be added, to the HADB server definition operand `adb_sys_multi_node_info` on all nodes.

5. Start the multi-node configuration HADB server

   For details about the procedure for starting the HADB server in a multi-node configuration, see 16.4.1 Starting the HADB servers in the multi-node configuration.

> **Important**
>
> In environments where synonym searches are used, you need to synchronize the synonym dictionary files after adding a node. For details about how to synchronize the synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.

16.17.2 Notes on executing the adbinit command when adding nodes

When adding a node, you execute the `adbinit` command on the node you are adding, but you might not be able to use the initialization options used on other nodes as is. If you have changed the configuration of the DB area, you need to apply those changes to the specifications in the initialization options, and execute the `adbinit` command. Therefore, you need to apply the information that differs due to the changes to the configuration of the DB area, to the initialization

options used on the slave nodes. If you cannot do that, create initialization options according to the procedure explained in this section.

The overall flow of the procedure explained in this section is as follows: First execute the command to extract the current DB area information to a file. Edit that file to create the initialization options.

Important

- In this explanation, `$DBDIR` means the DB directory. When actually executing the command, specify the DB directory path.
- Execute the following procedure on the master node.

(1) Creating the base for the initialization options

```
echo "set adb_init_mst_blk_path = \"$(ls -l $DBDIR/ADBMST \
| sed -e 's/.*ADBMST ->/ADBMST ->/g' | awk '{print $3}')\"" >adbinit.opt
```

Execute the preceding command, and create the base of the initialization option file (`adbinit.opt` file). In this example, the `adbinit.opt` file is created in the current directory.

When you execute the preceding command, the following `adbinit.opt` file is created:

▪ Content of the `adbinit.opt` file at this point

```
set adb_init_mst_blk_path = "/dev/disk/by-id/wwn-0x60....93bc"
```

(2) Add information such as the adbinitdbarea operand to the adbinit.opt file

```
adbdbstatus -d used | grep Used | awk -F',' '{print $3, $9}' | uniq \
| awk '{print $1, $2}' \
| sed -e 's/DBarea_name DBarea_filename/set adb_init_dbarea_initialize = N/g' \
      -e 's/\"ADBDIC\"/set adb_init_dic_blk_path =/g' \
      -e 's/\"ADBSTBL\"/set adb_init_stbl_blk_path =/g' \
| sed -e 's/^\"\(.*\)\" \"\(.*\)\"/adbinitdbarea -n \\\"\1\\\" -i 0K,x -v \2,\\/g' \
>>adbinit.opt
```

When you execute the preceding `adbdbstatus` command, information such as the `adbinitdbarea` operand is added to the `adbinit.opt` file you created in (1). When you execute the preceding command, the `adbinit.opt` file is created in a somewhat organized state, which can save trouble in editing the `adbinit.opt` file from (3) onward.

▪ Content of the `adbinit.opt` file at this point

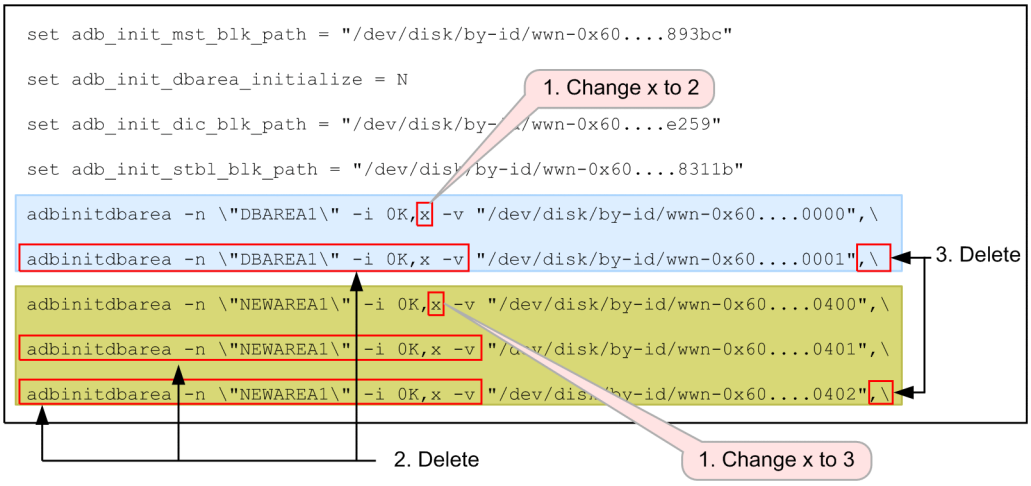
```
set adb_init_mst_blk_path = "/dev/disk/by-id/wwn-0x60....93bc"
set adb_init_dbarea_initialize = N                                          ...1
set adb_init_dic_blk_path = "/dev/disk/by-id/wwn-0x60....e259"              ...2
set adb_init_stbl_blk_path = "/dev/disk/by-id/wwn-0x60....311b"             ...3
adbinitdbarea -n \"DBAREA1\" -i 0K,x -v "/dev/disk/by-id/wwn-0x60....0000",\   ...4
adbinitdbarea -n \"DBAREA1\" -i 0K,x -v "/dev/disk/by-id/wwn-0x60....0001",\   ...5
adbinitdbarea -n \"NEWAREA1\" -i 0K,x -v "/dev/disk/by-id/wwn-0x60....0400",\  ...6
adbinitdbarea -n \"NEWAREA1\" -i 0K,x -v "/dev/disk/by-id/wwn-0x60....0401",\  ...7
adbinitdbarea -n \"NEWAREA1\" -i 0K,x -v "/dev/disk/by-id/wwn-0x60....0402",\  ...8
```

The preceding operands 1-8 are added.

(3) Correcting the content of the adbinit.opt file

Open the `adbinit.opt` file in a text editor, and correct the specifications in the initialization options. For information about specifying the initialization options, see *Format of initialization options* in *Specification format for the adbinit command* in *adbinit (Initialize the Database)* in the manual *HADB Command Reference*.

In this example, you need to correct the specified options for the `adbinitdbarea` operand.



[Explanation]

Specify the `adbinitdbarea` operand as follows:

- If any rows contain duplicate DB area names in the `-n` option of the `adbinitdbarea` operand
  Perform steps 1-3 in the preceding figure.
- If no rows contain duplicate DB area names in the `-n` option of the `adbinitdbarea` operand
  Perform steps 1 and 3 in the preceding figure.

The numbers in the following explanation correspond to the numbers in the preceding figure:

1. Change the specified number of files in the `-i` option, from `x` to the number of duplicate rows. In this example, the value is changed to `2` and `3`.
2. Delete the enclosed specifications.
3. Delete the final `",\"` at the end of each duplicate row.

▪ Content of the `adbinit.opt` file after correction

```
set adb_init_mst_blk_path = "/dev/disk/by-id/wwn-0x60....93bc"
set adb_init_dbarea_initialize = N
set adb_init_dic_blk_path = "/dev/disk/by-id/wwn-0x60....e259"
set adb_init_stbl_blk_path = "/dev/disk/by-id/wwn-0x60....311b"
adbinitdbarea -n \"DBAREA1\" -i 0K,2 -v "/dev/disk/by-id/wwn-0x60....0000",\
                                       "/dev/disk/by-id/wwn-0x60....0001"
adbinitdbarea -n \"NEWAREA1\" -i 0K,3 -v "/dev/disk/by-id/wwn-0x60....0400",\
                                         "/dev/disk/by-id/wwn-0x60....0401",\
                                         "/dev/disk/by-id/wwn-0x60....0402"
```

The preceding enclosed areas are corrected.

(4) Checking the files immediately under the DB directory

If you are storing any symbolic links to a DB area file in a subdirectory of the DB directory, you need to add a specification of the `-f` option to the `adbinitdbarea` operand that specifies that DB area's information. In this case, perform the following procedure:

```
ls $DBDIR
```

▪ Example display of the execution results

```
ADBDIC   ADBMST   ADBSTBL   ADBSYS   ADBWORK   ADBWRK   AREAS   DBAREA1   DBAREA1.00001   SPOO
L
```

Use the OS `ls` command to display the list of files immediately under the DB directory. Check whether there are any files with the same name as the DB area.

In the preceding example, there is a file called `DBAREA1`, but not `NEWAREA1`. Therefore, you need to add the `-f` option to the `adbinitdbarea` operand that specifies `NEWAREA1`.

(5) Investigating the directory storing NEWAREA1

```
find $DBDIR -name NEWAREA1 2>/dev/null
```

Execute the preceding command to investigate the storage directory in the DB directory of the DB area file `NEWAREA1`.

▪ Example display of the execution results

```
$DBDIR/AREAS/NEW/NEWAREA1       ...1
$DBDIR/AREAS/OLD/NEWAREA1       ...2
```

In the following explanation, 1 denotes file 1, and 2 denotes file 2.

If multiple files with the same name but different paths are displayed as in the preceding example, you need to investigate the link destinations by using the OS `ls` command.

(6) Checking the link destination for file 1

```
ls -l $DBDIR/AREAS/NEW/NEWAREA1* | sed -e 's/.* -> //g'
```

Execute the OS `ls -l` command to check the link destination for file 1.

▪ Example display of the execution results

```
/dev/disk/by-id/wwn-0x60....0400
/dev/disk/by-id/wwn-0x60....0401
/dev/disk/by-id/wwn-0x60....0402
```

The preceding three links' destinations match the content of the `adbinit.opt` file.

▪ Content of the `adbinit.opt` file

```
adbinitdbarea -n \"NEWAREA1\" -i 0K,3 -v "/dev/disk/by-id/wwn-0x60....0400",\
                                        "/dev/disk/by-id/wwn-0x60....0401",\
                                        "/dev/disk/by-id/wwn-0x60....0402"
```

(7) Checking the link destination for file 2

```
ls -l $DBDIR/AREAS/OLD/NEWAREA1* | sed -e 's/.* -> //g'
```

Execute the OS `ls -l` command to check the link destination for file 2.

▪ Example display of the execution results

```
/dev/disk/by-id/wwn-0x60....a000
```

The preceding link's destination does not match the content of the `adbinit.opt` file.

(8) Correcting the specifications in the initialization options

Based on the results of checking (6) and (7), we can see that file 1 corresponds to the adbinit.opt file.

Among the path names of file 1, add the following underlined part to the specification in the `-f` option:

```
$DBDIR/AREAS/NEW/NEWAREA1
```

In the preceding example, the `-f` option is added as follows:

```
-f \"AREAS/NEW/\"
```

▪ Content of the `adbinit.opt` file after correction

```
set adb_init_mst_blk_path = "/dev/disk/by-id/wwn-0x60....93bc"
set adb_init_dbarea_initialize = N
set adb_init_dic_blk_path = "/dev/disk/by-id/wwn-0x60....e259"
set adb_init_stbl_blk_path = "/dev/disk/by-id/wwn-0x60....311b"
adbinitdbarea -n \"DBAREA1\" -i 0K,2 -v "/dev/disk/by-id/wwn-0x60....0000",\
                                       "/dev/disk/by-id/wwn-0x60....0001"
adbinitdbarea -n \"NEWAREA1\" -f \"AREAS/NEW/\" -i 0K,3\
              -v "/dev/disk/by-id/wwn-0x60....0400",\
                 "/dev/disk/by-id/wwn-0x60....0401",\
                 "/dev/disk/by-id/wwn-0x60....0402"
```

Add the underlined part in the preceding commands.

(9) Adding operands relating to the work table DB area

Add the following operands as necessary:

- `adb_init_wrk_page_size`
- `adb_init_wrk_blk_path`

In the `adb_init_wrk_blk_path` operand, specify the block special file name of the work table DB area file to be assigned to each node. For details, see (4) Creating a DB directory on each slave node in 16.3.8 Creating a database.

With the preceding steps, correction of the initialization options is now complete. When adding a node, use the initialization options you created here to execute the `adbinit` command.

16.17.3 Deleting nodes

The following procedure shows how to delete nodes that are no longer necessary:

Procedure:

1. Normally terminate the multi-node configuration HADB server

   For details about the procedure for terminating the HADB server normally in a multi-node configuration, see (a) Normally terminating HADB servers in the multi-node configuration in (1) Termination procedures for HADB servers in a multi-node configuration under 16.4.2 Terminating HADB servers in the multi-node configuration.

2. Correct the server definition

   Delete the information for the node you are deleting from the `adb_sys_multi_node_info` operands in the server definitions of the HADB servers on all nodes.

3. Start the multi-node configuration HADB server

   For details about the procedure for starting the HADB server in a multi-node configuration, see 16.4.1 Starting the HADB servers in the multi-node configuration.

16.18 Changing the host name or IP address of the server machine's OS (when the multi-node function is being used)

This section explains how to change the host name or IP address of the OS of a server machine on which the HADB server is installed in a case where the multi-node function is used.

■ **Tasks to be performed on the HADB servers in a multi-node configuration**

The following explains the tasks to be performed on the HADB servers in a multi-node configuration to change the host name or IP address of the server machine's OS.

1. **Normally terminate the HADB servers in a multi-node configuration.**

   Normally terminate the HADB server on all nodes. For details about the termination procedure, see (a) Normally terminating HADB servers in the multi-node configuration in (1) Termination procedures for HADB servers in a multi-node configuration under 16.4.2 Terminating HADB servers in the multi-node configuration.

   Also, normally stop HA Monitor on all nodes by using the `monstop` command.

2. **Change the HA Monitor settings on all nodes.**

   Change the HA Monitor settings on all nodes. For details, see 16.3.4 Setting up an HA Monitor environment.

   - To change the host name of the LAN to be used as the monitoring path of HA Monitor

     Change the settings in the HA Monitor environment settings file (`sysdef` file).

   - To change the alias IP address

     Change the settings in the LAN status settings file for connection (*`server-identification-name`*`.up` file) and the LAN status settings file for disconnection (*`server-identification-name`*`.down` file).

3. **Change the server definition on all nodes.**

   To change the IP address for communication between HADB servers, change the value of the `adb_sys_multi_node_info` operand in the server definition on all nodes. For details, see 16.3.5 Creating server definitions on all nodes.

4. **Start the HADB servers in the multi-node configuration.**

   Use the `monstart` command to start HA Monitor on all nodes. After that, start the HADB server on all nodes. For details about the start procedure, see 16.4.1 Starting the HADB servers in the multi-node configuration.

■ **Tasks to be performed on the HADB client**

The following explains the tasks to be performed on the HADB client to change the host name or IP address of the server machine's OS. Perform the following tasks when changing the alias IP address used for communication between the HADB client and the HADB server.

1. **Terminate the application programs.**

   Terminate all application programs that connect to the HADB servers in a multi-node configuration.

2. **Change the alias IP address specified in the client definition.**

   To change the alias IP address that is used for communication between the HADB client and HADB server, change the value of the `adb_clt_rpc_srv_host` operand in the client definition. For details, see 16.3.6 Creating client definitions.

   For details about the `adb_clt_rpc_srv_host` operand in the client definition, see *Contents of operands in the client definition* under *Designing Client Definitions* in the *HADB Application Development Guide*.

   - If the JDBC driver is used

     Change the host name or IP address of the HADB server specified with the property that corresponds to the `adb_clt_rpc_srv_host` operand in the client definition. The JDBC driver can use a system property,

user property, or connection URL property. For details, see *Setting Up an Environment for the JDBC Driver* in the *HADB Application Development Guide*.

- If the ODBC driver or a CLI function is used

  Change the host name or IP address of the HADB server specified for the `adb_clt_rpc_srv_host` operand in the client definition. For details, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

3. **Start the application programs.**

   Start all application programs that connect to the HADB servers in a multi-node configuration.

Important

If the multi-node function is not used, see 8.11 Changing the host name or IP address of the server machine's OS.

16.19 Migrating to a system that uses the multi-node function

This section explains how to migrate to a system that uses the multi-node function.

Important

When performing data migration, as explained in the subsequent procedure, always back up the server directory and the DB directory. If you become unable to start the HADB servers due to an error in the procedure, you need to recover the server directory and the DB directory using the back-ups.

16.19.1 Terminating the HADB server normally

Before migrating to a system that uses the multi-node function, terminate the HADB server normally.

If the HADB server has already terminated, check the following two points to make sure that it terminated normally:

1. Whether `STOP` is displayed under the output item `STATUS` of the `adbls -d srv` command
2. Whether the termination mode in message `KFAA91154-I`, output in the message log file or syslog, is `normally` (normal termination)

If the HADB server did not terminate normally, start it by executing the `adbstart` command, and then terminate it normally by executing the `adbstop` command.

16.19.2 Backing up the server directory

If you accidentally migrate the system to one that uses the multi-node function while the HADB server is in abnormally terminated status, you will not be able to start the HADB server. Therefore, first back up the files under the server directory.

Delete the backup that was made after you have confirmed that the migrated system is operating correctly.

16.19.3 Backing up the DB directory

Before migrating to a system that uses the multi-node function, you must acquire a full backup of the DB directory. Delete this full backup after confirming that the migrated system is operating correctly.

For details about how to acquire a full backup, see 10.3.1 Backup acquisition method.

16.19.4 Upgrading the HADB server version

If you want to use the multi-node function, and the version of the HADB server to be migrated is 02-02 or earlier (which does not support the multi-node function), you must upgrade the HADB server version.

Be sure to upgrade the HADB server version before migrating to a system that uses the multi-node function. If you accidentally migrate to a system that uses the multi-node function by swapping the HADB server library without upgrading the HADB server version, the HADB server will not start.

For details about upgrading the HADB server version, see 8.6 Upgrading the HADB server version.

16.19.5 Installing HA Monitor and the HADB server

Install HA Monitor and the HADB server on the server machines of all nodes. For details about installing HA Monitor, see *System Configuration* in the manual *HA Monitor for Linux(R) (x86)*. For details about installing the HADB server, see 8.2 Installing the HADB server.

16.19.6 Setting up the nodes

Perform the following on all nodes:

- Change the environment variable specification.
  For details, see 16.3.2 Installing HADB server and setting up an environment.
- Set up the HA Monitor environment.
  For details, see 16.3.4 Setting up an HA Monitor environment.
- Create a server definition.
  For details, see 16.3.5 Creating server definitions on all nodes.
- Create a client definition.
  For details, see 16.3.6 Creating client definitions.

16.19.7 Preparing the DB directory

Prepare the DB directory by taking the steps described in the procedure below.

On the HADB server before migration, if regular files are being used for DB area files, other than work table DB area files, you need to change these files to block special files. For details about how to make these changes, see 16.10.4 Changing the storage location of DB areas.

1. Prepare a file system for the system directory.
   Prepare a file system to be used as the system directory. For details about the file system to be used as the system directory, see (2) File systems inherited at node switchover in 16.2.4 Storage configuration.
2. Prepare the file system that stores synonym dictionary files.
   For details about the file system that stores synonym dictionary files, see (2) File systems inherited at node switchover in 16.2.4 Storage configuration.
   This task is only required in environments that use synonym searches.
3. Prepare a disk for the work table DB area to be used by the added nodes.
   For each node, prepare a disk to be used as the work table DB area. For details about the disk to be used as the work table DB area, see (3) Disks for DB area files in 16.2.4 Storage configuration.
4. Execute the `adbinit` command on the added nodes.

Create the DB directory by executing the `adbinit` command on the added nodes.

Note the following points when doing so:

- Specify `N` for the `adb_init_dbarea_initialize` operand in the initialization option, and create a framework only for the DB directory.
- For the path of the block special file to be specified for the `adb_init_wrk_blk_path` operand in the initialization options, specify the disk to be used as the work table DB area, which you prepared in step 2.
- For the path of the block special file to be specified for the `adb_init_mst_blk_path`, `adb_init_dic_blk_path`, and `adb_init_stbl_blk_path` operands in the initialization option and for the `adbinitdbarea` option, specify the path of the block special file of the DB area file that was being used by the HADB server before migration.

16.19.8 Preparing to use the audit trail facility

Perform the tasks described in this subsection if the audit trail facility was used in the pre-migration system.

1. Prepare the file systems where the audit trail directory will be created.

   Prepare for each node the file system where the audit trail directory will be created. For details about the file system where the audit trail directory will be created, see 16.25.1 Setting up the audit trail facility environment.

2. Create the audit trail storage directory on a shared file system.

   You must configure the environment in such a way that the audit trail files specified in `ADB_AUDITREAD` functions that retrieve audit trail data are accessible from all nodes using the same absolute path. For this reason, the audit trail storage directory must be created on a file system that is shared among all nodes. For details, see 16.25.1 Setting up the audit trail facility environment.

16.19.9 Starting the HADB servers in the multi-node configuration

Start the HADB servers in the multi-node configuration. For details about how to start the HADB servers, see 16.4.1 Starting the HADB servers in the multi-node configuration.

16.19.10 Preparing for synonym search operations

Perform the tasks described in this subsection if synonym searches were used in the pre-migration system.

1. **Re-create synonym dictionary files**

   If you changed the value specified for the `adb_syndict_storage_path` operand in the server definition, you need to re-create all synonym dictionary files. For details about how to re-create synonym dictionary files, see 15.14.3 Steps to take if synonym dictionary files or their storage directory is accidentally deleted.

   Note that even if you do not change the value of the `adb_syndict_storage_path` operand, mounting of the file system (disk) might entail changes to the file system in which the directory that stores synonym dictionary files was created. In this case, copy all synonym dictionary files to a new directory.

2. **Synchronize synonym dictionary files**

   Execute the `adbsyndict` command to synchronize the synonym dictionary files. For details about how to synchronize synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.

16.20 Upgrading the HADB server version (when the multi-node function is used)

If you are performing a version upgrade of an HADB server that is using the multi-node function, see 8.6 Upgrading the HADB server version.

16.21 Swapping the HADB server with its revised version (when the multi-node function is used)

You can swap the HADB server with its revised version by using either of the following methods:

- Normally terminating and then swapping the HADB server in a multi-node configuration
- Performing swapping on a per-node basis

16.21.1 Normally terminating and then swapping the HADB server in a multi-node configuration

The following shows the procedure for normally terminating the HADB server and then swapping it with its revised version in a multi-node configuration:

Procedure:

1. Normally terminate the HADB server in a multi-node configuration

   For details about the procedure for terminating the HADB server normally in a multi-node configuration, see (a) Normally terminating HADB servers in the multi-node configuration in (1) Termination procedures for HADB servers in a multi-node configuration under 16.4.2 Terminating HADB servers in the multi-node configuration.
2. Swapping of the HADB server with its revised version on all nodes

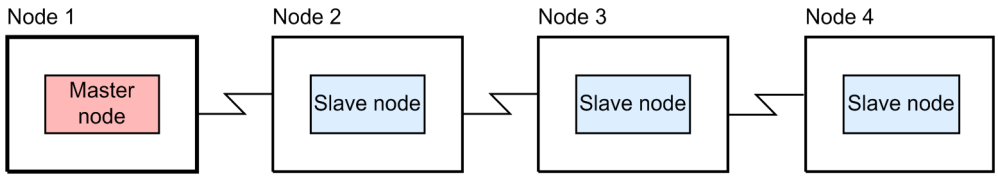
   For details about how to swap the HADB server with a revised version, see 8.8 Swapping the HADB server with its revised version.
3. Normally start the multi-node configuration HADB server

   For details about the procedure for starting the HADB server normally in a multi-node configuration, see (1) Startup procedure for the HADB servers in a multi-node configuration in 16.4.1 Starting the HADB servers in the multi-node configuration.

16.21.2 Performing swapping on a per-node basis

The following procedure shows how to swap the HADB server with its revised version on a per-node basis: If you use this method, you do not have to stop the multi-node configuration HADB server, so there is no need to stop work operations.

The multi-node configuration on which swapping is performed is as follows:



Procedure:

1. Switch over the master node by using a command

   For details about how to switch over the master node using a command, see 16.7 Switching over the master node by using a command.

When switchover of the master node is complete, node 1's HADB server terminates normally, and node 2 becomes the master node.

2. Swap the HADB server with its revised version on node 1

   For details about how to swap the HADB server with a revised version, see 8.8 Swapping the HADB server with its revised version.

3. Return node 1 to the multi-node configuration

   For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

   Node 1 is returned as a slave node to the multi-node configuration.

4. Switch over the master node by using a command

   For details about how to switch over the master node using a command, see 16.7 Switching over the master node by using a command.

   When switchover of the master node is complete, node 2's HADB server terminates normally, and node 1 becomes the master node.

5. Swap the HADB server with its revised version on node 2

   For details about how to swap the HADB server with a revised version, see 8.8 Swapping the HADB server with its revised version.

6. Return node 2 to the multi-node configuration

   For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

   Node 2 is returned as a slave node to the multi-node configuration.

7. Normally terminate the HADB server on node 3

   For details about how to terminate a slave node normally, see (c) Normally terminating the HADB server on a slave node only in (2) Terminating an HADB server on a specific node under 16.4.2 Terminating HADB servers in the multi-node configuration.

8. Swap the HADB server with its revised version on node 3

   For details about how to swap the HADB server with a revised version, see 8.8 Swapping the HADB server with its revised version.

9. Return node 3 to the multi-node configuration

   For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

   Node 3 is returned as a slave node to the multi-node configuration.

Perform the preceding steps 7-9 on node 4 as well.

With the preceding step, swapping of the HADB server with its revised version is complete.

- **If swapping of the HADB server with its revised version fails**

  If swapping of the HADB server with its revised version fails, use the following procedure to return the HADB server to its state before the swapping: You need to perform this procedure on all nodes where swapping of the HADB server with its revised version failed.

  Procedure:

  1. **Forcibly terminating the HADB server on nodes where swapping failed**

     If the HADB server is running on any nodes where swapping failed, execute the `adbstop --force` command to forcibly terminate the HADB server.

  2. **Uninstalling the revised version of the HADB server**

Uninstall the revised version of the HADB server on nodes where swapping with its revised version failed.

3. **Recovering the server directory**

   Use the backup of the server directory you took before performing the swapping, to recover the server directory.

4. **Returning the specifications in the environment variables and kernel parameters to their original values**

   If you changed the specifications in the environment variables and kernel parameters, return those specifications to their original values.

5. **Return nodes to the multi-node configuration**

   Return all nodes where you completed up to step 4, to the multi-node configuration. For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

6. **Switch over the master node by using the command**

   To switch back over to the original master node, perform a switchover by using the command. For details about how to switch over the master node using a command, see 16.7 Switching over the master node by using a command.

7. **Return nodes that were separated from the multi-node configuration, back to the configuration.**

   Return nodes that were separated from the multi-node configuration as a result of performing the master node switchover in step 6, to the multi-node configuration. For details about how to return nodes to a multi-node configuration, see 16.15.3 Returning a node to the multi-node configuration.

16.22 Notes about using SQL tracing (when the multi-node function is being used)

For notes about using SQL tracing, see 10.11.11 Notes about using the multi-node function.

16.23 Handling of data retrieval from CSV files (when the multi-node function is being used)

For details about the process of data retrieval from CSV files, see 11.15 Handling of data retrieval from CSV files. This section provides notes specifically for when the multi-node function is used.

When you are using the multi-node function, ensure that the CSV files can be referenced from all nodes. In addition, ensure that the absolute paths of the CSV files are the same on all nodes.

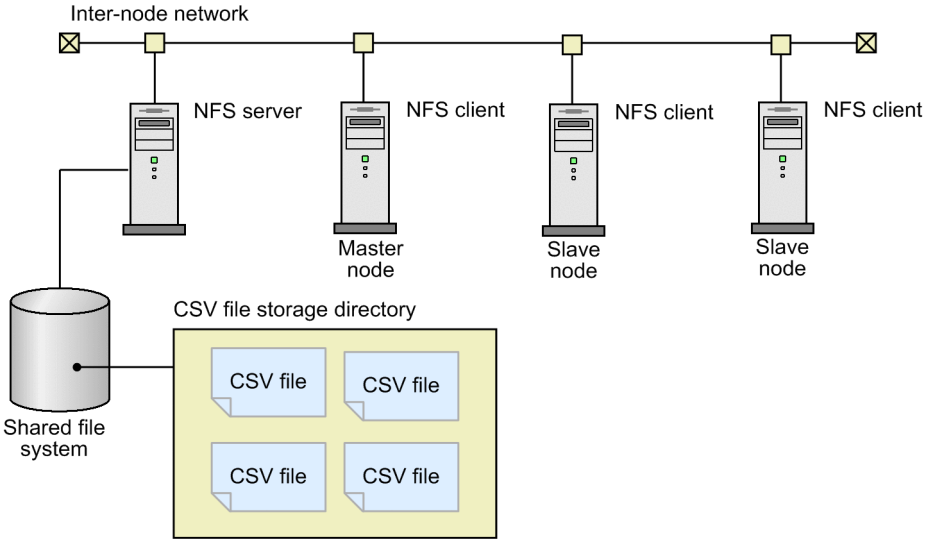
For example, if you are using NFS, which is a distributed file system, prepare directories for storing CSV files by using the following procedure.

Procedure:

1. After preparing the NFS server, create the CSV file storage directory on the NFS server.
2. Export the CSV file storage directory you created in step 1.
   Perform this operation on the NFS server.
3. Mount the directory exported in step 2 so that the paths are the same on all nodes.
   Perform this operation on all nodes.

The following figure provides an example of a system configuration that uses NFS.

Figure 16-9: Example of a system configuration that uses NFS



16.24 Performing synonym search operations (when using the multi-node function)

This section describes the preparatory tasks you need to perform before performing synonym searches, and how to perform synonym search operations.

16.24.1 Preparing for synonym search operations

This subsection describes the preparatory tasks required before performing synonym searches. The information in this subsection assumes that the multi-node function is enabled.

(1) Considering defining text indexes

When you perform synonym search operations, performance improvement is expected if you use text indexes. Therefore, we recommend that you define text indexes for the columns in which text data (document data) is stored. For details, see (1) Considering defining text indexes under 11.16.1 Preparing for synonym search operations.

(2) Estimating the size required for the directory for storing synonym dictionary files

If you intend to perform synonym searches in environments that use the multi-node function, you need to create the directory for storing synonym dictionary files on a file system that must be inherited when node switchover occurs. Before preparing the file system, you need to perform the following tasks:

Estimate the size of the directory for storing synonym dictionary files

Estimate the size of the directory for storing synonym dictionary files, and keep the results of this estimation in mind when preparing the file system. For details about how to estimate the size of the storage directory for synonym dictionary files, see (3) Estimating the size required for the directory for storing synonym dictionary files in 11.16.1 Preparing for synonym search operations.

> **Important**
>
> If there is insufficient free space for the directory for storing synonym dictionary files, you need to change to a new directory for storing synonym dictionary files. If you change the storage directory for synonym dictionary files, you will need to re-register the contents of all synonym dictionaries. Therefore, take care to estimate the size of the directory so that the disk on which the directory is created will have sufficient free space.

> **Note**
>
> The directory for storing synonym dictionary files is the directory specified for the `adb_syndict_storage_path` operand in the server definition.

(3) Preparing the file system where the directory that stores synonym dictionary files is to be created

You must create the directory for storing synonym dictionary files on a file system that must be inherited when node switchover occurs. Create file systems that must be inherited when node switchover occurs on an LV in a VG that

consists of disks that can be referenced from all nodes. For details about shared disks, see *Shared disk configuration* in the manual *HA Monitor for Linux(R) (x86)*.

(4) Initializing the file system where the directory that stores synonym dictionary files is to be created

Initialize the file system where the directory that stores synonym dictionary files is to be created. The following execution example initializes `/dev/vg_hadb02/hadb_syndict`, which was created as the LV where the directory for storing synonym dictionary files is to be created, as an ext4 file system.

```
mkfs -t ext4 /dev/vg_hadb02/hadb_syndict
```

(5) Creating the directory for storing synonym dictionary files

After you have finished preparing the file system, create the directory for storing synonym dictionary files and assign the appropriate permissions to the directory you created. For details, see (4) Creating the directory for storing synonym dictionary files to (5) Assigning permissions to the directory for storing synonym dictionary files in 11.16.1 Preparing for synonym search operations.

Important

- Do not delete the storage directory for synonym dictionary files. Also, do not use any method other than the `adbsyndict` command to delete files under the storage directory for synonym dictionary files. If you delete the directory or use another method to delete files, an error will occur when you execute a synonym search.
- Do not use any method other than the `adbsyndict` command to update files under the storage directory for synonym dictionary files. If you use another method to update files, the results of subsequent synonym searches might be invalid, or an error might occur when you execute a synonym search.

Note

Include the directory for storing synonym dictionary files when taking a full backup. For details, see (2) Backup procedure under 10.3.1 Backup acquisition method.

(6) Creating the multi-node synonym dictionary storage directory

If you will perform synonym searches in an environment that uses the multi-node function, create the multi-node synonym dictionary storage directory on a node-local file system. You must create the multi-node synonym dictionary storage directory on all nodes.

Note

The multi-node synonym dictionary storage directory is the directory specified for the `adb_syndict_node_storage_path` operand in the server definition.

When SQL statements that perform synonym searches are executed on the slave node, these SQL statements use the synonym dictionary files in the directory you create. When you use the `adbsyndict` command to register, modify,

or delete a synonym dictionary file, the synonym dictionary file on the master node is automatically copied to the slave node and stored in the directory you create.

> **Important**
>
> You can still perform synonym searches when there are no synonym dictionary files stored on the slave node or the synonym dictionary files on the slave node are not the latest files. However, in this case, when execution processing for SQL statements that perform synonym searches is assigned to such a slave node, that slave node will not execute the SQL statements. Execution of the SQL statements will take place on the master node. If this state continues, it can lead to an increased load on the master node. Ensure that you synchronize the synonym dictionary files so that the latest synonym dictionary files are always stored on the slave node. For details about how to synchronize synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.

The following shows the procedure for creating the multi-node synonym dictionary storage directory.

Procedure:

1. Estimate the size of the multi-node synonym dictionary storage directory.

Estimate the size of the multi-node synonym dictionary storage directory, and consider the results of this estimation when creating the directory. For details about the how to estimate the size of the multi-node synonym dictionary storage directory, see 6.17 Estimating the size of the multi-node synonym dictionary storage directory.

> **Important**
>
> If there is insufficient free space available for the multi-node synonym dictionary storage directory, you need to change to a new directory for storing synonym dictionary files. If you change the multi-node synonym dictionary storage directory, you will need to synchronize the synonym dictionary files. Therefore, take care to estimate the size of the directory so that the disk on which the directory is created will have sufficient free space.

2. Create the multi-node synonym dictionary storage directory.

Create the multi-node synonym dictionary storage directory. Note the following when creating this directory:

- You cannot use the following directories as the multi-node synonym dictionary storage directory:
  - Server directory
  - Subdirectories of the server directory
  - Directories of which the server directory is subordinate
  - DB directory
  - Subdirectories of the DB directory
  - Directories of which the DB directory is subordinate
  - Root directory
  - Directory specified in the `adb_syndict_storage_path` operand
  - Subordinate directory of the directory specified in the `adb_syndict_storage_path` operand
  - Directory to which the directory specified in the `adb_syndict_storage_path` operand is subordinate

The following table shows examples of directories that can and cannot be used as the multi-node synonym dictionary storage directory when the DB directory is `/HADB/db`.

Examples of multi-node synonym dictionary storage directories		Reason
Example of directory that can be used as multi-node synonym dictionary storage directory	/HADB/syndict	None.
Examples of directories that cannot be used as multi-node synonym dictionary storage directory	/HADB/db	This directory cannot serve as the multi-node synonym dictionary storage directory because it is also the DB directory.
	/HADB/db/synonym	This directory cannot serve as the multi-node synonym dictionary storage directory because it is a subordinate directory of the DB directory.
	/HADB	This directory cannot serve as the multi-node synonym dictionary storage directory because the DB directory is one of its subordinate directories.

- Do not specify the directory in which you stored the installation data when installing the HADB server as the multi-node synonym dictionary storage directory.

3. Assign permissions for the multi-node synonym dictionary storage directory.

The following shows the procedure for assigning the necessary permissions in relation to the multi-node synonym dictionary storage directory.

Procedure:

1. Assign read, write, and execution permissions to the HADB administrator for the multi-node synonym dictionary storage directory.
2. Assign execution permission to the HADB administrator for all directories that are included in the path of the multi-node synonym dictionary storage directory.

Example: When the multi-node synonym dictionary storage directory is /HADB/syndict

- Read, write, and execution permissions must be assigned to /HADB/syndict.
- Execution permission must be assigned to / and /HADB.

Important

- If you change the multi-node synonym dictionary storage directory, you must execute the adbsyndict command to synchronize the synonym dictionary files. If you do not synchronize the files, the node on which SQL statements that perform synonym searches are executed will change from the slave node to the master node. This can increase the load on the master node and introduce overhead associated with switching the execution nodes, potentially leading to a drop in performance.
- Do not delete the multi-node synonym dictionary storage directory. Also, do not use any method other than the adbsyndict command to delete files under the multi-node synonym dictionary storage directory. If you delete the directory, the node on which SQL statements that perform synonym searches are executed will change from the slave node to the master node. This can increase the load on the master node and introduce overhead associated with switching the execution nodes, potentially leading to a drop in performance.
- Do not use any method other than the adbsyndict command to update files under the multi-node synonym dictionary storage directory. If you use another method to update files, the results of subsequent synonym searches might be invalid, or an error might occur when you execute a synonym search.

(7) Changing the specification of the HA Monitor servers file

Terminate the HADB servers in the multi-node configuration before changing the specification of the HA Monitor `servers` file.

In the example in Figure 16-1: Example of a system configuration using the multi-node function, `FS002` is the file system in which the storage directory for synonym dictionary files is to be created. The name of the VG is `vg_hadb02`, and the name of the LV is `hadb_syndict`.

Change the values specified for the following operands:

- `disk`

  Add the absolute path of the VG that contains the file system where the directory that stores synonym dictionary files is to be created. For the example in Figure 16-1: Example of a system configuration using the multi-node function, you add `/dev/vg_hadb02`.

- `fs_name`

  Add the absolute path of the logical volume that corresponds to the file system where the directory that stores synonym dictionary files is to be created. For the example in Figure 16-1: Example of a system configuration using the multi-node function, you add `/dev/vg_hadb02/hadb_syndict`.

- `fs_mount_dir`

  Add the mount directory path of the file system where the directory that stores synonym dictionary files is to be created. As the mount directory path of the file system where the directory that stores the synonym dictionary files is to be created, specify the same path on each node.

- `fs_mount_opt`

  Add the mount option for mounting the file system where the directory that stores synonym dictionary files is to be created.

- `vg_neck`

  Add a `use` specification.

- `fs_neck`

  Add a `use` specification.

- `scsi_device`

  In this operand, specify the absolute paths of the device names of the file systems that store the synonym dictionary files (the device names that are to be subject to SCSI reservation).

  When using host reset, you do not need to specify this operand.

  Specify this operand when you are using SCSI reservation for shared disk and the shared disk meets any of the following conditions:

  - The shared disk is in a single-path configuration.
  - The shared disk is in a VMware ESXi virtual environment (excluding situations where DMMP is used).
  - The shared disk is in a redundant configuration realized using multipath software (HDLM).

- `dmmp_device`

  In this operand, specify the device names of the file systems that store the synonym dictionary files (the device names that are to be subject to SCSI reservation) as the absolute paths of the logical devices in DMMP.

  When using host reset, you do not need to specify this operand.

  Specify this operand when you are using SCSI reservation for shared disk, and the shared disk is in a redundant configuration based on multipath software (DMMP).

For examples of specifying the `servers` file, see the following sections:

- When using host reset

  See (b) servers file specification examples in (5) File specification examples (when using host reset) under 16.3.4 Setting up an HA Monitor environment.

- When using SCSI reservation for shared disk

  For a single-path configuration, see (b) Specification example of a servers file (when using a single-path configuration) in (6) File specification examples (when using SCSI reservation for shared disk) under 16.3.4 Setting up an HA Monitor environment.

  For a redundant configuration realized by multipath software, see (c) Specification example of a servers file (when using a redundant configuration realized by multipath software) in (6) File specification examples (when using SCSI reservation for shared disk) under 16.3.4 Setting up an HA Monitor environment.

For details about the operands in the `servers` file, see *Server environment definition (servers)* in the manual *HA Monitor for Linux(R) (x86)*.

(8) Amending the server definition

Specify the following operands in the server definition:

- `adb_syndict_storage_path`

  In the `adb_syndict_storage_path` operand in the server definition, specify the name of the directory for storing synonym dictionary files. Specify this operand in the server definitions of all nodes.

  For details about the `adb_syndict_storage_path` operand, see the explanation of the adb_syndict_storage_path operand in 7.2.8 Operands related to synonym search (set format).

  If you specify a symbolic link to the storage directory for synonym dictionary files in the `adb_syndict_storage_path` operand, a check is performed to determine whether the absolute path name generated after the symbolic link is resolved conforms to the rules explained in (4) Creating the directory for storing synonym dictionary files under 11.16.1 Preparing for synonym search operations.

- `adb_syndict_node_storage_path`

  In the `adb_syndict_node_storage_path` operand in the server definition, specify the name of the multi-node synonym dictionary storage directory. Specify this operand in the server definitions of all nodes.

  For details about the `adb_syndict_node_storage_path` operand, see the explanation of the adb_syndict_node_storage_path operand in 7.2.8 Operands related to synonym search (set format).

  If you specify a symbolic link to the multi-node synonym dictionary storage directory in the `adb_syndict_node_storage_path` operand, a check is performed to determine whether the absolute path name generated after the symbolic link is resolved conforms to the rules explained in **2. Create the multi-node synonym dictionary storage directory.** under (6) Creating the multi-node synonym dictionary storage directory.

> **Note**
>
> For details about the procedure for changing the server definition, see 8.5.2 Modifying the server definition.

(9) Starting the HADB servers in the multi-node configuration

Start the HADB servers in the multi-node configuration.

For details about how to start the HADB servers in the multi-node configuration, see 16.4.1 Starting the HADB servers in the multi-node configuration.

(10) Defining text indexes

If you decide to define text indexes after reading (1) Considering defining text indexes, use the following procedure to define the text indexes:

Procedure:

1. Define text indexes by executing `CREATE INDEX` statements.
2. Create the index data for the text indexes by executing the `adbidxrebuild` command.

(11) Registering synonym dictionaries

To register synonym dictionaries, perform the tasks in (7) Creating a synonym list definition file to (10) Making a backup of the directory for storing synonym dictionary files in 11.16.1 Preparing for synonym search operations.

> **Important**
>
> We recommend that you create synonym list definition files, dictionary creation files, and dictionary deletion files on a file system that is accessible from the master node. You could create these files on a file system that is inherited when node switchover occurs, like the storage directory for synonym dictionary files. Alternatively, you might create the files on a disk that is accessible from all nodes and whose mount directory is the same on all nodes.

When you register synonym dictionaries by executing the `adbsyndict` command on the master node, the synonym dictionary files are automatically synchronized. This means that you do not need to specify the `-s` option when executing the `adbsyndict` command.

16.24.2 Performing synonym searches

The procedures for performing synonym searches are the same as the procedures when the multi-node function is not used. For details about how to perform synonym searches, see 11.16.2 Performing synonym search operations.

Synonym searches performed in an environment that uses the multi-node function use the synonym dictionary files on the node to which execution processing of the SQL statement was assigned. The synonym dictionary files used in the multi-node function are stored in the directory specified for the `adb_syndict_node_storage_path` operand in the server definition. When execution processing of an SQL statement is assigned to a slave node that does not have the latest synonym dictionary files in this directory, that slave node does not process the SQL statement. In this case, execution of the SQL statement takes place on the master node instead. At this time, the `KFAA51537-W` message is output.

If this state continues, performance might drop for the following reasons:

- Execution processing of SQL statements that execute synonym searches becomes concentrated on the master node, increasing the load on the master node
- Processing takes place to change the node on which SQL statements that perform synonym searches are executed from the slave node to the master node

For this reason, take care to synchronize the synonym dictionary files so that the latest synonym dictionary files are always stored on the slave node. For details about how to synchronize the synonym dictionary files, see 16.24.5 Synchronizing synonym dictionary files.

16.24.3 Registering or deleting synonym dictionaries

The procedures for registering and deleting synonym dictionaries are the same as the procedures when the multi-node function is not used. For details about how to register synonym dictionaries, see 11.16.8 Registering a synonym dictionary. For details about how to delete synonym dictionaries, see 11.16.9 Deleting synonym dictionaries.

> **Note**
>
> When you register or delete synonym dictionaries by executing the `adbsyndict` command on the master node, the synonym dictionary files are automatically synchronized. This means that you do not need to specify the `-s` option when executing the `adbsyndict` command.

16.24.4 Adding or deleting synonyms

The procedures for adding and deleting synonyms are the same as the procedures when the multi-node function is not used. For details about how to add and delete synonyms, see 11.16.5 Adding synonyms and 11.16.6 Deleting synonyms.

When you add or delete synonyms by executing the `adbsyndict` command on the master node, the synonym dictionary files are automatically synchronized. This means that you do not need to specify the `-s` option when executing the `adbsyndict` command.

> **Note**
>
> The procedures for adding and deleting synonym groups are the same as the procedures when the multi-node function is not used. For details about how to add synonym groups, see 11.16.7 Adding synonym groups.

16.24.5 Synchronizing synonym dictionary files

To synchronize synonym dictionary files, you execute the following command:

```
adbsyndict -s
```

About synchronization of synonym dictionary files

When using the multi-node function, the synonym dictionary files must be present on all nodes. The process of ensuring that each node has the same synonym dictionary files is called synchronizing synonym dictionary files. This synchronization process involves copying the synonym dictionary files on the master node to the slave nodes.

The synonym dictionary files on a slave node are stored in the directory specified for the `adb_syndict_node_storage_path` operand. When SQL statements that perform synonym searches are executed on a slave node, these SQL statements use the synonym dictionary files on the slave node.

If synonym dictionary files are not synchronized

You can still perform synonym searches if the synonym dictionary files have not been synchronized. However, if execution processing for SQL statements that perform synonym searches is assigned to a slave node whose synonym dictionary files have not been synchronized, that slave node will not execute the SQL statements. In this case, execution of the SQL statements will take place on the master node. If this state continues, it can lead to an increased

load on the master node. For this reason, ensure that the synonym dictionary files are synchronized and the synonym dictionary files on the slave nodes are the latest files.

Cases where synonym dictionary files need to be synchronized

Circumstances in which you need to synchronize synonym dictionary files include returning a node to a multi-node configuration or adding a node. If the latest synonym dictionary files are not stored on the slave node, the warning message `KFAA51537-W` is output when you start the HADB server and then you execute a synonym search. If you encounter this message, execute the `adbsyndict` command to synchronize the synonym dictionary files.

Note

If you register, update, or delete a synonym dictionary, the HADB server will synchronize the synonym dictionary files automatically.

16.25 Audit trail facility operations (when the multi-node function is being used)

This section describes how to set up the environment when using the audit trail facility, and how to use the audit trail facility.

To collect and centrally manage audit trails by linking with JP1/Audit, see 16.25.3 Linkage with JP1/Audit Management - Manager (when the multi-node function is being used).

16.25.1 Setting up the audit trail facility environment

This subsection provides notes that apply when using the audit trail facility together with the multi-node function. For details about how to set up the environment for the audit trail facility, see 12.2 Setting up the audit trail facility environment.

■ Creating the audit trail directory

- You must create an audit trail directory on each node.
- We recommend that you create the audit trail directory on a node-local file system on each node.
- You must specify the `adb_audit_log_path` operand in the server definition for each node. This operand specifies the location of the audit trail directory.

> **Note**
>
> Audit trails are output on the node where the corresponding operation was performed. For this reason, you need to create an audit trail directory for each node, and specify the name of that directory in the applicable `adb_audit_log_path` operand.

■ Preparing the file systems where the audit trail storage directories will be created

Configure the environment such that the paths of the audit trail files specified in the `ADB_AUDITREAD` function that retrieves audit trail data are accessible from all nodes using the same absolute paths.

> **Important**
>
> - If audit trail files cannot be read by the same absolute path from all nodes, the results of transactions that execute SQL statements that retrieve audit trail data will differ depending on the node to which the transaction was assigned.
> - In the `ADB_AUDITREAD` function, you must specify the absolute path of the audit trail file from which to retrieve data.

Create the audit trail storage directory on a file system that is shared among all nodes. The following is an example of environment setup when using an NFS (a type of distributed file system):

Procedure:

1. After preparing the NFS server, create the audit trail storage directory on the NFS server.
2. Export the audit trail storage directory you created in step 1.
   Perform this operation on the NFS server.
3. Mount the directory exported in step 2 so that the paths are the same on all nodes.
   Perform this operation on all nodes.

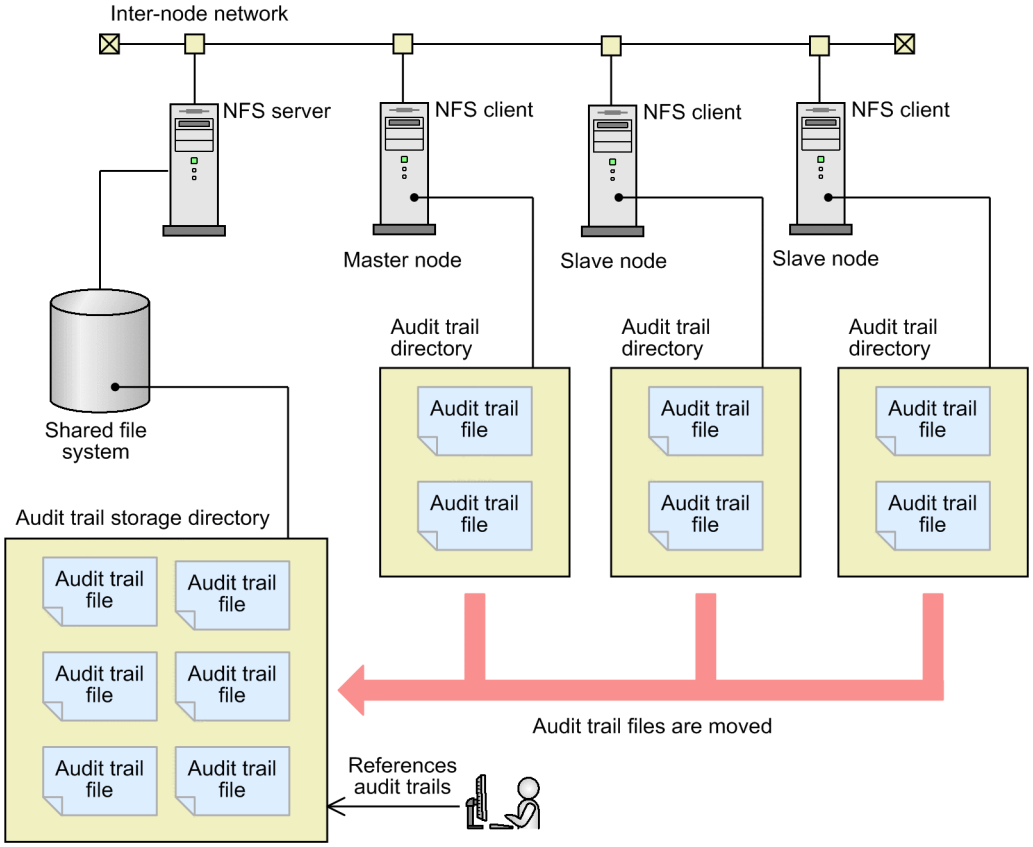
4. Move the audit trail files that were output to the audit trail directory on each node to the audit trail storage directory on the NFS server.

   Perform this operation on all nodes.

5. When retrieving audit trail data, specify in the `ADB_AUDITREAD` function the path of the audit trail files in the audit trail storage directory.

The following figure shows an example of a system configuration that uses NFS:

Figure 16-10: Example of a system configuration that uses NFS



16.25.2 Operating the audit trail facility

This subsection provides notes that apply to the operation of the audit trail facility in environments where it is used together with the multi-node function. For details about how to operate the audit trail facility, see 12. Audit Trail Facility Operations.

■ Enabling or disabling the audit trail facility

To enable the audit trail facility, execute the `adbauditttrail --start` command on the master node. This enables the audit trail facility on all nodes.

To disable the audit trail facility, execute the `adbauditttrail --stop` command on the master node. This disables the audit trail facility on all nodes.

Note

You cannot enable or disable the audit trail facility on individual nodes.

■ **Swapping audit trail files**

On the node whose audit trail file you want to swap, execute the `adbauditttrail --swap` command.

You can also swap the audit trail file from the master node. To do this, specify the node number of the node whose audit trail file you want to swap for the `-n` option when executing the `adbauditttrail --swap` command.

To identify the node number of the node whose audit trail file you want to swap, execute the `adbls -d node` command. For details about the `adbls -d node` command, see *adbls -d node (Display the HADB Server Status on Each Node)* in the manual *HADB Command Reference*.

■ **Displaying information related to the audit trail facility**

On the node for which you want to check the information related to the audit trail facility, execute the `adbauditttrail -d` command.

You can also check the information related to the audit trail facility from the master node. To do this, specify the node number of the node for which you want to check the information related to the audit trail facility for the `-n` option when executing the `adbauditttrail -d` command.

To identify the node number of the node for which you want to check the information related to the audit trail facility, execute the `adbls -d node` command. For details about the `adbls -d node` command, see *adbls -d node (Display the HADB Server Status on Each Node)* in the manual *HADB Command Reference*.

16.25.3 Linkage with JP1/Audit Management - Manager (when the multi-node function is being used)

To collect and centrally manage audit trails by linking with JP1/Audit, use the `adbconvertaudittrailfile` command to convert audit trails, and then output the conversion results to a common format audit trail file. For details about conversion of audit trail information, see 2.18.9 Conversion of audit trail information (linkage with JP1/Audit Management - Manager).

The `adbconvertaudittrailfile` command can be executed on the master node or slave node. You do not need to execute the `adbconvertaudittrailfile` command on the node that output the audit trail file. Therefore, the following two operation methods can be used:

- Operation method in which the master node and slave node output and convert (by using the `adbconvertaudittrailfile` command) audit trail files individually
- Operation method in which the master node uses the `adbconvertaudittrailfile` command to convert all audit trail files

Whichever operation method you use, you can check the same information with JP1/Audit. Note that you must create the output-directory for common format audit trails on a node-local file system.

The following provides notes on using each operation method.

■ **If the master node and slave node output and convert audit trail files individually (by using the `adbconvertaudittrailfile` command)**

Specify the environment settings of JP1/Audit on all nodes on which the `adbconvertaudittrailfile` command is executed.

For details about the environment settings of JP1/Audit, see (2) Environment settings for JP1/Audit in 12.8.3 Environment settings for linking the audit trail facility with JP1/Audit. For details about the `adbconvertaudittrailfile` command, see *adbconvertaudittrailfile (Convert the Audit Trail File)* in the manual *HADB Command Reference*.

You do not need to set up JP1/Base in a cluster environment.

> **Important**
>
> Create the output-directory for common format audit trails on each node. If multiple nodes simultaneously attempt to output common format audit trail files to the same directory, these files are not correctly output.

The following shows an example of the operation procedure:

1. On each node, after an audit trail file has been output to the audit trail directory, use the `adbconvertaudittrailfile` command to convert the file to a common format audit trail file.
2. When step 1 finishes, move the audit trail files from the audit trail directories of all nodes to the audit trail storage directory that is shared by all nodes.

■ If the master node uses the `adbconvertaudittrailfile` command to convert all audit trail files

Specify the environment settings of JP1/Audit on the master node on which the `adbconvertaudittrailfile` command is executed.

When the master node is switched, if you want the `adbconvertaudittrailfile` command to be executed on the new master node, specify the environment settings of JP1/Audit on all nodes. In this case, you do not need to set up JP1/Base in a cluster environment.

For details about the environment settings of JP1/Audit, see (2) Environment settings for JP1/Audit in 12.8.3 Environment settings for linking the audit trail facility with JP1/Audit. For details about the `adbconvertaudittrailfile` command, see *adbconvertaudittrailfile (Convert the Audit Trail File)* in the manual *HADB Command Reference*.

The following shows an example of the operation procedure:

1. On each node, move the audit trail file from the audit trail directory to the audit trail storage directory that is shared by all nodes.
2. After the audit trail files have been moved to the audit trail storage directory, on the master node, execute the `adbconvertaudittrailfile` command to convert those files into common format audit trail files.

16.26 Updated-row columnizing facility operations (when the multi-node function is being used)

For details about the operation of the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables). This section provides notes specifically for when the multi-node function is used.

- Execute the `adbcolumnize` command on the master node.
- The status of the updated-row columnizing facility (whether the facility is enabled or disabled) is maintained after a master node switchover occurs.
- The maintenance processing of the updated-row columnizing facility is performed on the master node.
- The maintenance processing is not performed while a transaction or a command that connects to an HADB server is running on an HADB server in a multi-node configuration.
- If a master node switchover occurs when the updated-row columnizing facility is enabled, the maintenance processing is performed after the switchover is completed.
- If you leave the updated-row columnizing facility enabled without handling any errors related to that facility, the maintenance processing that is performed after the master node switchover is complete will result in an error, and an error message will be output.
- If no maintenance processing is performed for 24 hours consecutively, the `KFAA51277-I` message is output. However, the monitoring timer is reset when a master node switchover occurs. Therefore, if no maintenance processing is performed for 24 hours consecutively again after a master node switchover occurs, the `KFAA51277-I` message is output again.
- The processing that starts I/O control of files used by the updated-row columnizing facility is performed on each node. If such processing fails on the new master node, the maintenance processing that is performed after the master node switchover is complete will result in an error.

16.27 Notes on using the multi-node function

- If processing cannot be executed by the HADB servers on all nodes, the HADB servers in the multi-node configuration terminate abnormally.

17 Operations When Using the Cold Standby Configuration

This chapter explains how to build and run a system in the cold standby configuration.

17.1 Notes on reading this chapter

The explanation of the cold standby configuration in this chapter assumes that the reader has a basic knowledge of HA Monitor. For details about HA Monitor, see the manual *HA Monitor for Linux(R) (x86)*.

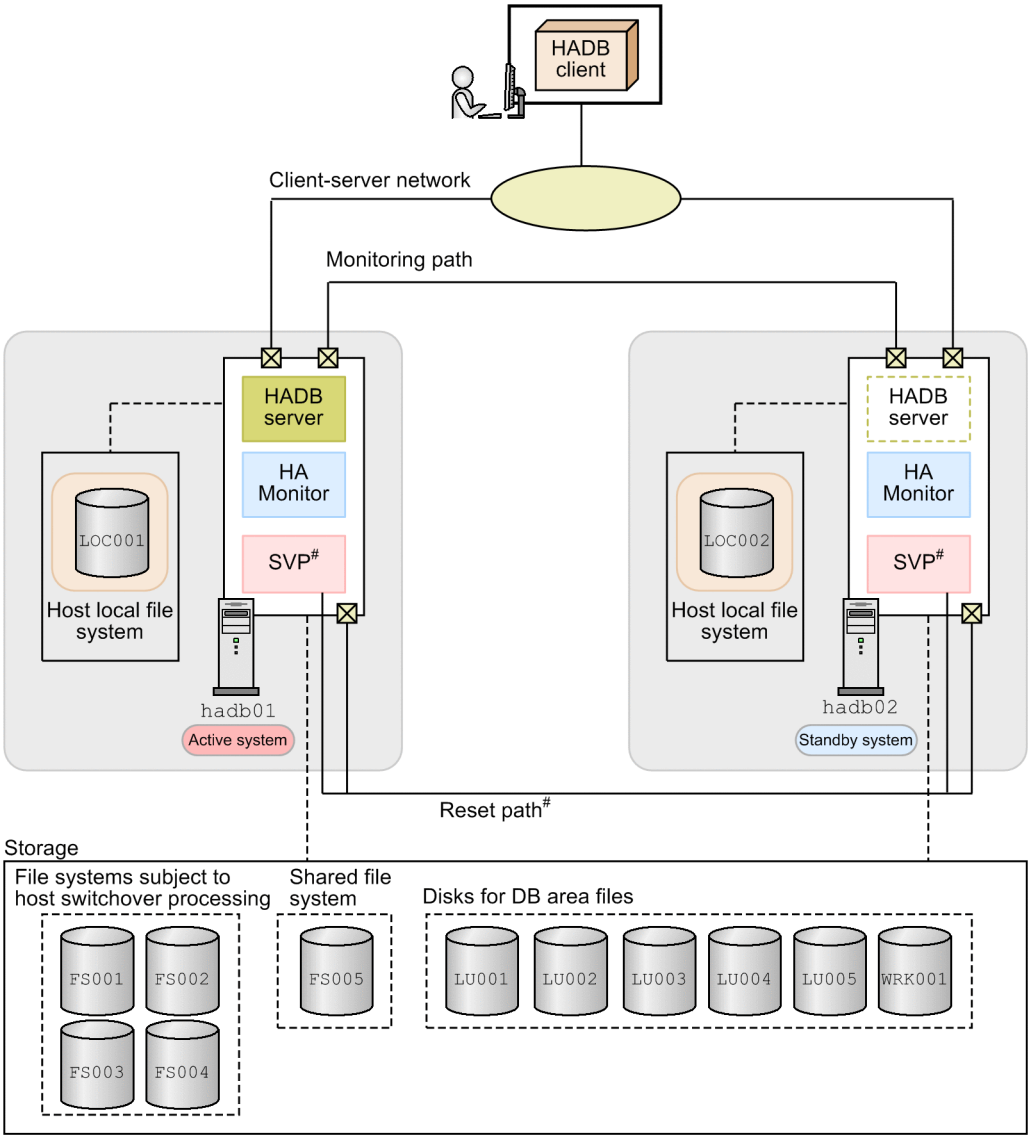
When reading the manual *HA Monitor for Linux(R) (x86)*, note the following points:

- The HADB server operates in monitor mode.
- The configuration inherits the alias IP address.
- The HADB server runs as a program that does not have an interface to HA Monitor.
- Server failures are detected without issuing an API.
- HA Monitor Extension cannot be used.

17.2 Example of system configuration using the cold standby configuration

The following figure shows an example of a system configuration using the cold standby configuration.

Figure 17-1: Example of a system configuration using the cold standby configuration



#

When using the SCSI reservation for shared disk method, neither an SVP nor a reset path are required.

Important

You can select either of the following as the shared disk data protection method that uses HA Monitor. We recommend you use host reset.

- Host reset
- SCSI reservation for shared disk

17.2.1 Prerequisite software program

Cold standby configurations require HA Monitor.

HADB uses HA Monitor's monitor mode function to monitor failures in the active system. If a failure occurs in the active system, HA Monitor's hot standby function switches operation to the standby system.

For details on the host switching that takes place in the event of a failure, see 17.12 Operations when a failure occurs.

Note that depending on the version of Linux you are using, the prerequisite version of HA Monitor differs as follows:

- Red Hat Enterprise Linux Server 6 (64-bit x86_64): 01-56-01 or later
- Red Hat Enterprise Linux Server 7 (64-bit x86_64): 01-61 or later

17.2.2 Server configuration

The example configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration consists of two hosts (`hadb01` and `hadb02`). The active system is `hadb01`, and the standby system is `hadb02`.

When you use the cold standby configuration, the server machines of the active system and the standby system do not need to be equivalent in terms of performance characteristics (CPU and memory size, for example). However, if performance varies between the systems, the processing performance for SQL statements might also vary after a system switchover has occurred. For this reason, we recommend that you use server machines of equivalent or nearly equivalent performance for the active and standby systems.

Note that HA Monitor's hot standby function is used in the cold standby configuration. Select either of the following as the shared disk data protection method when switching the host. We recommend that you select host reset.

- Host reset
- SCSI reservation for shared disk

If a server machine in each system contains a failure management processor (SVP shown in Figure 17-1: Example of a system configuration using the cold standby configuration), use the recommended host reset.

The name of the failure management processor depends on the machine being used, such as SVP, BMC, or management server. For details about host reset, failure management processor, and SCSI reservation for shared disk, see the following sections in the manual *HA Monitor for Linux(R) (x86)*:

- Host reset: *Host reset*
- Failure management processor: *Required hardware*
- SCSI reservation for shared disk: *SCSI reservation for shared disk*

17.2.3 Network configuration

The cold standby configuration uses the three networks listed below. Keep these networks physically separated.

- Client-server network
- Monitoring path

- Reset path

(1) Client-server network

This network is used for communication between HADB clients and HADB servers.

An HADB client uses an alias IP address to connect to the HADB server. Therefore, set the alias IP addresses based on the explanation in *Inheriting a LAN* in the manual *HA Monitor for Linux(R) (x86)*.

The following table shows an example of IP address and port number settings for the system configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration:

Table 17-1: Setting examples for IP addresses and port numbers in a client-server network

No.	Setting location	IP address	Port number
1	Client machine	`10.196.108.111`	Setting not required
2	Server machine `hadb01` (active system)	`10.196.108.11`	`23650`
3	Server machine `hadb02` (standby system)	`10.196.108.12`	`23650`
4	Alias IP address	`10.196.108.143`	`23650`

(2) Monitoring path

This network is used as HA Monitor's monitoring path.

When you build a network, we recommend that you achieve NIC redundancy using Linux's bonding function. Furthermore, when you use the bonding function, have multiple LAN cards available.

For details about the bonding function, see the documentation for the operating system you are using. For details about how to set HA Monitor's monitoring path, see *Configuring the monitoring paths* in the manual *HA Monitor for Linux(R) (x86)*.

The following table shows an example of IP address and port number settings for the system configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration:

Table 17-2: Setting examples for IP addresses and port numbers of monitoring paths

No.	Setting location	IP address	Port number
1	HA Monitor's monitoring path for server machine `hadb01` (active system)	`172.16.0.11`	`7777`
2	HA Monitor's monitoring path for server machine `hadb02` (standby system)	`172.16.0.12`	`7777`

(3) Reset path

When using host reset, a reset path is required. SCSI reservation for shared disk does not require a reset path.

A reset path is a network that is used to reset input/output on the host on which a failure is detected by HA Monitor. This network needs to be dedicated to the reset path. For details, see *Reset path configuration* in the manual *HA Monitor for Linux(R) (x86)*.

The following table shows an example of IP address and port number settings for the system configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration:

Table 17-3: Setting examples for IP addresses and port numbers in a reset path

No.	Setting location	IP address	Port number
1	Reset path of server machine `hadb01` (active system)	`192.168.0.11`	`11111`
2	SVP of server machine `hadb01` (active system)	`192.168.0.21`	`22222`
3	Reset path of server machine `hadb02` (standby system)	`192.168.0.12`	`11111`
4	SVP of server machine `hadb02` (standby system)	`192.168.0.22`	`22222`

As the shared disk data protection method used by HA Monitor, we recommend that you use host reset. In this case, a reset path is required. In addition, the server machine on which the HADB server operates must be equipped with a failure management processor such as an SVP. For details, see *Required hardware* in the manual *HA Monitor for Linux(R) (x86)*.

17.2.4 Storage configuration

You must provide the following file systems and disks when you use the cold standby configuration.

- **Host's local file system**
  In Figure 17-1: Example of a system configuration using the cold standby configuration, `LOC001` and `LOC002` are the host's local file systems.
- **File systems subject to host switchover processing**
  In Figure 17-1: Example of a system configuration using the cold standby configuration, `FS001` to `FS004` are file systems subject to host switchover processing.
- **Shared file system**
  In Figure 17-1: Example of a system configuration using the cold standby configuration, `FS005` is the shared file system.
- **Disks for DB area files**
  In Figure 17-1: Example of a system configuration using the cold standby configuration, `LU001` to `LU005` and `WRK001` are the disks for DB area files.

Note that some file systems and disks require device names used to identify disks in the active system and the standby system. If the path between the server and storage is a single-path configuration, we recommend that you use WWNs. When implementing redundant paths between server and storage, the device name will differ according to the redundancy method being used. Examples of device names in Figure 17-1: Example of a system configuration using the cold standby configuration are as follows:

- Examples of device names in single-path configuration

```
FS001: /dev/disk/by-id/wwn-0x60060e8010205850051104c500000005
FS002: /dev/disk/by-id/wwn-0x60060e8010205850051104c500000006
FS003: /dev/disk/by-id/wwn-0x60060e8010205850051104c500000007
FS004: /dev/disk/by-id/wwn-0x60060e8010205850051104c500000008
LU001: /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000f
LU002: /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000e
LU003: /dev/disk/by-id/wwn-0x60060e8010205850051104c500000010
LU004: /dev/disk/by-id/wwn-0x60060e8010205850051104c500000011
```

LU005: /dev/disk/by-id/wwn-0x60060e8010205850051104c500000013
WRK001: /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000d

- Examples of device names in redundant configuration using multipath software (DMMP)

FS001: /dev/mapper/mpath1
FS002: /dev/mapper/mpath2
FS003: /dev/mapper/mpath3
FS004: /dev/mapper/mpath4
LU001: /dev/mapper/mpath11
LU002: /dev/mapper/mpath12
LU003: /dev/mapper/mpath13
LU004: /dev/mapper/mpath14
LU005: /dev/mapper/mpath15
WRK001: /dev/mapper/mpath16

(1) Host's local file system

The directories to be placed in the host's local file system are as follows:

- Server directory

  To create a server directory, use the `adbinstall` command.

  > **Important**
  >
  > Create a server directory on both the active and standby systems. Do not create a server directory on the shared disk (host switchover target file system) or in a shared file system.

- Output-directory for common format audit trails

  For details about the output-directory for common format audit trails, see 2.18.9 Conversion of audit trail information (linkage with JP1/Audit Management - Manager).

In Figure 17-1: Example of a system configuration using the cold standby configuration, `LOC001` and `LOC002` are the host's local file systems.

- `LOC001`

  Local file system of server machine `hadb01` (active system)

- `LOC002`

  Local file system of server machine `hadb02` (standby system)

(2) File systems subject to host switchover processing

The following file systems are subject to host switchover processing:

- File system that stores the DB directory
- File system that stores temporary work files
- File system that stores synonym dictionary files
- File system where the audit trail directory will be created

Create a file system subject to host switchover processing on an LV in a VG that consists of disks (shared disks) that can be referenced from both the active system and the standby system. For details about shared disks, see *Shared disk configuration* in the manual *HA Monitor for Linux(R) (x86)*.

> **Note**
>
> For details about how to create the DB directory, see 17.3.7 Creating a database.

In Figure 17-1: Example of a system configuration using the cold standby configuration, `FS001` to `FS004` are the file systems inherited at host switchover.

- `FS001`
  File system that stores the DB directory
  In this example, the name of the VG is `vg_hadb01` and the name of the LV is `hadb_db`.
- `FS002`
  File system that stores temporary work files
  In this example, the name of the VG is `vg_hadb02` and the name of the LV is `hadb_workarea`.
- `FS003`
  File system that stores synonym dictionary files
  In this example, the name of the VG is `vg_hadb03`, and the name of the LV is `hadb_syndict`.
  The file system that stores synonym dictionary files is needed to perform synonym searches.
- `FS004`
  File system where the audit trail directory will be created
  In this example, the name of the VG is `vg_hadb04` and the name of the LV is `hadb_audit`.
  The file system where the audit trail directory will be created is needed when the audit trail facility is used.

> **Important**
>
> When using SCSI reservation for shared disk, make sure that disks on which file systems subject to host switchover processing have been created are subject to SCSI reservation processing. Do not create any area other than a file system subject to host switchover processing on a disk subject to SCSI reservation processing.

(3) Shared file system

The following file system is placed on a shared file system:

- File system for storing input data files that are used for data import
- File system for storing CSV files used by the `ADB_CSVREAD` function
- File system where audit trail storage directory will be created

Configure the shared file system on a disk (shared disk) that can be referenced from both the active system and the standby system, and mount the shared file system in a directory with the same path in both the active system and the standby system. For details about shared disks, see *Shared disk configuration* in the manual *HA Monitor for Linux(R) (x86)*.

In Figure 17-1: Example of a system configuration using the cold standby configuration, FS005 is the shared file system.

- FS005

  File system for storing input data files for data import, file system for storing CSV files, and file system for storing the audit trail storage directory

For the shared file system, you can use, for example, an NFS, which is a distributed file system.

(4) Disks for DB area files

Provide a disk for each DB area file.

For the DB area files listed below, allocate a block special file.

- Master directory DB area file
- Dictionary DB area file
- System-table DB area file
- Data DB area file

Ensure that the disks for the following DB area files can be referenced from both the active system and the standby system:

- Master directory DB area file
- Dictionary DB area file
- System-table DB area file
- Data DB area file
- Work table DB area file

Important

When using SCSI reservation for shared disk, configure the disks that store the preceding DB area files as subject to SCSI reservation.

In Figure 17-1: Example of a system configuration using the cold standby configuration, LU001 to LU005 and WRK001 are the disks for DB area files.

- LU001

  DB area disk that comprises a dictionary DB area
- LU002

  DB area disk that comprises a master directory DB area
- LU003

  DB area disk that comprises a system-table DB area
- LU004

  DB area disk that comprises a data DB area (ADBUTBL01)
- LU005

  DB area disk that comprises a data DB area (ADBUIDX01)

- WRK001
  DB area disk that comprises a work table DB area (ADBWRK)

17.3 Building a system with the cold standby configuration

This section explains how to build a system with the cold standby configuration.

17.3.1 Installing HA Monitor

Install HA Monitor in the active system and the standby system. For details about how to install HA Monitor, see *System Configuration* in the manual *HA Monitor for Linux(R) (x86)*.

17.3.2 Installing HADB server and setting up an environment

Install HADB servers in the active system and the standby system and set up their environments. For details about how to install the HADB server and set up the environment, see 8. Building a System.

The following notes apply to HADB server installation and environment setup:

- Use the same version and revision for the HADB servers in the active system and the standby system.
- For details about how to set the environment variables, see 8.4 Setting environment variables. Note that you must specify the same values in the active system and the standby system for the following environment variables:
  - `ADBLANG`
  - `TZ`
- If the HADB server in the active system terminates abnormally in the cold standby configuration, that HADB server is restarted, but only after troubleshooting information has been collected. You must create directories for storing the troubleshooting information in the active system and the standby system.

  In the following example, directory `/HADB/adbinfo` is created in the host's local file system:

```
mkdir -p /HADB/adbinfo                    ...1
chmod 755 /HADB/adbinfo                   ...2
```

  1. The HADB administrator uses the OS's `mkdir` command to create a directory.
  2. The OS's `chmod` command grants the privileges that are needed so that the HADB administrator can write data into this directory.

  Files created in the directory for storing troubleshooting-information files are not deleted automatically. You must delete them manually once the information is no longer needed.

> **Note**
>
> The HADB server's server directory (during installation and during operation) does not need to be the same in the active system and the standby system.

17.3.3 Installing HADB client and setting up the environment

For details about installing and setting up the environment for the HADB client, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

17.3.4 Setting up an HA Monitor environment

Set up an HA Monitor environment in the active system and the standby system. For details about how to set up an HA Monitor environment, see *System Configuration* in the manual *HA Monitor for Linux(R) (x86)*.

This subsection provides notes on setting up an HA Monitor environment.

(1) Setting environment variables

Set the following HADB administrator environment variable:

- `PATH`

  Add the following path so that HA Monitor commands can be executed:
  `/opt/hitachi/HAmon/bin`

(2) Specifying settings in the sysdef file

Among the operands that you need to specify for HA Monitor's `sysdef` file, the values to specify are predetermined for the following operands:

- `address`

  Specifies the order of priority for resetting hosts. The smaller the specification value, the higher the priority of resetting that host. Specify different values for the individual hosts. The value for the active system must be smaller than the value for the standby system.
- `monbegin_restart`

  Specify `nouse`.
- `termcmd_at_abort`

  Specify `nouse`.
- `fence_reset`

  When using host reset, either omit this operand or specify the default value (`use`).

  When using SCSI reservation for shared disk, specify `nouse`.
- `fence_scsi`

  When using host reset, either omit this operand or specify the default value (`nouse`).

  When using SCSI reservation for shared disk, specify `use`.
- `fence_lan`

  Either omit this operand or specify the default value (`nouse`).

For details about the above operands, see *HA Monitor environment settings (sysdef)* in the manual *HA Monitor for Linux(R) (x86)*.

(3) Specifying the servers file

Among the operands that you need to specify for HA Monitor's `servers` file, the values to specify are predetermined for the following operands:

- `name`

  Specify the absolute path for the server directory.

Note that you cannot specify a character string that includes spaces for the `name` operand as a path name. Therefore, in a cold standby configuration, we recommend that you use as the server directory a directory whose path name does not include spaces.

- `acttype`

  Specify `monitor` because the cold standby configuration is run in HA Monitor's monitor mode.

- `termcommand`

  Specify the absolute path for the server termination command that describes the settings for HADB.

  Create the server termination command based on the appropriate template. For details about how to create a server termination command, see (c) Creating a server termination command in (4) Creating environment variable definitions for commands, and creating commands, and *Creating a server termination command* in the manual *HA Monitor for Linux(R) (x86)*.

- `initial`

  Specify `online` for the active system and `standby` for the standby system.

- `disk`

  Specify the absolute path for VGs that include the following file systems:

  - File system that stores the DB directory
  - File system that stores temporary work files
  - File system that stores synonym dictionary files (not needed if you will not be conducting synonym searches)
  - File system where the audit trail directory will be created (not needed if the audit trail facility will not be used)

  In the case of Figure 17-1: Example of a system configuration using the cold standby configuration, specify `/dev/vg_hadb01`, `/dev/vg_hadb02`, `/dev/vg_hadb03`, and `/dev/vg_hadb04`.

- `fs_name`

  Specify the absolute path for logical volumes that correspond to the following file systems:

  - File system that stores the DB directory
  - File system that stores temporary work files
  - File system that stores synonym dictionary files (not needed if you will not be conducting synonym searches)
  - File system where the audit trail directory will be created (not needed if the audit trail facility will not be used)

  In the case of Figure 17-1: Example of a system configuration using the cold standby configuration, specify `/dev/vg_hadb01/hadb_db`, `/dev/vg_hadb02/hadb_workarea`, `/dev/vg_hadb03/hadb_syndict`, and `/dev/vg_hadb04/hadb_audit`.

- `fs_mount_dir`

  Specify the directory path in which the following file systems are mounted. This path must be the same on both the active system and the standby system.

  - File system that stores the DB directory
  - File system that stores temporary work files
  - File system that stores synonym dictionary files (not needed if you will not be conducting synonym searches)
  - File system where the audit trail directory will be created (not needed if the audit trail facility will not be used)

- `fs_mount_opt`

  Specify mount options for mounting the following file systems:

  - File system that stores the DB directory
  - File system that stores temporary work files

- File system that stores synonym dictionary files (not needed if you will not be conducting synonym searches)
- File system where the audit trail directory will be created (not needed if the audit trail facility will not be used)

- `lan_updown`

  Specify `use` because the LAN status settings files are used.

  Configure an alias IP address in the LAN status settings files. For details, see *Specifying LAN status settings files* in the manual *HA Monitor for Linux(R) (x86)*.

- `patrolcommand`

  Specify the absolute path for the server-monitoring command that describes the settings for HADB.

  Create the server-monitoring command based on the appropriate template. For details about how to create a server-monitoring command, see (d) Creating a server-monitoring command in (4) Creating environment variable definitions for commands, and creating commands, and *Creating a server monitoring command* in the manual *HA Monitor for Linux(R) (x86)*.

- `servexec_retry`

  Specify `2`.

- `waitserv_exec`

  Specify `yes`.

- `ip_neck`

  Specify `use`.

- `uoc_neck`

  Specify `nouse`.

- `vg_neck`

  Specify `use` for the VG that contains the file system for the DB directory and the file system where the audit trail directory will be created.

  Specify `nouse` for any other VG.

- `fs_neck`

  Specify `use` for the file system that contains the DB directory and the file system where the audit trail directory will be created.

  Specify `nouse` for any other file system.

- `actcommand`

  Specify the absolute path for the server startup command that describes the settings for HADB.

  Create the server startup command based on the appropriate template. For details about how to create a server startup command, see (b) Creating a server startup command in (4) Creating environment variable definitions for commands, and creating commands, and *Creating a server start command* in the manual *HA Monitor for Linux(R) (x86)*.

- `scsi_device`

  In this operand, specify the absolute paths of the device names that are to be subject to SCSI reservation.

  When using host reset, you do not need to specify this operand.

  Specify this operand when you are using SCSI reservation for shared disk and the shared disk meets any of the following conditions:

  - The shared disk is in a single-path configuration.
  - The shared disk is in a VMware ESXi virtual environment (excluding situations where DMMP is used).
  - The shared disk is in a redundant configuration realized using multipath software (HDLM).

When specifying this operand, specify the device names of the following file systems and disks by absolute path:

- File systems subject to host switchover processing

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `FS001` to `FS004` are file systems subject to host switchover processing.

- Disks for DB area files

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `LU001` to `LU005` and `WRK001` are disks for DB area files.

If you do not intend to perform synonym searches, you do not need to specify the device name of the file system where the synonym dictionary files are stored. If you do not intend to use the audit trail facility, you do not need to specify the device name of the file system where the audit trail directory will be created.

Important

Do not specify the device name of one of the following file systems in this operand.

- Host's local file system

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `LOC001` and `LOC002` are the host's local file systems.

- Shared file system

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `FS005` is the shared file system.

When specifying this operand, see *Determining the operand value needed for SCSI reservation for shared disk* in the manual *HA Monitor for Linux(R) (x86)*.

- `dmmp_device`

In this operand, specify the absolute paths of the logical DMMP devices that are to be subject to SCSI reservation.

When using host reset, you do not need to specify this operand.

Specify this operand when you are using SCSI reservation for shared disk, and the shared disk is in a redundant configuration based on multipath software (DMMP).

When specifying this operand, specify the device names of the following file systems and disks by absolute path:

- File systems subject to host switchover processing

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `FS001` to `FS004` are file systems subject to host switchover processing.

- Disks for DB area files

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `LU001` to `LU005` and `WRK001` are disks for DB area files.

If you do not intend to perform synonym searches, you do not need to specify the device name of the file system where the synonym dictionary files are stored. If you do not intend to use the audit trail facility, you do not need to specify the device name of the file system where the audit trail directory will be created.

Important

Do not specify the device name of one of the following file systems in this operand.

- Host's local file system

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `LOC001` and `LOC002` are the host's local file systems.

- Shared file system

  In Figure 17-1: Example of a system configuration using the cold standby configuration, `FS005` is the shared file system.

When specifying this operand, see *Determining the operand value needed for SCSI reservation for shared disk* in the manual *HA Monitor for Linux(R) (x86)*.

For details about the preceding operands specified in the `servers` file, see *Server environment definition (servers)* in the manual *HA Monitor for Linux(R) (x86)*.

(4) Creating environment variable definitions for commands, and creating commands

To link to HA Monitor, create environment variable definitions for commands, and create the following commands:

- Server startup command
- Server termination command
- Server-monitoring command

Templates for environment variable definitions for commands, as well as the above commands, are located under `$ADBDIR/sample/scripts`. Create the commands based on the templates.

(a) Creating environment variable definitions for commands

`$ADBDIR/sample/scripts/coldstandby.env` is the template. Copy this template and correct the following locations:

- Environment variable `ADBMGR`

  Specify the user name of the HADB administrator (OS user).

(b) Creating a server startup command

`$ADBDIR/sample/scripts/coldstandby_act.sh` is the template. Copy this template and correct the following location:

- `source` statement

  Specify for the argument of the `source` statement the absolute path for the file that specifies the environment variable definitions for commands.

(c) Creating a server termination command

`$ADBDIR/sample/scripts/coldstandby_term.sh` is the template. Copy this template and correct the following location:

- `source` statement

  Specify for the argument of the `source` statement the absolute path for the file that specifies the environment variable definitions for commands.

(d) Creating a server-monitoring command

`$ADBDIR/sample/scripts/coldstandby_patrol.sh` is the template. Copy this template and correct the following location:

- `source` statement

  Specify for the argument of the `source` statement the absolute path for the file that specifies the environment variable definitions for commands.

(5) File specification examples (when using host reset)

This subsection provides specification examples for the following files. The following are specification examples when you are using host reset.

- `sysdef` file
- `servers` file
- File for environment variable definitions for commands (`/HADB/scripts/coldstandby.env` in this example)
- Server startup command file (`/HADB/scripts/coldstandby_act.sh` in this example)
- Server termination command file (`/HADB/scripts/coldstandby_term.sh` in this example)
- Server-monitoring command file (`/HADB/scripts/coldstandby_patrol.sh` in this example)

The specification examples explained here assume the system configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration.

(a) sysdef file specification examples

■ Specification example of the `sysdef` file for server machine `hadb01` (active system)

```
environment  name hadb01,
             address 1,                  ...1
             patrol 60,
             lan path11,                 ...2
             lanport HAmon1;             ...3
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             termcmd_at_abort nouse;
```

Explanation

1. Specifies the order of priority for resetting hosts. Because this host is the active system, its reset priority needs to be the highest, so `1` is specified.
2. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path11`, which corresponds to IP address `172.16.0.11`.
3. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.

■ Specification example of the `sysdef` file for server machine `hadb02` (standby system)

```
environment  name hadb02,
             address 2,                  ...1
             patrol 60,
             lan path21,                 ...2
```

```
            lanport HAmon1;             ...3
function    pathpatrol 240,
            connect_retry 5:200,
            monbegin_restart nouse,
            termcmd_at_abort nouse;
```

Explanation

1. Specifies the order of priority for resetting hosts. This host is the standby system, so specify `2`.
2. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path21`, which corresponds to IP address `172.16.0.12`.
3. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.

(b) servers file specification examples

The examples described here assume that you are using ext4 as the file system. If you are using a file system other than ext4, the OS `mount` command options will differ from those in the examples shown here.

■ Specification example of the `servers` file for server machine `hadb01` (active system)

```
server name  /HADB/server,                      ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb01:                    ...2
            /dev/vg_hadb02:
            /dev/vg_hadb03:
            /dev/vg_hadb04,
       lan_updown use,
       fs_name /dev/vg_hadb01/hadb_db:         ...3
               /dev/vg_hadb02/hadb_workarea:
               /dev/vg_hadb03/hadb_syndict:
               /dev/vg_hadb04/hadb_audit,
       fs_mount_dir /HADB/db:                  ...4
                    /mnt/workarea:
                    /mnt/syndict:
                    /mnt/audit,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":    ...5
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/coldstandby_act.sh",         ...6
       termcommand "/HADB/scripts/coldstandby_term.sh",       ...7
       patrolcommand "/HADB/scripts/coldstandby_patrol.sh",   ...8
       servexec_retry 2,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:nouse:nouse:use,   ...9
       fs_neck use:nouse:nouse:use,   ...10
       initial online;                ...11
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - DB directory
   - Directory that stores temporary work files

- The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
- Audit trail directory (this directory is created in order to use the audit trail facility)

3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used in the cold standby configuration.
7. Specify the absolute path for the server termination command to be used in the cold standby configuration.
8. Specify the absolute path for the server-monitoring command to be used in the cold standby configuration.
9. Specify as follows:
   - Specify `use` for the VG that contains the file system where the DB directory will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores temporary work files will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores synonym dictionary files will be created.
   - Specify `use` for the VG that contains the file system where the audit trail directory will be created.
10. Specify as follows:
    - Specify `use` for the file system where the DB directory will be created.
    - Specify `nouse` for the file system where the directory that stores temporary work files will be created.
    - Specify `nouse` for the file system where the directory that stores synonym dictionary files will be created.
    - Specify `use` for the file system where the audit trail directory will be created.
11. Specify `online` to make this system the active system.

■ **Specification example of the `servers` file for server machine `hadb02` (standby system)**

```
server name  /HADB/server,                        ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb01:                       ...2
            /dev/vg_hadb02:
            /dev/vg_hadb03:
            /dev/vg_hadb04,
       lan_updown use,
       fs_name /dev/vg_hadb01/hadb_db:            ...3
               /dev/vg_hadb02/hadb_workarea:
               /dev/vg_hadb03/hadb_syndict:
               /dev/vg_hadb04/hadb_audit,
       fs_mount_dir /HADB/db:                     ...4
                    /mnt/workarea:
                    /mnt/syndict:
                    /mnt/audit,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":    ...5
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/coldstandby_act.sh",         ...6
       termcommand "/HADB/scripts/coldstandby_term.sh",       ...7
       patrolcommand "/HADB/scripts/coldstandby_patrol.sh",   ...8
       servexec_retry 2,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
```

```
        vg_neck use:nouse:nouse:use,    ...9
        fs_neck use:nouse:nouse:use,    ...10
        initial standby;                ...11
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - DB directory
   - Directory that stores temporary work files
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
   - Audit trail directory (this directory is created in order to use the audit trail facility)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used in the cold standby configuration.
7. Specify the absolute path for the server termination command to be used in the cold standby configuration.
8. Specify the absolute path for the server-monitoring command to be used in the cold standby configuration.
9. Specify as follows:
   - Specify `use` for the VG that contains the file system where the DB directory will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores temporary work files will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores synonym dictionary files will be created.
   - Specify `use` for the VG that contains the file system where the audit trail directory will be created.
10. Specify as follows:
    - Specify `use` for the file system where the DB directory will be created.
    - Specify `nouse` for the file system where the directory that stores temporary work files will be created.
    - Specify `nouse` for the file system where the directory that stores synonym dictionary files will be created.
    - Specify `use` for the file system where the audit trail directory will be created.
11. Specify `standby` to make this system the standby system.

(c) Specification example of a file for environment variable definitions for commands

In this example, environment variable definitions for commands are stored in `/HADB/scripts/coldstandby.env`. Specify the same settings for the active system and the standby system.

■ Specification example of the file for environment variable definitions for commands

```
# The environment variables for HADB
export ADBMGR=adbmanager                    ...1
export ADBDIR=/HADB/server                  ...2
export ADBINFODIR=/HADB/adbinfo             ...3
```

Explanation

1. Specify user name `adbmanager` for the HADB administrator (OS user).
2. Specify the absolute path for the server directory.
3. Specify the absolute path for the directory for storing troubleshooting-information files.

(d) Specification example of the server startup command file

In this example, the server startup command is stored in `/HADB/scripts/coldstandby_act.sh`. Specify the same settings for the active system and the standby system.

■ Specification example of the server startup command file

```
#!/bin/sh

# Sample of the actcommand for HADB

SU=/bin/su
ECHO=/bin/echo

# Setting environment variables for HADB
source /HADB/scripts/coldstandby.env       ...1

# Execute adbstart command
$SU - $ADBMGR -c "$ECHO y|$ADBDIR/bin/adbstart"

# always 0 return
exit 0
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/coldstandby.env`).

(e) Specification example of the server termination command file

In this example, the server termination command is stored in `/HADB/scripts/coldstandby_term.sh`. Specify the same settings for the active system and the standby system.

■ Specification example of the server termination command file

```
#!/bin/sh

# Sample of the termcommand for HADB

STS_ACTIVE="ACTIVE"
STS_STOP="STOP"
STS_STARTING="STARTING"
STS_STOPPING="STOPPING"
STS_ABORT="ABORT"
STS_QUIESCE="QUIESCE"
STS_OFFLINE="OFFLINE"
STS_CHGMODE="CHGMODE"
STS_STOPWAIT="STOPWAIT"
STS_COREDUMP="COREDUMP"
STS_FORCE="FORCE"
STS_MAINTNCE="MAINTNCE"

AWK=/bin/awk
ECHO=/bin/echo
PS=/bin/ps
KILL=/bin/kill
GREP=/bin/grep
```

```
SU=/bin/su
READLINK=/bin/readlink

# Setting environment variables for HADB
source /HADB/scripts/coldstandby.env       ...1

# Resolve the path for adbsrvd
ADBSRVD=`$READLINK -e $ADBDIR/bin/adbsrvd`
if [ $? -ne 0 ]
then
  exit 0
fi

# Execute adbstop command
GET_STS=`$SU - $ADBMGR -c "$ADBDIR/bin/adbls -d srv 2>/dev/null" | $GREP -v SVID |
 $AWK '{ if ($1 ~ /[0-9]+/) { print $2 } else { print $1 } }'`

case "$1" in
"-e" )
  # Normal stop operations of active system.
  # (when the monend command is executed.)
  if [ "$GET_STS" = "$STS_ACTIVE"   -o "$GET_STS" = "$STS_STARTING" -o \
       "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
       "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
       "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_MAINTNCE" ]
  then
    $SU - $ADBMGR -c "$ECHO y|$ADBDIR/bin/adbstop --force"
    STOPRES=$?
    if [ $STOPRES -ne 0 -a $STOPRES -ne 4 ]
    then
      ADB_ID=`$SU - $ADBMGR -c "$PS x" | $GREP $ADBSRVD | $GREP -v $GREP | $AWK '{
print $1}'`
      if [ "$ADB_ID" != "" ]
      then
        $SU - $ADBMGR -c "$KILL $ADB_ID"
      fi
    fi
  fi

  ;;
"-w" )
  # Plan stop operations of active system.
  # (when the monswap command is executed.)
  if [ "$GET_STS" = "$STS_ACTIVE"   -o "$GET_STS" = "$STS_STARTING" -o \
       "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
       "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
       "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_MAINTNCE" ]
  then
    $SU - $ADBMGR -c "$ECHO y|$ADBDIR/bin/adbstop --force"
    STOPRES=$?
    if [ $STOPRES -ne 0 -a $STOPRES -ne 4 ]
    then
      ADB_ID=`$SU - $ADBMGR -c "$PS x" | $GREP $ADBSRVD | $GREP -v $GREP | $AWK '{
print $1}'`
      if [ "$ADB_ID" != "" ]
      then
        $SU - $ADBMGR -c "$KILL $ADB_ID"
      fi
    fi
  fi

  ;;
"-c" )
  # Retry operations of active system.
```

```
  # (Before the act_command is resterted.)
  $SU - $ADBMGR -c "adbinfoget -l -o $ADBINFODIR"

  ;;
esac

# stop HADB
if [ "$GET_STS" = "$STS_STARTING" ]
then
  ADB_ID=`$SU - $ADBMGR -c "$PS x" | $GREP $ADBSRVD | $GREP -v $GREP | $AWK '{prin
t $1}'`
  $SU - $ADBMGR -c "$KILL $ADB_ID"

  exit 0
fi

# Wait for end of HADB
while [ "$GET_STS" = "$STS_ACTIVE"   -o "$GET_STS" = "$STS_STARTING" -o \
        "$GET_STS" = "$STS_STOPPING" -o "$GET_STS" = "$STS_QUIESCE"  -o \
        "$GET_STS" = "$STS_OFFLINE"  -o "$GET_STS" = "$STS_CHGMODE"  -o \
        "$GET_STS" = "$STS_STOPWAIT" -o "$GET_STS" = "$STS_COREDUMP" -o \
        "$GET_STS" = "$STS_MAINTNCE" ]
do
  GET_STS=`$SU - $ADBMGR -c "$ADBDIR/bin/adbls -d srv 2>/dev/null" | $GREP -v SVID
 | $AWK '{ if ($1 ~ /[0-9]+/) { print $2 } else { print $1 } }'`
  sleep 1
done

exit 0
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/coldstandby.env`).

(f) Specification example of the server-monitoring command file

In this example, the server-monitoring command is stored in `/HADB/scripts/coldstandby_patrol.sh`. Specify the same settings for the active system and the standby system.

■ Specification example of the server-monitoring command file

```
#!/bin/sh

# Sample of the patrolcommand for HADB

AWK=/bin/awk
GREP=/bin/grep
PS=/bin/ps
PGREP=/usr/bin/pgrep
PKILL=/usr/bin/pkill
SU=/bin/su
READLINK=/bin/readlink

# Setting environment variables for HADB
source /HADB/scripts/coldstandby.env       ...1

# Resolve the path for adbsrvd
ADBSRVD=`$READLINK -e $ADBDIR/bin/adbsrvd`
if [ $? -ne 0 ]
then
  exit 0
fi
```

```
# Get adbsrvd process id
CHKPID=`$SU - $ADBMGR -c "$PS x" | $GREP $ADBSRVD | $GREP -v $GREP | $AWK '{print
$1}'`
if [ "$CHKPID" = "" ]
then
  exit 0
fi

# Execute adbmonitor command for active system
$SU - $ADBMGR -c "$ADBDIR/bin/adbmonitor -n" &
PID=$!

# Trap SIGTERM and terminate adbmonitor
trap "$PKILL -P $PID" 15

## Wait process terminated
CPID=""
CHKPID=`$PS aux | $AWK -v PID=$PID '{ if ($2 == PID) { print $2 } }'`
while [ "$CHKPID" != "" ]
do

  ## Get adbmonitor process id
  if [ "$CPID" = "" ]
  then
    CPID=`$PGREP -P $PID`
  fi

  ## Wait
  if [ "$CPID" != "" ]
  then
    CHKCPID=`$PS aux | $AWK -v CPID=$CPID '{ if ($2 == CPID) { print $2 } }'`
    while [ "$CHKCPID" != "" ]
    do
      sleep 1
      CHKCPID=`$PS aux | $AWK -v CPID=$CPID '{ if ($2 == CPID) { print $2 } }'`
    done
  fi

  sleep 1
  CHKPID=`$PS aux | $AWK -v PID=$PID '{ if ($2 == PID) { print $2 } }'`
done

exit 0
```

Explanation

1. For the `source` statement, specify the absolute path for the file for environment variable definitions for commands (`/HADB/scripts/coldstandby.env`).

(6) File specification examples (when using SCSI reservation for shared disk)

This subsection provides specification examples for the following files. The following are specification examples when you are using SCSI reservation for shared disk.

- `sysdef` file
- `servers` file
- File for environment variable definitions for commands (`/HADB/scripts/coldstandby.env` in this example)

- Server startup command file (`/HADB/scripts/coldstandby_act.sh` in this example)
- Server termination command file (`/HADB/scripts/coldstandby_term.sh` in this example)
- Server-monitoring command file (`/HADB/scripts/coldstandby_patrol.sh` in this example)

The specification examples explained here assume the system configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration.

(a) sysdef file specification examples

■ Specification example of the `sysdef` file for server machine `hadb01` (active system)

```
environment  name hadb01,
             address 1,                 ...1
             patrol 60,
             lan path11,                ...2
             lanport HAmon1;            ...3
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             termcmd_at_abort nouse,
             fence_reset nouse,         ...4
             fence_scsi use,            ...5
             fence_lan nouse;
```

Explanation

1. Specifies the order of priority for resetting hosts. Because this host is the active system, its reset priority needs to be the highest, so `1` is specified.
2. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path11`, which corresponds to IP address `172.16.0.11`.
3. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
4. Because host reset will not be used, specify `nouse`.
5. Because SCSI reservation for shared disk will be used, specify `use`.

■ Specification example of the `sysdef` file for server machine `hadb02` (standby system)

```
environment  name hadb02,
             address 2,                 ...1
             patrol 60,
             lan path21,                ...2
             lanport HAmon1;            ...3
function     pathpatrol 240,
             connect_retry 5:200,
             monbegin_restart nouse,
             termcmd_at_abort nouse,
             fence_reset nouse,         ...4
             fence_scsi use,            ...5
             fence_lan nouse;
```

Explanation

1. Specifies the order of priority for resetting hosts. This host is the standby system, so specify `2`.
2. Specify the host name of the LAN that is used for HA Monitor's monitoring path. Specify host name `path21`, which corresponds to IP address `172.16.0.12`.

3. Specify the service name of the LAN that is used for HA Monitor's monitoring path. Specify service name `HAmon1`, which corresponds to port number `7777`.
4. Because host reset will not be used, specify `nouse`.
5. Because SCSI reservation for shared disk will be used, specify `use`.

(b) servers file specification examples

The following describes specification examples of a `servers` file in the following cases:

- The shared disk is in a single-path configuration
- A redundant configuration is used with multipath software (DMMP).

The examples described here assume that you are using ext4 as the file system. If you are using a file system other than ext4, the OS `mount` command options will differ from those in the examples shown here.

■ Specification example of the `servers` file in a single-path configuration

▪ Specification example of the `servers` file for server machine `hadb01` (active system)

```
server name  /HADB/server,                          ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb01:                        ...2
            /dev/vg_hadb02:
            /dev/vg_hadb03:
            /dev/vg_hadb04,
       lan_updown use,
       fs_name /dev/vg_hadb01/hadb_db:             ...3
               /dev/vg_hadb02/hadb_workarea:
               /dev/vg_hadb03/hadb_syndict:
               /dev/vg_hadb04/hadb_audit,
       fs_mount_dir /HADB/db:                      ...4
                    /mnt/workarea:
                    /mnt/syndict:
                    /mnt/audit,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":    ...5
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/coldstandby_act.sh",         ...6
       termcommand "/HADB/scripts/coldstandby_term.sh",       ...7
       patrolcommand "/HADB/scripts/coldstandby_patrol.sh",   ...8
       servexec_retry 2,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:nouse:nouse:use,                                          ...9
       fs_neck use:nouse:nouse:use,                                          ...10
       scsi_device /dev/disk/by-id/wwn-0x60060e8010205850051104c500000005:   ...11
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c500000006:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c500000007:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c500000008:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000f:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000e:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c500000010:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c500000011:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c500000013:
                   /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000d,
       initial online;                                                       ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - DB directory
   - Directory that stores temporary work files
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
   - Audit trail directory (this directory is created in order to use the audit trail facility)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used in the cold standby configuration.
7. Specify the absolute path for the server termination command to be used in the cold standby configuration.
8. Specify the absolute path for the server-monitoring command to be used in the cold standby configuration.
9. Specify as follows:
   - Specify `use` for the VG that contains the file system where the DB directory will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores temporary work files will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores synonym dictionary files will be created.
   - Specify `use` for the VG that contains the file system where the audit trail directory will be created.
10. Specify as follows:
    - Specify `use` for the file system where the DB directory will be created.
    - Specify `nouse` for the file system where the directory that stores temporary work files will be created.
    - Specify `nouse` for the file system where the directory that stores synonym dictionary files will be created.
    - Specify `use` for the file system where the audit trail directory will be created.
11. Specify the absolute paths for the disks that contain file systems subject to host switchover processing and the absolute paths for the disks that contain the DB areas. Specify the absolute paths in the same order for the active system and the standby system.
12. Specify `online` to make this system the active system.

▪ **Specification example of the `servers` file for server machine `hadb02` (standby system)**

```
server name  /HADB/server,                        ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb01:                      ...2
            /dev/vg_hadb02:
            /dev/vg_hadb03:
            /dev/vg_hadb04,
       lan_updown use,
       fs_name /dev/vg_hadb01/hadb_db:           ...3
               /dev/vg_hadb02/hadb_workarea:
               /dev/vg_hadb03/hadb_syndict:
               /dev/vg_hadb04/hadb_audit,
       fs_mount_dir /HADB/db:                    ...4
```

```
                /mnt/workarea:
                /mnt/syndict:
                /mnt/audit,
        fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":    ...5
                "-t ext4 -o defaults,noatime,_netdev":
                "-t ext4 -o defaults,noatime,_netdev":
                "-t ext4 -o defaults,noatime,_netdev",
        actcommand "/HADB/scripts/coldstandby_act.sh",         ...6
        termcommand "/HADB/scripts/coldstandby_term.sh",       ...7
        patrolcommand "/HADB/scripts/coldstandby_patrol.sh",   ...8
        servexec_retry 2,
        waitserv_exec yes,
        ip_neck use,
        uoc_neck nouse,
        vg_neck use:nouse:nouse:use,                                        ...9
        fs_neck use:nouse:nouse:use,                                        ...10
        scsi_device /dev/disk/by-id/wwn-0x60060e8010205850051104c500000005:   ...11
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c500000006:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c500000007:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c500000008:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000f:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000e:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c500000010:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c500000011:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c500000013:
                    /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000d,
        initial standby;                                                    ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - DB directory
   - Directory that stores temporary work files
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
   - Audit trail directory (this directory is created in order to use the audit trail facility)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used in the cold standby configuration.
7. Specify the absolute path for the server termination command to be used in the cold standby configuration.
8. Specify the absolute path for the server-monitoring command to be used in the cold standby configuration.
9. Specify as follows:
   - Specify `use` for the VG that contains the file system where the DB directory will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores temporary work files will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores synonym dictionary files will be created.
   - Specify `use` for the VG that contains the file system where the audit trail directory will be created.
10. Specify as follows:

- Specify `use` for the file system where the DB directory will be created.
- Specify `nouse` for the file system where the directory that stores temporary work files will be created.
- Specify `nouse` for the file system where the directory that stores synonym dictionary files will be created.
- Specify `use` for the file system where the audit trail directory will be created.

11. Specify the absolute paths for the disks that contain file systems subject to host switchover processing and the absolute paths for the disks that contain the DB areas. Specify the absolute paths in the same order for the active system and the standby system.
12. Specify `standby` to make this system the standby system.

■ **Specification example of the `servers` file in a redundant configuration using multipath software (DMMP)**

▪ **Specification example of the `servers` file for server machine `hadb01` (active system)**

```
server name  /HADB/server,                        ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb01:                      ...2
            /dev/vg_hadb02:
            /dev/vg_hadb03:
            /dev/vg_hadb04,
       lan_updown use,
       fs_name /dev/vg_hadb01/hadb_db:           ...3
               /dev/vg_hadb02/hadb_workarea:
               /dev/vg_hadb03/hadb_syndict:
               /dev/vg_hadb04/hadb_audit,
       fs_mount_dir /HADB/db:                    ...4
                    /mnt/workarea:
                    /mnt/syndict:
                    /mnt/audit,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":    ...5
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev",
       actcommand "/HADB/scripts/coldstandby_act.sh",         ...6
       termcommand "/HADB/scripts/coldstandby_term.sh",       ...7
       patrolcommand "/HADB/scripts/coldstandby_patrol.sh",   ...8
       servexec_retry 2,
       waitserv_exec yes,
       ip_neck use,
       uoc_neck nouse,
       vg_neck use:nouse:nouse:use,       ...9
       fs_neck use:nouse:nouse:use,       ...10
       dmmp_device /dev/mapper/mpath1:    ...11
                   /dev/mapper/mpath2:
                   /dev/mapper/mpath3:
                   /dev/mapper/mpath4:
                   /dev/mapper/mpath11:
                   /dev/mapper/mpath12:
                   /dev/mapper/mpath13:
                   /dev/mapper/mpath14:
                   /dev/mapper/mpath15:
                   /dev/mapper/mpath16,
       initial online;                    ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:

- DB directory
- Directory that stores temporary work files
- The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
- Audit trail directory (this directory is created in order to use the audit trail facility)

3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used in the cold standby configuration.
7. Specify the absolute path for the server termination command to be used in the cold standby configuration.
8. Specify the absolute path for the server-monitoring command to be used in the cold standby configuration.
9. Specify as follows:
   - Specify `use` for the VG that contains the file system where the DB directory will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores temporary work files will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores synonym dictionary files will be created.
   - Specify `use` for the VG that contains the file system where the audit trail directory will be created.
10. Specify as follows:
    - Specify `use` for the file system where the DB directory will be created.
    - Specify `nouse` for the file system where the directory that stores temporary work files will be created.
    - Specify `nouse` for the file system where the directory that stores synonym dictionary files will be created.
    - Specify `use` for the file system where the audit trail directory will be created.
11. Specify the absolute paths for the disks that contain file systems subject to host switchover processing and the absolute paths for the disks that contain the DB areas. Specify the absolute paths in the same order for the active system and the standby system.
12. Specify `online` to make this system the active system.

▪ **Specification example of the `servers` file for server machine `hadb02` (standby system)**

```
server name  /HADB/server,                        ...1
       alias HADB,
       acttype monitor,
       disk /dev/vg_hadb01:                      ...2
            /dev/vg_hadb02:
            /dev/vg_hadb03:
            /dev/vg_hadb04,
       lan_updown use,
       fs_name /dev/vg_hadb01/hadb_db:           ...3
               /dev/vg_hadb02/hadb_workarea:
               /dev/vg_hadb03/hadb_syndict:
               /dev/vg_hadb04/hadb_audit,
       fs_mount_dir /HADB/db:                    ...4
                    /mnt/workarea:
                    /mnt/syndict:
                    /mnt/audit,
       fs_mount_opt "-t ext4 -o defaults,noatime,_netdev":    ...5
                    "-t ext4 -o defaults,noatime,_netdev":
                    "-t ext4 -o defaults,noatime,_netdev":
```

```
                "-t ext4 -o defaults,noatime,_netdev",
        actcommand "/HADB/scripts/coldstandby_act.sh",          ...6
        termcommand "/HADB/scripts/coldstandby_term.sh",        ...7
        patrolcommand "/HADB/scripts/coldstandby_patrol.sh",    ...8
        servexec_retry 2,
        waitserv_exec yes,
        ip_neck use,
        uoc_neck nouse,
        vg_neck use:nouse:nouse:use,        ...9
        fs_neck use:nouse:nouse:use,        ...10
        dmmp_device /dev/mapper/mpath1:     ...11
                    /dev/mapper/mpath2:
                    /dev/mapper/mpath3:
                    /dev/mapper/mpath4:
                    /dev/mapper/mpath11:
                    /dev/mapper/mpath12:
                    /dev/mapper/mpath13:
                    /dev/mapper/mpath14:
                    /dev/mapper/mpath15:
                    /dev/mapper/mpath16,
        initial standby;                    ...12
```

Explanation

1. Specify the absolute path for the server directory.
2. Specify the absolute paths of the VGs that contain the file systems in which the following directories will be created:
   - DB directory
   - Directory that stores temporary work files
   - The directory storing synonym dictionary files (a directory created in order to perform synonym searches)
   - Audit trail directory (this directory is created in order to use the audit trail facility)
3. Specify the absolute paths of the LVs where the file systems are set up that will store the directories in 2.
4. Specify the absolute paths for the mount points at which to mount the file systems for the directories in 2.
5. Specify options for the `mount` command to be used for mounting the file systems for the directories in 2.
6. Specify the absolute path for the server startup command to be used in the cold standby configuration.
7. Specify the absolute path for the server termination command to be used in the cold standby configuration.
8. Specify the absolute path for the server-monitoring command to be used in the cold standby configuration.
9. Specify as follows:
   - Specify `use` for the VG that contains the file system where the DB directory will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores temporary work files will be created.
   - Specify `nouse` for the VG that contains the file system where the directory that stores synonym dictionary files will be created.
   - Specify `use` for the VG that contains the file system where the audit trail directory will be created.
10. Specify as follows:
    - Specify `use` for the file system where the DB directory will be created.
    - Specify `nouse` for the file system where the directory that stores temporary work files will be created.
    - Specify `nouse` for the file system where the directory that stores synonym dictionary files will be created.

- Specify `use` for the file system where the audit trail directory will be created.

11. Specify the absolute paths for the disks that contain file systems subject to host switchover processing and the absolute paths for the disks that contain the DB areas. Specify the absolute paths in the same order for the active system and the standby system.
12. Specify `standby` to make this system the standby system.

(c) Specification example of a file for environment variable definitions for commands

For a specification example of a file for environment variable definitions for commands, see (c) Specification example of a file for environment variable definitions for commands in (5) File specification examples (when using host reset).

(d) Specification example of the server startup command file

For a specification example of the server startup command file, see (d) Specification example of the server startup command file in (5) File specification examples (when using host reset).

(e) Specification example of the server termination command file

For a specification example of the server stop command file, see (e) Specification example of the server termination command file in (5) File specification examples (when using host reset).

(f) Specification example of the server-monitoring command file

For a specification example of the server-monitoring command file, see (f) Specification example of the server-monitoring command file in (5) File specification examples (when using host reset).

(7) HA Monitor startup setting

Set up HA Monitor to start automatically when the operating system starts. For details about how to configure this setting, see *Automating the operation from system start through server start* in the manual *HA Monitor for Linux(R) (x86)*.

17.3.5 Creating server definitions on both HADB servers

When you use the cold standby configuration, you must create a server definition on the HADB server in the active system as well as on the HADB server in the standby system.

Important

- Ensure that the server definition settings are the same for both the HADB server in the active system and the HADB server in the standby system.
- The `adb_sys_multi_node_info` operand cannot be specified.

When you create server definitions, make sure that the `adb_blk_path_wrk` operand is specified. Specify for the `adb_blk_path_wrk` operand the path that is specified in the `adb_init_wrk_blk_path` initialization option in the `adbinit` command. For details, see 17.3.7 Creating a database.

17.3.6 Creating client definitions

The following shows a client definition specification example.

For details about how to create a client definition, see *Creating a client definition* in *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*. For details about the operands in a client definition, see *Contents of operands in the client definition* under *Designing Client Definitions* in the *HADB Application Development Guide*.

■ Client definition specification example

```
set adb_clt_rpc_srv_host = 10.196.108.143     ...1
set adb_clt_rpc_srv_port = 23650
set adb_clt_ap_name = "AP01"
```

Explanation

1. For the connection-destination host name, specify alias IP address `10.196.108.143`, which is used for client-server communication.

17.3.7 Creating a database

This subsection explains how to create databases in the cold standby configuration.

(1) General procedure for creating a database

The following shows the general procedure for creating a database:

1. Configure a file system for the DB directory (performed in the active system).
2. Mount the file system for the DB directory (performed in the active system).
3. Create a database (performed in the active system).
4. Unmount the file system for the DB directory (performed in the active system).
5. Mount the file system for the DB directory (performed in the standby system).
6. Unmount the file system for the DB directory (performed in the standby system).
7. Configure a file system for storing temporary work files (performed in the active system).
8. Mount and unmount the file system for storing temporary work files (performed in both the active and standby systems).
9. Build a file system for storing synonym dictionary files (performed in the active system)[#1].
10. Mount and unmount the file system for storing synonym dictionary files (performed in both the active and standby systems)[#1].
11. Build the file system where the audit trail directory will be created (performed in the active system)[#2].
12. Mount and unmount the file system where the audit trail directory will be created (performed in both the active and standby systems)[#2].
13. Configure the file systems that will be shared (performed in the active system).
14. Share the file systems (performed in both the active and standby systems).

#1

You must complete the aforementioned procedure in order to perform synonym searches.

#2

You must perform this procedure in order to use the audit trail facility.

(2) Configuring a file system for the DB directory (performed in the active system)

Perform this task as a superuser.

Configure a file system for the DB directory by executing an OS command in the active system.

The following execution example configures `/dev/vg_hadb01/hadb_db`, which has been created as an LV for the DB directory, in an ext4 file system.

■ Command execution example

```
mkfs -t ext4 /dev/vg_hadb01/hadb_db
```

(3) Mounting the file system for the DB directory (performed in the active system)

Perform this task as a superuser.

Mount the file system for the DB directory in the active system according to the following procedure.

Procedure:

1. Create a mount point for the DB directory.

   Execute an OS command to create the mount point for the DB directory.

   ■ Command execution example

   ```
   mkdir -p /HADB/db
   ```

2. Activate the VG that contains the file system for the DB directory.

   Execute an OS command to activate the VG that contains the file system for the DB directory.

   ■ Command execution example

   ```
   vgchange -a y /dev/vg_hadb01
   ```

3. Mount the file system for the DB directory.

   Execute an OS command to mount the file system for the DB directory.

   ■ Command execution example

   ```
   mount /dev/vg_hadb01/hadb_db /HADB/db -t ext4 -o defaults,noatime,_netdev
   ```

4. Change the owner of the DB directory.

   Execute an OS command to change the owner of the DB directory. Specify the HADB administrator (OS user) and the HADB administrators group that will be using the DB directory.

   The following example specifies `adbmanager` as the user name for the HADB administrator (OS user) and `adbgroup` as the HADB administrators group.

   ■ Command execution example

   ```
   chown adbmanager.adbgroup /HADB/db
   ```

(4) Creating a database (performed in the active system)

Perform this task as the HADB administrator.

Create a database in the active system according to the following procedure.

Procedure:

1. Verify that the file system for the DB directory is mounted at the mount point for the DB directory.
2. Execute the `adbinit` command to create a database.
   The following shows an example of specifying the initialization options in the `adbinit` command.
   ■ **Initialization option specification example** (in a single-path configuration)

```
set adb_init_dbarea_initialize = Y
set adb_init_wrk_blk_path = /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000d
set adb_init_mst_blk_path = /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000e
set adb_init_dic_blk_path = /dev/disk/by-id/wwn-0x60060e8010205850051104c50000000f
set adb_init_stbl_blk_path = /dev/disk/by-id/wwn-0x60060e8010205850051104c50000001
0
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/disk/by-id/wwn-0x60060e8010205850051104c500000011
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/disk/by-id/wwn-0x60060e8010205850051104c500000013
```

   ■ **Initialization option specification example** (in a redundant configuration using multipath software (DMMP))

```
set adb_init_dbarea_initialize = Y
set adb_init_wrk_blk_path = /dev/mapper/mpath16
set adb_init_mst_blk_path = /dev/mapper/mpath12
set adb_init_dic_blk_path = /dev/mapper/mpath11
set adb_init_stbl_blk_path = /dev/mapper/mpath13
adbinitdbarea -n ADBUTBL01 -i 2G \
              -v /dev/mapper/mpath14
adbinitdbarea -n ADBUIDX01 -i 2G \
              -v /dev/mapper/mpath15
```

(5) Unmounting the file system for the DB directory (performed in the active system)

Perform this task as a superuser.

Unmount the file system for the DB directory in the active system according to the following procedure.

Procedure:

1. Unmount the file system for the DB directory.
   Execute an OS command to unmount the file system for the DB directory.
   ■ Command execution example

```
umount /dev/vg_hadb01/hadb_db
```

2. Deactivate the VG that contains the file system for the DB directory.
   Execute an OS command on the active system to deactivate the VG that contains the file system for the DB directory.
   ■ Command execution example

```
vgchange -a n /dev/vg_hadb01
```

(6) Mounting the file system for the DB directory (performed in the standby system)

Perform this task as a superuser.

Similarly to the active system, mount the file system for the DB directory in the standby system. Create a mount point with the same path as in the active system.

For details about how to mount a file system, see (3) Mounting the file system for the DB directory (performed in the active system).

> **Important**
>
> In the standby system, you only create a mount point for the file system for the DB directory. You do not execute the `adbinit` command.

(7) Unmounting the file system for the DB directory (performed in the standby system)

Perform this task as a superuser.

Similarly to the active system, unmount the file system for the DB directory in the standby system.

For details about how to unmount a file system, see (5) Unmounting the file system for the DB directory (performed in the active system).

(8) Configuring a file system for storing temporary work files (performed in the active system)

Perform this task as a superuser.

Configure a file system for storing temporary work files by executing an OS command in the active system.

The following execution example builds `/dev/vg_hadb02/hadb_workarea`, which was created as the LV for storing temporary work files, as an ext4 file system.

■ **Command execution example**

```
mkfs -t ext4 /dev/vg_hadb02/hadb_workarea
```

(9) Mounting and unmounting the file system for storing temporary work files (performed in both the active and standby systems)

Perform this task as a superuser.

After you have mounted the file system for storing temporary work files in the active system, unmount it.

Also in the standby system, mount the file system for storing temporary work files, and then unmount it.

Create the mount points so that the paths are the same in both the active system and the standby system.

For details about how to mount a file system, see (3) Mounting the file system for the DB directory (performed in the active system). For details about how to unmount a file system, see (5) Unmounting the file system for the DB directory (performed in the active system).

(10) Building file systems for storing synonym dictionary files (performed in active systems)

To perform synonym searches, you must first follow this procedure.

The superuser must perform this procedure.

Execute OS commands in the active system, to build a file system for storing synonym dictionary files.

The following execution example builds `/dev/vg_hadb03/hadb_syndict`, which was created as the LV for storing synonym dictionary files, as an ext4 file system.

▪ Command execution example

```
mkfs -t ext4 /dev/vg_hadb03/hadb_syndict
```

(11) Mounting and unmounting file systems for storing synonym dictionary files (performed in both the active and standby systems)

To perform synonym searches, you must first follow this procedure.

The superuser must perform this procedure.

After mounting the file system for storing synonym dictionary files in the active system, the file system is unmounted.

Similarly, after mounting the file system for storing synonym dictionary files in the standby system, the file system is unmounted.

Create mount points so that the active and standby systems both use the same path.

For details about how to mount a file system, see (3) Mounting the file system for the DB directory (performed in the active system). For details about how to unmount a file system, see (5) Unmounting the file system for the DB directory (performed in the active system).

(12) Building the file system where the audit trail directory will be created (performed in the active system)

To use the audit trail facility, you must first follow this procedure.

Perform this task as a superuser.

Build the file system for the audit trail directory by executing an OS command in the active system.

The following execution example builds `/dev/vg_hadb04/hadb_audit`, which was created as the LV for outputting audit trail files, as an ext4 file system.

■ Command execution example

```
mkfs -t ext4 /dev/vg_hadb04/hadb_audit
```

(13) Mounting and unmounting the file system where the audit trail directory will be created (performed in both the active and standby systems)

To use the audit trail facility, you must first follow this procedure.

Perform this task as a superuser.

In the active system, mount and then unmount the file system where the audit trail directory will be created.

Similarly, in the standby system, mount and then unmount the file system where the audit trail directory will be created.

Create the mount points so that the paths are the same in both the active system and the standby system.

For details about how to mount a file system, see (3) Mounting the file system for the DB directory (performed in the active system). For details about how to unmount a file system, see (5) Unmounting the file system for the DB directory (performed in the active system).

(14) Configuring the file systems that will be shared (performed in the active system)

Perform this task as a superuser.

Share the following file systems between the active system and the standby system:

- File system for storing input data files that are used in data import processing
- File system for storing CSV files used by the `ADB_CSVREAD` function
- File system for the archive directory
- File system where audit trail storage directory will be created

Configure these file systems.

(15) Sharing the file systems (performed in both the active and standby systems)

Share between the active system and the standby system the file systems configured in (14) Configuring the file systems that will be shared (performed in the active system).

An example of sharing a file system by using an NFS server is below. Perform this task as a superuser.

Procedure:

1. Provide an NFS server and export the directories that you want to share between the active system and the standby system.
2. Mount the directories exported in step 1 so that the paths are the same between the active system and the standby system.

17.4 Starting and terminating the cold standby configuration

This section explains how to start and terminate the cold standby configuration.

Also see 10.2 Starting and terminating the HADB server and its operation modes.

17.4.1 How to start the cold standby configuration

(1) Procedure for starting the cold standby configuration

The following procedure explains how to start the cold standby configuration.

Procedure:

1. Confirm that HA Monitor is running in the active system and in the standby system.
   For the confirmation method, see *System Operation* in the manual *HA Monitor for Linux(R) (x86)*.
2. Execute HA Monitor's `monbegin` command in the active system and the standby system.

> **Note**
>
> When HA Monitor's `monbegin` command is executed, the `adbstart` command is executed automatically in the active system as part of the `monbegin` command's processing.

(2) Startup modes for the cold standby configuration

The following table lists and describes the startup modes supported for the cold standby configuration.

Table 17-4: Startup modes for the cold standby configuration

No.	Startup mode	Command to be executed	Explanation	Previous termination mode
1	Normal start	`monbegin`	This is the normal startup mode. In a normal start, database recovery processing does not take place.	The cold standby configuration terminated normally.
2	Restart		If the previous termination mode is as described on the right, the server will automatically restart. During the restart, the HADB server in the active system uses the system log to perform database recovery processing.	The cold standby configuration terminated abnormally or forcibly.

(3) Checking for completion of cold standby configuration startup processing

Check the following items:

- Execute HA Monitor's `monshow` command in the active system. Verify that the active system is executing (`ONL`) and the standby system is on standby (`SBY`).
  For details about the `monshow` command, see *Commands* in the manual *HA Monitor for Linux(R) (x86)*.

- Check the message log file of the HADB server in the active system. Confirm that the return code of the `adbstart` command is `0` or `4`.

17.4.2 How to terminate the cold standby configuration

(1) Procedures for terminating the cold standby configuration

(a) Terminating the cold standby configuration normally

This subsection explains how to terminate the cold standby configuration normally.

Procedure:

1. Execute the `adbstop` command in the active system.
2. Execute HA Monitor's `monend` command in the active system.

(b) Terminating the cold standby configuration forcibly

Execute HA Monitor's `monend` command in the active system.

(2) Termination modes for the cold standby configuration

The following table lists and describes the termination modes supported for the cold standby configuration.

Table 17-5: Termination modes for the cold standby configuration

No.	Termination mode		Command and option to be executed	Explanation	Termination mode for cold standby configuration
1	Termination of cold standby configuration	Normal termination of cold standby configuration	`adbstop` and `monend`	This is the normal termination mode. When HA Monitor's `monend` command is executed after the `adbstop` command has terminated normally in the active system, the HADB server terminates normally, and then the cold standby configuration terminates normally.	Normal termination
2		Forced termination of cold standby configuration	`monend`	When only HA Monitor's `monend` command is executed in the active system, the HADB server is terminated forcibly, and then the cold standby configuration is terminated forcibly.	Forced termination
3		Abnormal termination of cold standby configuration	--	When operations cannot be continued due to an error, the HADB server is terminated immediately without waiting for completion of the current transaction. If there is no standby system, the cold standby configuration is terminated abnormally.	Abnormal termination

No.	Termination mode		Command and option to be executed	Explanation	Termination mode for cold standby configuration
4	Occurrence of hot standby	Normal termination of HADB server in the active system	`adbstop` and `monswap`	When there is a standby system and HA Monitor's `monswap` command is executed after the `adbstop` command has terminated normally in the active system, the HADB server terminates normally in the active system, and then the operation is switched over to the standby system.	--
5		Forced termination of HADB server in the active system	`monswap`	When there is a standby system and only HA Monitor's `monswap` command is executed in the active system, the HADB server is terminated forcibly in the active system, and then the operation is switched over to the standby system.	--
6		Abnormal termination of HADB server in the active system	--	When operations cannot be continued due to an error, the HADB server is terminated immediately without waiting for completion of the current transaction. If there is a standby system and the active system is terminated abnormally, operations are switched over to the standby system. Transaction recovery processing is performed during the hot standby processing.	--

Legend:

--: Not applicable.

17.5 Application operations (in the case of the cold standby configuration)

This section provides notes about running application programs. For details about how to create an application and other information, see the *HADB Application Development Guide*.

17.5.1 Connecting to the HADB server

In the cold standby configuration, set your application programs to connect to the HADB server in the active system by using an alias IP address. For details about how to set an alias IP address, see the manual *HA Monitor for Linux(R) (x86)*.

17.5.2 In the event of a failure in the active system

If a failure occurs in the active system, a transaction that was running at the time of the failure rolls back.

In addition, all application programs that were connected to the HADB server are disconnected. Therefore, before you can re-execute the application programs, you must connect to the HADB server again.

17.6 Backing up a database (in the case of the cold standby configuration)

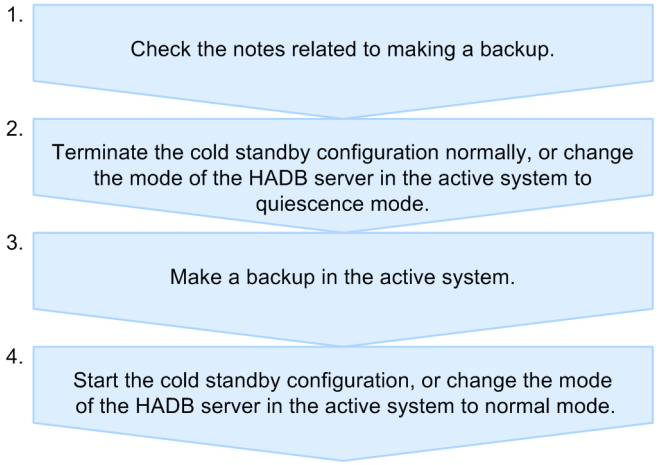
This section explains how to make a backup and recover a database from a backup in the cold standby configuration.

For details about how to make a backup and recover a database from a backup in a system that is not in the cold standby configuration, see 10.3 Backing up a database.

17.6.1 Backup acquisition method

The following figure shows the general procedure for backing up a database.

Figure 17-2: General procedure for backing up a database (in the case of the cold standby configuration)



The following subsections explain these steps in detail.

(1) Check the notes related to making a backup

Check the notes related to making a backup with reference to 10.3.1 Backup acquisition method.

(2) Terminate the cold standby configuration normally, or change the mode of the HADB server in the active system to the quiescence mode

Perform either of the following operations:

- Execute the `adbstop` command and HA Monitor's `monend` command to terminate the cold standby configuration normally.
- Execute the `adbchgsrvmode` command in the active system to change the mode of the HADB server in the active system to quiescence mode.

If you do not perform either of the above operations, database consistency cannot be achieved when the database is restored from a backup.

To check the status of the HADB server, execute the `adbls -d srv` command on the HADB server in the active system.

(3) Make a backup in the active system

Make a backup in the active system. For a list of files to back up, see (2) Backup procedure in 10.3.1 Backup acquisition method.

If a file to be backed up is a symbolic link file, you need a copy of the linked file or block special file. Do not use backed-up files as sparse files.

If the cold standby configuration has terminated normally, ensure that the DB directory has been mounted in `$DBDIR` before you start copying files. If the DB directory is not mounted, execute the `mount` OS command to mount the DB directory in `$DBDIR`.

(4) Start the cold standby configuration, or change the mode of the HADB server in the active system to normal mode

Perform either of the following operations:

- When the `adbstop` command and HA Monitor's `monend` command were used to terminate the cold standby configuration normally before making a backup

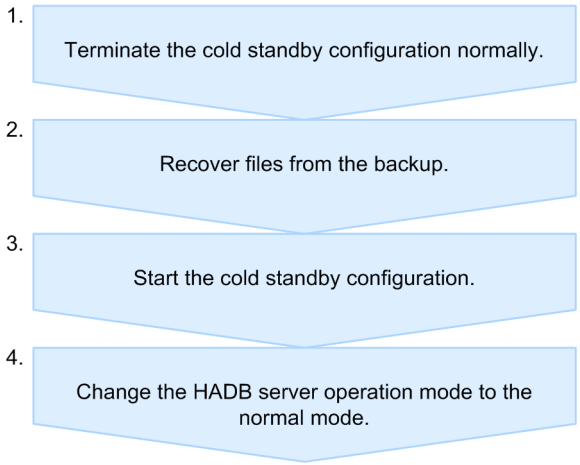
  Execute HA Monitor's `monbegin` command to start the cold standby configuration.
- When the `adbchgsrvmode` command was used to change the mode of the HADB server in the active system to quiescence mode before making a backup

  Execute the `adbchgsrvmode` command in the active system to change the mode of the HADB server in the active system to normal mode.

17.6.2 Recovering a database from a backup

The following figure shows the general procedure for recovering a database from a backup.

Figure 17-3: General procedure for recovering a database from a backup (in the case of the cold standby configuration)



When recovering a database from a backup, also see 10.3.2 Recovering the database from the backup.

The following subsections explain the steps shown in the above figure.

(1) Terminate the cold standby configuration normally

Execute the `adbstop` command and HA Monitor's `monend` command to terminate the cold standby configuration normally.

If you perform database recovery without first terminating the cold standby configuration normally, database compatibility cannot be achieved.

(2) Recover files from the backup

Recover the files from the backup you acquired in (3) Make a backup in the active system under 17.6.1 Backup acquisition method.

Furthermore, before recovering the files from backup, make sure that the DB directory is mounted to `$DBDIR`. If the directory is not mounted, execute the OS's `mount` command to mount the DB directory to `$DBDIR`.

(3) Start the cold standby configuration

Execute HA Monitor's `monbegin` command to start the cold standby configuration.

(4) Change the HADB server operation mode to the normal mode

If an HADB server is started after the database has been recovered from a backup that was acquired in the quiescence mode, the HADB server remains in the quiescence mode. In this case, execute the `adbchgsrvmode` command in the active system to change the HADB server operation mode to normal mode.

17.6.3 Backup operation example (using OS commands)

This subsection provides an example of a backup operation using OS commands (an operation example for acquiring a full backup).

(1) System configuration examples

This example assumes the cold standby configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration with `hadb01` as the active system. The DB directory, the archive directory, the directory for storing synonym dictionary files, and the audit trail directory have the following directory structures:

■ Directory structure of `hadb01`

- DB directory configuration

```
/HADB/db
 ├ADBDIC ... Block special file (10 MB in size)
 ├ADBMST ... Block special file (512 MB in size)
 ├ADBSTBL ... Block special file (512 MB in size)
 ├ADBWORK ... Directory on the file system
 ├ADBWRK ... Block special file (2 GB in size)
 ├ADBUTBL01 ... Block special file (4 GB in size)
 ├ADBUIDX01 ... Block special file (2 GB in size)
 ├SPOOL ... Directory on the file system
 └ADBSYS ... Directory on the file system
```

- Archive directory structure

```
/HADB/archive
```

The archive directory is created when defining an archivable multi-chunk table.

- Synonym dictionary file storage directory structure

```
/mnt/syndict
```

The directory storing synonym dictionary files is created in order to perform synonym searches.

- Audit trail directory structure

```
/mnt/audit
```

The audit trail directory is created in environments where the audit trail facility is used.

■ Directory structure of `hadb02`

- DB directory configuration

```
/HADB/db
```

- Archive directory structure

```
/HADB/archive
```

The archive directory is created when defining an archivable multi-chunk table.

- Mount point for synonym dictionary file storage directory

```
/mnt/syndict
```

The directory storing synonym dictionary files is created in order to perform synonym searches.

- Mount point for audit trail directory

```
/mnt/audit
```

The audit trail directory is created in environments where the audit trail facility is used.

(2) Making a backup

The following shows the procedure for using OS commands to back up a database.

Procedure:

1. Terminate the cold standby configuration normally.[#1]
   For details, see 17.4.2 How to terminate the cold standby configuration.
2. Use OS commands to make a backup.
   Back up the DB directory and archive directory by executing the OS's `cp` and `dd` commands on `hadb01` (active system).
   - Backing up the DB directory

```
cp -r /HADB/db /HADB_bkup/db/dbdir
dd if=/HADB/db/ADBMST of=/HADB_bkup/db/ADBMST bs=524288
dd if=/HADB/db/ADBDIC of=/HADB_bkup/db/ADBDIC bs=524288
dd if=/HADB/db/ADBSTBL of=/HADB_bkup/db/ADBSTBL bs=524288
dd if=/HADB/db/ADBUTBL01 of=/HADB_bkup/db/ADBUTBL01 bs=524288
dd if=/HADB/db/ADBUIDX01 of=/HADB_bkup/db/ADBUIDX01 bs=524288
```

The backup data is stored in the /HADB_bkup/db directory.

Ensure that the paths of the DB area files (symbolic links) in the backup of the DB directory differ from the paths of the backup DB area files that will be acquired later.

If the DB directory is not mounted in $DBDIR, mount it by executing the operating system's mount command.

When data in a block special file is backed up, the size of the backup will be larger than the actual size of the data being used because the entire volume is copied.

- Backing up the archive directory

```
cp -r /HADB/archive /HADB_bkup/archive
```

If an archivable multi-chunk table is defined, you need to back up the archive directory.

If the archive directory was created on an NFS server, you do not need to back up the archive directory on hadb02 (standby system).

- Backing up the directory for storing synonym dictionary files

```
cp -r /mnt/syndict /HADB_bkup/syndict
```

If you are performing synonym searches, you need to back up the directory for storing synonym dictionary files.

> **Note**
>
> Because the audit trail files in the audit trail directory are regularly moved to the audit trail storage directory, you do not need to back up the audit trail directory.

3. Start the cold standby configuration.#2

For details, see 17.4.1 How to start the cold standby configuration.

#1

Instead of step 1, you can perform the following process:

On hadb01 (active system), execute the adbchgsrvmode command to change the HADB server operation mode in the active system to quiescence mode.

```
adbchgsrvmode --quiescence
```

#2

If you have used the method described in footnote 1 of step 1, execute the adbchgsrvmode command on hadb01 (active system) to change the HADB server operation mode in the active system to normal mode.

```
adbchgsrvmode --normal
```

(3) Recovering the database from a backup

The following shows the procedure for recovering the database from a backup.

Procedure:

1. Terminate the cold standby configuration normally.

For details, see 17.4.2 How to terminate the cold standby configuration.

2. Use OS commands to recover the database

On hadb01 (active system), execute the OS's cp and dd commands to recover the database from its backup.

- **Recovering the DB directory**

```
cp -r /HADB_bkup/db/dbdir/* /HADB/db
dd if=/HADB_bkup/db/ADBMST of=/HADB/db/ADBMST bs=524288
dd if=/HADB_bkup/db/ADBDIC of=/HADB/db/ADBDIC bs=524288
dd if=/HADB_bkup/db/ADBSTBL of=/HADB/db/ADBSTBL bs=524288
dd if=/HADB_bkup/db/ADBUTBL01 of=/HADB/db/ADBUTBL01 bs=524288
dd if=/HADB_bkup/db/ADBUIDX01 of=/HADB/db/ADBUIDX01 bs=524288
```

The backup file is stored in the directory `/HADB_bkup/db`.

If the DB directory is not mounted in `$DBDIR`, mount it by executing the operating system's `mount` command.

- **Recovering the archive directory**

```
rm -r /HADB/archive/*
cp -r /HADB_bkup/archive/* /HADB/archive
```

If an archivable multi-chunk table is defined, you need to recover the archive directory.

If the archive directory was created on an NFS server, you do not need to recover the archive directory on `hadb02` (standby system).

- **Recovering the directory for storing synonym dictionary files**

```
rm -r /mnt/syndict/*
cp -r /HADB_bkup/syndict/* /mnt/syndict
```

If you are performing synonym searches, you need to recover the directory for storing synonym dictionary files.

3. Start the cold standby configuration.

   For details, see 17.4.1 How to start the cold standby configuration.

   If the HADB server operation mode in the active system is in quiescence mode after the cold standby configuration has started, execute the `adbchgsrvmode` command on `hadb01` (active system) to change the HADB server operation mode to normal mode.

```
adbchgsrvmode --normal
```

17.6.4 Backup operation example (using ShadowImage)

This subsection provides an example of a backup operation using ShadowImage.

(1) System configuration examples

This example assumes the cold standby configuration shown in Figure 17-1: Example of a system configuration using the cold standby configuration with `hadb01` as the active system. The DB directory, the archive directory, and the directory storing the synonym dictionary files have the following directory structures:

■ Directory structure of `hadb01`

- DB directory configuration

```
/HADB/db (Built on FS001)
 ├ADBDIC ... Block special file (using LU002)
 ├ADBMST ... Block special file (using LU001)
 ├ADBSTBL ... Block special file (using LU003)
 ├ADBWORK ... Directory on the file system
 ├ADBWRK ... Block special file (using WRK003)
 ├ADBUTBL01 ... Block special file (using LU004)
 ├ADBUIDX01 ... Block special file (using LU005)
```

```
 ├SPOOL ... Directory on the file system
 └ADBSYS ... Directory on the file system
```

- Archive directory structure

```
/HADB/archive
```

  The archive directory is created when defining an archivable multi-chunk table.

- Synonym dictionary file storage directory structure

```
/mnt/syndict
```

  The directory storing synonym dictionary files is created in order to perform synonym searches.

- Audit trail directory structure

```
/mnt/audit
```

  The audit trail directory is created in environments where the audit trail facility is used.

■ Directory structure of `hadb02`

- DB directory configuration

```
/HADB/db
```

- Archive directory structure

```
/HADB/archive
```

  The archive directory is created when defining an archivable multi-chunk table.

- Mount point for synonym dictionary file storage directory

```
/mnt/syndict
```

  The directory storing synonym dictionary files is created in order to perform synonym searches.

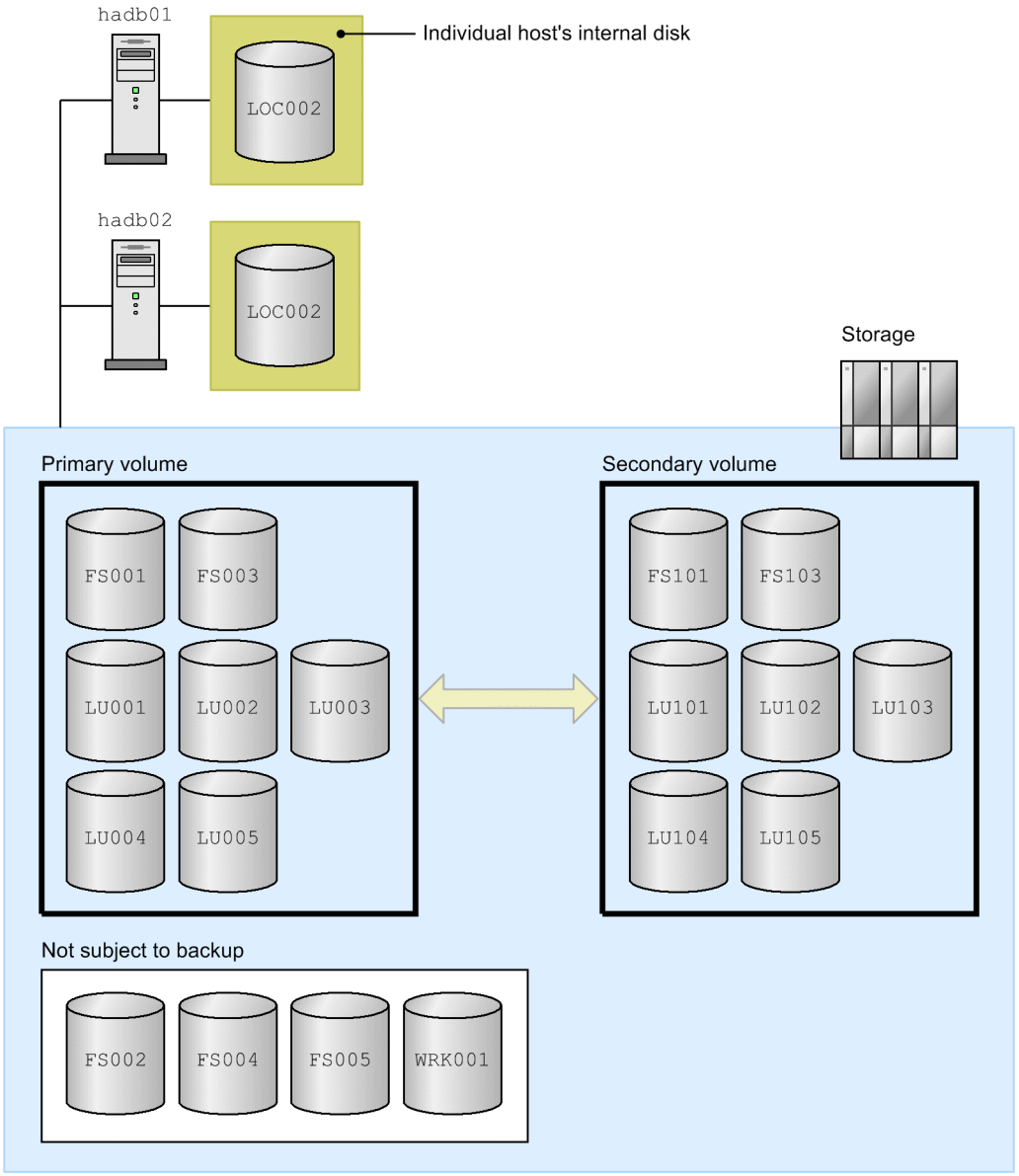
- Mount point for audit trail directory

```
/mnt/audit
```

  The audit trail directory is created in environments where the audit trail facility is used.

■ Logical unit configuration

The following figure shows an example of storage configuration in the cold standby configuration:



Explanation

- FS001 is a file system for the DB directory.
- FS002 is a file system storing temporary work files.
- FS003 is a file system storing synonym dictionary files.
- FS004 is a file system to which audit trail files are output.
- FS005 is the following file systems:
  - File system for storing input data files that are used in data import processing
  - File system for storing CSV files used by the ADB_CSVREAD function
  - File system where audit trail storage directory will be created
- LU001 to LU005 are used for DB area files.
- WRK001 is used for work table DB area files.
- The sub-volumes that correspond to the primary volumes FS00X and LU00X are FS10X and LU10X.

(2) Making a backup

The following procedure explains the procedure for using ShadowImage to make a backup.

Procedure:

1. Terminate the cold standby configuration normally
   For details, see 17.4.2 How to terminate the cold standby configuration.
2. Unmount the file system subject to backup processing.
3. Make the backup.
   Back up `FS001`, `FS003`, and all logical units `LU001` to `LU005` by following the procedure described in (3) Making a backup under 10.3.4 Backup operation example (using ShadowImage).
4. Start the cold standby configuration.
   For details, see 17.4.1 How to start the cold standby configuration.

(3) Recovering the database from a backup

The following procedure explains the procedure for recovering the database from a backup.

Procedure:

1. Terminate the cold standby configuration.

   ```
   adbstop
   monend
   ```

   On `hadb01` (active system), execute the `adbstop` command and HA Monitor's `monend` command to terminate the cold standby configuration.
   A backup cannot be acquired in quiescence mode because the file system will be unmounted in the subsequent step.
2. Unmount the file system subject to backup processing.
3. Recover the database.
   Recover `FS001`, `FS003`, and all logical units `LU001` to `LU005` by following the procedure described in (4) Recovering a database from a backup under 10.3.4 Backup operation example (using ShadowImage).
4. Start the cold standby configuration.
   For details, see 17.4.1 How to start the cold standby configuration.

17.7 Status monitoring (in the case of the cold standby configuration)

This section explains the system monitoring method that is unique to the cold standby configuration.

17.7.1 Checking the memory usage

To check the usage status of the memory used by HADB servers, execute the `adbstat` command. For details about how to check memory usage, see 10.6.1 Checking the usage status of all memory.

By executing the `adbstat` command, you can check the maximum usage of all memory (`Total_memory_max_size`) for the HADB server in the active system.

17.8 DB area operations (in the case of the cold standby configuration)

This section explains how to use DB areas in the cold standby configuration.

17.8.1 Adding, deleting, or expanding data DB areas (in the case of the cold standby configuration)

Use WWNs because the same logical units must be specified in the active system and the standby system.

■ **When using SCSI reservation for shared disk**

When using SCSI reservation for shared disk, you need to change the operand specification in the `servers` file of HA Monitor. The following shows the procedure.

- **Procedure (when adding or expanding a data DB area):**

  1. Terminate the cold standby configuration normally.

     For details about how to terminate the cold standby configuration, see 17.4.2 How to terminate the cold standby configuration.

  2. Change the specification of the `servers` file of HA Monitor.

     Add the device name of the disk to be added in the `scsi_device` or `dmmp_device` operand. Change the specification of the `servers` file in both the active system and the standby system.

  3. Start the cold standby configuration normally.

     For details about how to start the cold standby configuration, see 17.4.1 How to start the cold standby configuration.

  4. Add or expand a data DB area by executing the `adbmodarea` command.

     For details about how to add data DB areas, see 11.10.1 Adding data DB areas.

     For details about how to expand data DB areas, see 11.10.3 Expanding a data DB area (adding a data DB area file).

- **Procedure (when deleting a data DB area):**

  1. Delete a data DB area by executing the `adbmodarea` command.

     For details about how to delete data DB areas, see 11.10.2 Deleting a data DB area.

  2. Terminate the cold standby configuration normally.

     For details about how to terminate the cold standby configuration, see 17.4.2 How to terminate the cold standby configuration.

  3. Change the specification of the `servers` file of HA Monitor.

     Delete the device name of the deleted disk from the `scsi_device` or `dmmp_device` operand. Change the specification of the `servers` file in both the active system and the standby system.

  4. Start the cold standby configuration normally.

     For details about how to start the cold standby configuration, see 17.4.1 How to start the cold standby configuration.

17.8.2 Work table DB area operations (in the case of the cold standby configuration)

Specifying the same logical units in the active system and the standby system will not cause any problems.

17.9 Table and index operations (in the case of the cold standby configuration)

This section explains how to use tables and indexes in the cold standby configuration.

17.9.1 Base table operations (in the case of the cold standby configuration)

This subsection explains how to use base tables in the cold standby configuration.

The following base table operation is unique to the cold standby configuration:

- Method of releasing a base table from non-updatable status
- Operations when using archivable multi-chunk tables

All other operations are the same as when the cold standby configuration is not used. See 11.1 Base table operations.

(1) Releasing a base table from non-updatable status (in the case of the cold standby configuration)

If hot standby processing occurs while the `adbimport` command, the `adbidxrebuild` command, or the `adbunarchivechunk` command is executing, the target base table becomes non-updatable. If this happens, re-execute the command in the new active system.

If the following file systems can be accessed from both hosts using the same paths, you can resume the processing from the point where it was interrupted when the command is re-executed:

- File system storing the DB directory
- File system storing the input data files that are used in data import processing (applicable to the `adbimport` command)
- File system storing temporary work files

For details about the non-updatable status of base tables, see 15.8.1 Steps to take when a base table becomes non-updatable.

(2) Operation when using archivable multi-chunk tables (in the case of the cold standby configuration)

Make sure that the archive directory can be referenced using the same absolute path from both the active and standby systems.

For example, when using the distributed file system NFS, use the following procedure to prepare the archive directory:

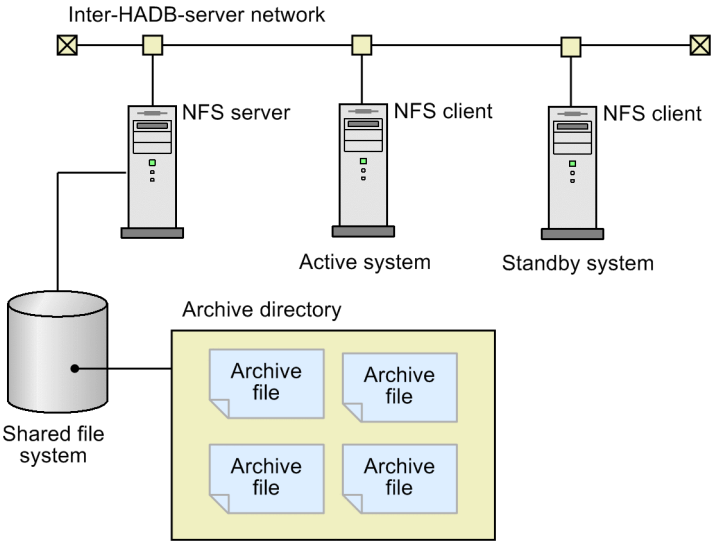
Procedure:

1. Prepare the NFS server, and then create an archive directory on the NFS server.
2. Export the archive directory you created in step 1.
   Perform this operation on the NFS server.
3. Mount the directory exported in step 2 so that the paths are the same on the active and standby systems.

Perform this operation on the active system and standby system.

The following figure shows an example system configuration using NFS:

Figure 17-4: Example of a system configuration that uses NFS



17.9.2 Index operations (in the case of the cold standby configuration)

This subsection explains how to use B-tree indexes, text indexes, and range indexes in the cold standby configuration.

The following operations related to B-tree indexes, text indexes, and range indexes are unique to the cold standby configuration:

- Releasing a B-tree index from unfinished status
- Releasing a text index from unfinished status
- Releasing a range index from unfinished status

All other operations are the same as when the cold standby configuration is not used. See 11.3 Index operations.

(1) Releasing a B-tree index from unfinished status (in the case of the cold standby configuration)

If hot standby processing occurs while the `adbimport` command or the `adbidxrebuild` command is executing, B-tree indexes are placed in unfinished status. If this happens, re-execute the command in the new active system.

If the following file systems can be accessed from both hosts using the same paths, you can resume the processing from the point where it was interrupted when the command is re-executed:

- File system storing the DB directory
- File system storing the input data files that are used in data import processing (applicable to the `adbimport` command)
- File system storing temporary work files

For details about the unfinished status of B-tree indexes, see 15.9.1 Steps to take when unfinished status is applied to a B-tree index.

(2) Releasing a text index from unfinished status (in the case of the cold standby configuration)

If hot standby processing occurs while the `adbimport` command or the `adbidxrebuild` command is executing, text indexes are placed in unfinished status. If this happens, re-execute the command in the new active system.

If the following file systems can be accessed from both hosts using the same paths, you can resume the processing from the point where it was interrupted when the command is re-executed:

- File system storing the DB directory
- File system storing the input data files that are used in data import processing (applicable to the `adbimport` command)
- File system storing temporary work files

For details about the unfinished status of text indexes, see 15.10.1 Steps to take when unfinished status is applied to a text index.

(3) Releasing a range index from unfinished status (in the case of the cold standby configuration)

If hot standby processing occurs while the `adbidxrebuild` command is executing, range indexes are placed in unfinished status. If this happens, re-execute the command in the new active system.

If the following file systems can be accessed from both hosts using the same paths, you can resume the processing from the point where it was interrupted when the command is re-executed:

- File system storing the DB directory
- File system storing temporary work files

For details about the unfinished status of range indexes, see 15.11.1 Steps to take when unfinished status is applied to a range index.

17.10 DB directory operations (in the case of the cold standby configuration)

When the cold standby configuration is started, HA Monitor mounts the DB directory in the mount point.

When you execute the `adbinit` command, mount the DB directory manually.

17.11 Statistical information operations (in the case of the cold standby configuration)

In the cold standby configuration, the HADB server's statistical information in the standby system cannot be referenced.

For details about how to check statistical information, see 10.10 Performing statistical analysis (checking HADB server operation information).

17.12 Operations when a failure occurs

This section explains the operation in the event of a failure.

17.12.1 In the event of a failure in the active system

If a failure occurs in the active system and the HADB server terminates abnormally, as many attempts are made to restart the HADB server as the value specified in the `servexec_retry` operand in HA Monitor's `servers` file. If restart fails, hot standby processing is performed.

To identify the cause of the failure, take the action as described in 14.1 Error-handling flow. Perform the procedure provided in that section in the host resulting in the failure.

Also collect the following troubleshooting information in the host resulting in the failure:

- Use the `adbinfoget` command to acquire the HADB server's troubleshooting information.
- Use HA Monitor's `monts` command to acquire HA Monitor's troubleshooting information.

When the HADB server is restarted, the troubleshooting-information file is stored in the directory for storing troubleshooting-information files that was created in 17.3.2 Installing HADB server and setting up an environment. The troubleshooting-information files stored in that directory are not deleted automatically, so you must periodically delete any unneeded troubleshooting-information files.

17.12.2 In the event of a failure in the standby system

If a failure occurs in the standby system, operation continues in the active system.

To identify the cause of the failure, take the action as described in 14.1 Error-handling flow. Perform the procedure provided in that section in the host resulting in the failure.

Also collect the following troubleshooting information in the host resulting in the failure:

- Use the `adbinfoget` command to acquire the HADB server's troubleshooting information.
- Use HA Monitor's `monts` command to acquire HA Monitor's troubleshooting information.

17.12.3 In the event of a communication failure between the active system and the standby system

If a communication failure occurs between the active system and the standby system, HA Monitor stops the server machine that caused the communication failure, and then shuts its power supply off.

First, re-start the server machine that stopped, and then eliminate the cause of the communication failure. Then, restore the corresponding host in the cold standby configuration. For details about how to restore a host to the cold standby configuration, see 17.12.4 Returning a system to the cold standby configuration.

17.12.4 Returning a system to the cold standby configuration

If hot standby processing occurs as a result of a failure, the host resulting in the failure is stopped. To return this system to the cold standby configuration as the standby system, eliminate the cause of the failure, and then execute HA Monitor's `monbegin` command.

17.13 Planned hot standby in the cold standby configuration

To perform planned hot standby processing, execute the `adbstop` command in the active system, and then execute HA Monitor's `monswap` command. Hot standby processing occurs after the HADB server in the active system terminates normally.

If you only execute the HA Monitor's `monswap` command in the active system, the HADB server is terminated forcibly in the active system, and then hot standby processing is performed.

17.14 Troubleshooting (in the case of the cold standby configuration)

This section explains how to troubleshoot when problems occur in the cold standby configuration.

17.14.1 Problems related to startup or termination of the cold standby configuration

This subsection explains the steps to take when a problem related to startup or termination of the cold standby configuration occurs.

(1) When the cold standby configuration cannot be started normally

The following table lists and describes possible reasons why the cold standby configuration cannot be started normally and the corrective actions to take.

Table 17-6: Possible reasons why the cold standby configuration cannot be started normally and the corrective actions to take

No.	Possible cause	Action to take
1	A failure has occurred in the active system or the standby system.	Start the HADB server with the `adbstart` command, not HA Monitor's `monbegin` command. Note that in this case, the cold standby configuration is not used. Alternatively, if there is a substitute machine, set up the substitute machine's environment so that it is the same as the server machine on which the failure occurred. Then, start the HADB servers in the cold standby configuration by following the procedure in 17.4.1 How to start the cold standby configuration.
2	The DB directory has not been mounted.	Re-evaluate the values of the following operands specified in HA Monitor's `servers` file: • `disk` operand • `fs_name` operand • `fs_mount_dir` operand • `fs_mount_opt` operand
3	The DB directory has not been initialized correctly.	Check if the DB directory had been mounted when the `adbinit` command was executed.

If the cause is other than the above, eliminate the cause of the error according to the corrective action provided for the message that was output. Then, start the cold standby configuration according to the procedure described in17.4.1 How to start the cold standby configuration.

(2) When the cold standby configuration cannot be restarted

If a failure has occurred in the active system or the standby system, remove the system resulting in the failure from the cold standby configuration. Then, restart the cold standby configuration according to the procedure described in17.4.1 How to start the cold standby configuration.

For any other cause, eliminate the cause of the error according to the corrective action given for the message that was output. Then, restart the cold standby configuration according to the procedure described in17.4.1 How to start the cold standby configuration.

(3) When the cold standby configuration cannot be terminated

Eliminate the cause of the error according to the corrective action provided for the message that was output. Then, terminate the cold standby configuration according to the procedure described in17.4.2 How to terminate the cold standby configuration.

(4) When the HADB server cannot be started after hot standby processing

The following table lists and describes possible reasons why the HADB server cannot be started after hot standby processing and the corrective actions to take.

Table 17-7: Possible reasons why the HADB server cannot be started after hot standby processing and the corrective actions to take

No.	Possible cause	Action to take
1	The DB directory has not been mounted.	Re-evaluate the values of the following operands specified in HA Monitor's `servers` file: • `disk` operand • `fs_name` operand • `fs_mount_dir` operand • `fs_mount_opt` operand
2	The HADB server definitions do not match between the active system and the standby system.	Ensure that the HADB server definitions match between the active system and the standby system. If the HADB server in the active system is terminated forcibly or abnormally, the HADB server's startup mode is restart. In this case, the restart processing fails if the number of threads and the size of the memory for the HADB server to be started are smaller than those of the HADB server in the active system before hot standby processing.
3	When a DB area is expanded, a block special file that cannot be referenced by the standby system was specified.	Take the following corrective action: 1. Use the OS's `dd` command, for example, to copy the contents of the DB area files to block special files that can be referenced from both the active system and the standby system. 2. Change the symbolic links to the DB area files (block special files) in the DB directory to the link to the block special files used in step 1.
4	When a DB area is expanded, the path of the specified block special files differs from that specified in the standby system.	We recommend that you use WWNs in such a manner that the paths are the same in both the active system and the standby system. DB area files in the DB directory are symbolic links to block special files. Manually change the links so that WWNs are used in paths.

If the cause is other than the above, eliminate the cause of the error according to the corrective action provided for the message that was output. Then, restart the cold standby configuration according to the procedure described in17.4.1 How to start the cold standby configuration.

17.14.2 Application-related problems (in the case of the cold standby configuration)

If an application-related problem occurs, first take the appropriate corrective action as explained in 15.1 Application-related problems. If this does not resolve the problem, read the information provided in this subsection.

(1) Steps to take when an application cannot be executed

The following table describes the possible reasons why an application cannot be executed and the corrective actions to take.

Table 17-8: Possible reasons why an application cannot be executed and the corrective actions to take

No.	Possible cause	Action to take
1	The host name of the connection destination might be incorrect.	Check whether a host name is specified that corresponds to the alias IP address used for communication between HADB clients and HADB servers. Check the locations where the host name of the connection destination for the application is specified, such as the client definition, and correct the host name if necessary.

17.14.3 Command related problems (in the case of the cold standby configuration)

If an command-related problem occurs, first take the appropriate corrective action as explained in 15.2 Command-related problems. If this does not resolve the problem, read the information provided in this subsection.

(1) Steps to take when a command cannot be executed

The following table describes the possible reasons why the following commands cannot be executed, and the steps that can be taken:

- `adbimport`
- `adbidxrebuild`
- `adbmergechunk`
- `adbunarchivechunk`
- `adbreorgsystemdata`
- `adbsyndict`

Table 17-9: Possible reasons why commands cannot be executed and the corrective actions to take

No.	Possible cause	Action to take
1	The file system storing the input data files that are used in data import processing has not been mounted.	Re-examine the settings for the file systems that are shared. Share file systems in such a manner that their paths are the same between the active system and the standby system.
2	The following file systems have not been mounted: • File system that stores temporary work files • File system that stores synonym dictionary files	Re-evaluate the values of the following operands specified in HA Monitor's `servers` file: • `disk` operand • `fs_name` operand • `fs_mount_dir` operand • `fs_mount_opt` operand If specified values are invalid, mount the file system manually and then re-execute the command.

No.	Possible cause	Action to take
3	The path of the file system mount point differs between the active system and the standby system.	Re-evaluate the values of the following operands specified in HA Monitor's `servers` file: • `disk` operand • `fs_name` operand • `fs_mount_dir` operand • `fs_mount_opt` operand If specified values are invalid, mount the file system manually and then re-execute the command. If hot standby processing has occurred during command execution, but the following file systems can be referenced using the same paths in both the active system and the standby system, the command can be re-executed from the point where the processing was interrupted: • File system that stores the DB directory • File system that stores the input data files that are used in the data import processing • File system that stores temporary work files • File system that stores synonym dictionary files

17.14.4 Problems related to files in the DB directory (in the case of the cold standby configuration)

(1) Steps to take when a failure occurs in a system log file

Take the steps indicated in the message that is output.

Also, check whether the file system for the DB directory is mounted in `$DBDIR`.

In addition, see 15.5.2 Steps to take when a problem occurs in a system log file.

17.15 Changing the host name or IP address of the server machine's OS (in the case of the cold standby configuration)

This section explains how to change the host name or IP address of the OS of a server machine on which the HADB server is installed in a cold standby configuration.

If you change only the IP addresses of the active and standby systems, there is no task to be performed for an HADB server and HA Monitor.

■ **Tasks to be performed on an HADB server in a cold standby configuration**

The following explains the tasks to be performed on an HADB server in a cold standby configuration to change the host name or IP address of the server machine's OS.

1. **Normally terminate the cold standby configuration.**

   Normally terminate the cold standby configuration. For details about the termination procedure, see (a) Terminating the cold standby configuration normally in (1) Procedures for terminating the cold standby configuration under 17.4.2 How to terminate the cold standby configuration.

   Also, normally stop HA Monitor on all hosts by using the `monstop` command.

2. **Change the HA Monitor settings on all hosts.**

   Change the HA Monitor settings on all hosts. For details, see 17.3.4 Setting up an HA Monitor environment.

   - To change the host name of the LAN to be used as the monitoring path of HA Monitor

     Change the settings in the HA Monitor environment settings file (`sysdef` file).

   - To change the alias IP address

     Change the settings in the LAN status settings file for connection (*`server-identification-name`*`.up` file) and the LAN status settings file for disconnection (*`server-identification-name`*`.down` file).

3. **Start the cold standby configuration.**

   Use the `monstart` command to start HA Monitor on all hosts. Then, start the cold standby configuration. For details about the start procedure, see (1) Procedure for starting the cold standby configuration in 17.4.1 How to start the cold standby configuration.

■ **Tasks to be performed on the HADB client**

The following explains the tasks to be performed on the HADB client to change the host name or IP address of the server machine's OS. Perform the following tasks when changing the alias IP address used for communication between the HADB client and the HADB server.

1. **Terminate the application programs.**

   Terminate all application programs that connect to the HADB server in a cold standby configuration.

2. **Change the alias IP address specified in the client definition.**

   To change the alias IP address that is used for communication between the HADB client and HADB server, change the value of the `adb_clt_rpc_srv_host` operand in the client definition. For details, see 17.3.6 Creating client definitions.

   For details about the `adb_clt_rpc_srv_host` operand in the client definition, see *Contents of operands in the client definition* under *Designing Client Definitions* in the *HADB Application Development Guide*.

   - If the JDBC driver is used

     Change the host name or IP address of the HADB server specified with the property that corresponds to the `adb_clt_rpc_srv_host` operand in the client definition. The JDBC driver can use a system property,

user property, or connection URL property. For details, see *Setting Up an Environment for the JDBC Driver* in the *HADB Application Development Guide*.

- If the ODBC driver or a CLI function is used

  Change the host name or IP address of the HADB server specified for the `adb_clt_rpc_srv_host` operand in the client definition. For details, see *Setting Up an Environment for an HADB Client (If the ODBC Driver and CLI Functions Are Used)* in the *HADB Application Development Guide*.

3. **Start the application programs.**

   Start all application programs that connect to the HADB server in a cold standby configuration.

Important

If the system is not in a cold standby configuration, see 8.11 Changing the host name or IP address of the server machine's OS.

17.16 Upgrading the HADB server version (in the case of the cold standby configuration)

This section explains how to upgrade the HADB server version in the cold standby configuration.

Procedure:

1. Terminate the cold standby configuration normally.

   For details about the procedure for terminating a cold standby configuration normally, see 17.4.2 How to terminate the cold standby configuration.

2. Upgrade the HADB server version in the active system and the standby system.

   For details about the version upgrade procedure, see 8.6 Upgrading the HADB server version.

3. Start the cold standby configuration.

   For details about the procedure for starting a cold standby configuration, see 17.4.1 How to start the cold standby configuration.

First, terminate the cold standby configuration normally, and then upgrade the HADB server version. If you have terminated the cold standby configuration normally after hot standby processing had occurred, start the cold standby configuration normally, and then terminate the cold standby configuration normally. If this step is skipped, startup of the upgraded HADB server might fail (the `KFAA40002-E` message will be output).

When you upgrade the HADB server version in the cold standby configuration, ensure that the HADB servers in both the active system and the standby system have terminated normally.

17.17 Swapping the HADB server with its revised version (in the case of the cold standby configuration)

You can swap the HADB server with its revised version by using either of the following methods:

- Normally terminating and then swapping the HADB server in a cold standby configuration
- Performing swapping on a per-host basis

17.17.1 Performing a normal termination of a cold standby configuration HADB server and then swapping

The following procedure shows how to perform swapping of the HADB server with its revised version after performing a normal termination of the cold standby configuration:

Procedure:

1. Perform a normal termination of the cold standby configuration

   For details about the procedure for terminating a cold standby configuration normally, see 17.4.2 How to terminate the cold standby configuration.
2. Swap the HADB server with its revised version on the active and standby systems

   For details about how to swap the HADB server with a revised version, see 8.8 Swapping the HADB server with its revised version.
3. Perform a normal start of the cold standby configuration

   For details about the procedure for starting a cold standby configuration normally, see 17.4.1 How to start the cold standby configuration.

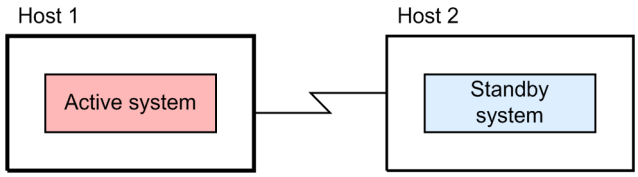
Important

- After performing a normal termination of the cold standby configuration, swap the HADB server with its revised version. To swap the HADB server with its revised version in a cold standby configuration, the HADB servers of both the active and standby systems need to have been terminated normally.
- If the cold standby configuration undergoes a normal termination after a host swapping has occurred, first perform a normal start of the cold standby configuration, and then perform a normal termination of the same configuration. If you do not perform these operations, starting the HADB server might fail after swapping of the HADB server with its revised version (the `KFAA40002-E` message is output).

17.17.2 Performing swapping on a per-host basis

The following procedure shows how to swap the HADB server with its revised version on a per-host basis after stopping one of the hosts: If you use this method, you can swap the HADB server with its revised version while the HADB server on the active system is running, so there is no need to stop work operations.

The cold standby configuration on which swapping is performed is as follows:



Procedure:

1. Stop host 2 (standby system)

   On host 2, execute the HA Monitor `monsbystp` command, and then stop host 2.

2. Swap the HADB server with its revised version on host 2

   For details about how to swap the HADB server with a revised version, see 8.8 Swapping the HADB server with its revised version.

3. Return host 2 to the cold standby configuration.

   For details about the procedure for returning a host to a cold standby configuration, see 17.12.4 Returning a system to the cold standby configuration.

   Host 2 is returned as the standby system to the cold standby configuration.

4. Perform a planned hot standby

   On host 1, after executing the `adbstop` command, execute the HA Monitor `monswap` command, and then perform a planned hot standby. When the planned hot standby is complete, host 2 becomes the active system, and host 1's HADB server terminates normally.

   For details about planned hot standby, see 17.13 Planned hot standby in the cold standby configuration.

5. Swap the HADB server with its revised version on host 1

   For details about how to swap the HADB server with a revised version, see 8.8 Swapping the HADB server with its revised version.

6. Return host 1 to the cold standby configuration

   For details about the procedure for returning a host to a cold standby configuration, see 17.12.4 Returning a system to the cold standby configuration.

   Host 1 is returned as the standby system to the cold standby configuration.

7. Perform a planned hot standby

   On host 2, after executing the `adbstop` command, execute the HA Monitor `monswap` command, and then perform a planned hot standby. When the planned hot standby is complete, host 1 becomes the active system, and host 2's HADB server terminates normally.

   For details about planned hot standby, see 17.13 Planned hot standby in the cold standby configuration.

8. Return host 2 to the cold standby configuration.

   For details about the procedure for returning a host to a cold standby configuration, see 17.12.4 Returning a system to the cold standby configuration.

   Host 2 is returned as the standby system to the cold standby configuration.

With the preceding step, swapping of the HADB server with its revised version is complete. Note that the relationship between the active and standby systems is the same as before the swapping.

17.18 Operation when performing synonym searches in a cold standby configuration

If you have to perform a synonym search after building a cold standby configuration, you must first prepare the file system for storing synonym dictionary files, and then change the specification of the `servers` file. This section explains each step.

17.18.1 Preparing the file system for storing synonym dictionary files

The file system that stores synonym dictionary files must be inherited when host switchover occurs. The file system to be inherited during host switchover must be created on an LV in a VG made up of disks that can be referenced by both the active and standby systems (shared disks). For information about shared disks, see *Shared disk configuration* in the manual *HA Monitor for Linux(R) (x86)*.

17.18.2 Changing the specification of the servers file

Here, the specified values of operands specified in the HA Monitor `servers` file will be changed.

`FS003` in Figure 17-1: Example of a system configuration using the cold standby configuration is to be the file system for storing synonym dictionary files. The VG name is to be `vg_hadb03`. The LV name is to be `hadb_syndict`.

- `disk` operand

  Add a specification of the absolute path of the VG containing the file system for storing synonym dictionary files. Add `/dev/vg_hadb03` to the `disk` operand.

- `fs_name` operand

  Add a specification of the absolute path of the logical volume corresponding to the file system for storing synonym dictionary files. Add `/dev/vg_hadb03/hadb_syndict` to the `fs_name` operand.

- `fs_mount_dir` operand

  Add a specification of the directory path to which to mount the file system for storing synonym dictionary files. Make sure that you configure the settings so that the active and standby systems both use the same path.

- `fs_mount_opt` operand

  Add mount options for mounting the file system for storing synonym dictionary files.

- `vg_neck`

  Specify `nouse`.

- `fs_neck`

  Specify `nouse`.

For an example of specifying the `servers` file, see the following.

- When using host reset:

  See (b) servers file specification examples in (5) File specification examples (when using host reset) under 17.3.4 Setting up an HA Monitor environment.

- When using SCSI reservation for shared disk:

See (b) servers file specification examples in (6) File specification examples (when using SCSI reservation for shared disk) under 17.3.4 Setting up an HA Monitor environment.

17.18.3 Building the file system for storing synonym dictionary files

The file system for storing synonym dictionary files is built in the active system.

The following execution example initializes `/dev/vg_hadb03/hadb_syndict`, which was created as the LV for storing synonym dictionary files, as an ext4 file system.

```
mkfs -t ext4 /dev/vg_hadb03/hadb_syndict
```

The superuser must execute the preceding command.

17.18.4 Mounting the file system for storing synonym dictionary files

Mount the file system for storing synonym dictionary files, in the active system. Then, unmount the file system for storing synonym dictionary files.

Similarly, mount the file system for storing synonym dictionary files in the standby system, and then unmount it.

For details about how to mount a file system, see (3) Mounting the file system for the DB directory (performed in the active system) in 17.3.7 Creating a database. For details about how to unmount a file system, see (5) Unmounting the file system for the DB directory (performed in the active system) in 17.3.7 Creating a database.

Important

Create mount points so that the active and standby systems both use the same path.

Perform the subsequent tasks by following the procedure explained in 11.16.1 Preparing for synonym search operations.

Important

- Specify the directory of the mounted file system for storing synonym dictionary files, in the server definition `adb_syndict_storage_path` operand.
- Store the synonym list definition file, dictionary creation file, and dictionary deletion file in a shared file system. If you do not create these items in a shared file system, the `adbsyndict` command cannot be executed from both hosts.

17.19 Operation when using the audit trail facility in a cold standby configuration

You must perform the following tasks if you decide to use the audit trail facility after having built a cold standby configuration:

- Prepare the file system where the audit trail directory will be created
- Prepare the audit trail storage directory
- Change the specification of the `servers` file

The following explains each of these tasks in detail.

To collect and centrally manage audit trails by linking with JP1/Audit, see 17.19.4 Linkage with JP1/Audit Management - Manager (in the case of the cold standby configuration).

17.19.1 Preparing the file system where the audit trail directory will be created

The file system where the audit trail directory will be created must be inherited when host switchover occurs. The file system to be inherited during host switchover must be created on an LV in a VG made up of disks that can be referenced by both the active and standby systems (shared disks). For information about shared disks, see *Shared disk configuration* in the manual *HA Monitor for Linux(R) (x86)*.

■ Building the file system where the audit trail directory will be created

Build the file system where the audit trail directory will be created on the active system. The following execution example initializes `/dev/vg_hadb04/hadb_audit`, which was created as the LV for storing audit trail files, as an ext4 file system.

```
mkfs -t ext4 /dev/vg_hadb04/hadb_audit
```

The superuser must execute the preceding command.

■ Mounting the file system where the audit trail directory will be created

Mount the file system where the audit trail directory will be created on the active system. Then, unmount the file system. Similarly, in the standby system, mount and then unmount the file system where the audit trail directory will be created.

For details about how to mount a file system, see (3) Mounting the file system for the DB directory (performed in the active system) in 17.3.7 Creating a database. For details about how to unmount a file system, see (5) Unmounting the file system for the DB directory (performed in the active system) in 17.3.7 Creating a database.

Important

- Create mount points so that the active and standby systems both use the same path.
- In the `adb_audit_log_path` server definition operand, specify the directory created in the mounted file system.

17.19.2 Preparing the file system where the audit trail storage directory will be created

The audit trail files specified in an `ADB_AUDITREAD` function that retrieves audit trail data must be accessible from both the active and standby system using the same absolute paths. For this reason, the audit trail storage directory must be created on a file system that is shared among all nodes.

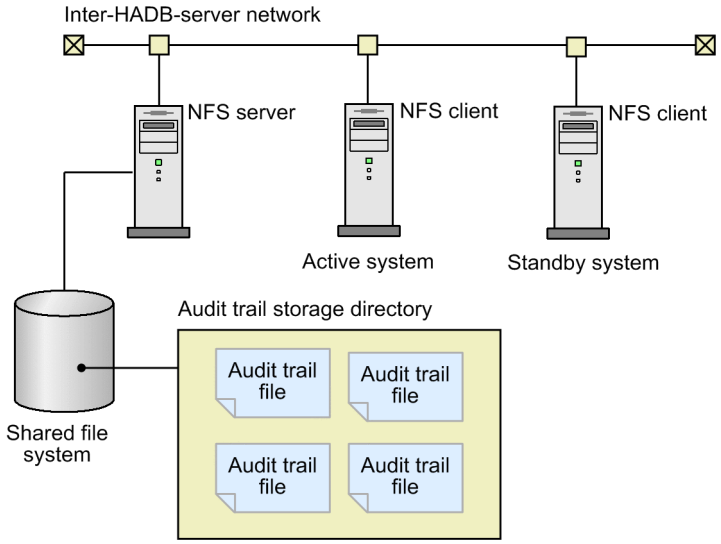
The following is an example of environment setup when using an NFS (a type of distributed file system):

Procedure:

1. After preparing the NFS server, create the audit trail storage directory on the NFS server.
2. Export the audit trail storage directory you created in step 1.
   Perform this operation on the NFS server.
3. Mount the directory exported in step 2 so that the paths are the same on the active and standby systems.
   Perform this operation on the active system and standby system.

The following figure provides an example of a system configuration that uses NFS.

Figure 17-5: Example of a system configuration that uses NFS



17.19.3 Changing the specification of the servers file

Here, the specified values of operands specified in the HA Monitor `servers` file will be changed.

`FS004` in Figure 17-1: Example of a system configuration using the cold standby configuration is the file system where the audit trail directory is to be created. The VG name is to be `vg_hadb04`. The LV name is to be `hadb_audit`.

- `disk`
  Add the absolute path of the VG that contains the file system where the audit trail directory is to be created. For the example in Figure 17-1: Example of a system configuration using the cold standby configuration, add `/dev/vg_hadb04` to the `disk` operand.
- `fs_name`

Add the absolute path of the logical volume that corresponds to the file system where the audit trail directory is to be created. For the example in Figure 17-1: Example of a system configuration using the cold standby configuration, add `/dev/vg_hadb04/hadb_audit` to the `fs_name` operand.

- `fs_mount_dir`

  Add the mount directory path of the file system where the audit trail directory is to be created. Make sure that you configure the settings so that the active and standby systems both use the same path.

- `fs_mount_opt`

  Add the mount option for mounting the file system where the audit trail directory is to be created.

- `vg_neck`

  Add a `use` specification.

- `fs_neck`

  Add a `use` specification.

- `scsi_device`

  In this operand, specify the absolute paths of the device names that are to be subject to SCSI reservation.

  When using host reset, you do not need to specify this operand.

  When you use SCSI reservation for shared disk and the shared disk meets one of the following conditions, add to this operand an absolute path for the device name of the file system that stores audit trail files.

  - The shared disk is in a single-path configuration.
  - The shared disk is in a VMware ESXi virtual environment (excluding situations where DMMP is used).
  - The shared disk is in a redundant configuration realized using multipath software (HDLM).

- `dmmp_device`

  In this operand, specify the absolute paths of the logical DMMP devices that are to be subject to SCSI reservation.

  When using host reset, you do not need to specify this operand.

  When using SCSI reservation for shared disk and the shared disk uses a redundant configuration using multipath software (DMMP), add to this operand an absolute path for the device name of the file system that stores audit trail files.

For an example of specifying the `servers` file, see (b) servers file specification examples under (5) File specification examples (when using host reset) in 17.3.4 Setting up an HA Monitor environment.

> **Note**
>
> After terminating the cold standby configuration, change the values specified for the operands in the `servers` file. For details about how to terminate a cold standby configuration, see 17.4.2 How to terminate the cold standby configuration.

17.19.4 Linkage with JP1/Audit Management - Manager (in the case of the cold standby configuration)

To collect and centrally manage audit trails by linking with JP1/Audit, use the `adbconvertaudittrailfile` command to convert audit trails, and then output the conversion results to a common format audit trail file. For details about conversion of audit trail information, see 2.18.9 Conversion of audit trail information (linkage with JP1/Audit Management - Manager).

■ **Building the file system where the output-directory for common format audit trails will be created**

When building the file system where the output-directory for common format audit trails will be created, make sure that you build the file system as a host's local file system.

■ **Specifying the environment settings of JP1/Audit**

Specify the environment settings of JP1/Audit for all hosts on which the `adbconvertaudittrailfile` command is executed.

For details about the environment settings of JP1/Audit, see (2) Environment settings for JP1/Audit in 12.8.3 Environment settings for linking the audit trail facility with JP1/Audit. For details about the `adbconvertaudittrailfile` command, see *adbconvertaudittrailfile (Convert the Audit Trail File)* in the manual *HADB Command Reference*.

You do not need to set up JP1/Base in a cluster environment.

17.20 Operation when using the updated-row columnizing facility in a cold standby configuration

For details about the updated-row columnizing facility, see 11.18 Using the updated-row columnizing facility (maintaining the retrieval performance for column store tables). This section provides notes on using the updated-row columnizing facility in a cold standby configuration.

- The status of the updated-row columnizing facility (whether the facility is enabled or disabled) is maintained after a host switchover occurs.
- If a host switchover occurs when the updated-row columnizing facility is enabled, the maintenance processing is performed after the host switchover is completed.
- If you leave the updated-row columnizing facility enabled without handling any errors related to that facility, the maintenance processing that is performed after the host switchover is complete will result in an error, and an error message will be output.
- If no maintenance processing is performed for 24 hours consecutively, the `KFAA51277-I` message is output. However, the monitoring timer is reset when a host switchover occurs. Therefore, if no maintenance processing is performed for 24 hours consecutively again after a host switchover occurs, the `KFAA51277-I` message is output again.
- When a host switchover occurs, the processing that starts I/O control of files used by the updated-row columnizing facility is performed on the host that becomes the new active system. If this processing fails, the maintenance processing that is performed after the host switchover is complete will result in an error.

Appendixes

A. HADB Server Directory Configuration

This appendix explains the configuration of the HADB server's server directory (for installation), server directory (for operation), and DB directory.

A.1 Configuration of the server directory (for installation)

The following table shows the configuration of the server directory when the HADB server is installed.

These directories and files are created at installation of the HADB server. They are deleted when the HADB server is uninstalled.

Table A-1: Configuration of the server directory (for installation)

No.	Directory name or file name	Description
1	`$INSTDIR/bin`	Directory that stores various executable commands
2	`$INSTDIR/bin/adbarchivechunk`	Command for archiving chunks
3	`$INSTDIR/bin/adbaudittrail`	Command for managing the audit trail facility
4	`$INSTDIR/bin/adbcancel`	Cancel command
5	`$INSTDIR/bin/adbchgchunkcomment`	Command for entering, changing, and deleting comments for chunks
6	`$INSTDIR/bin/adbchgchunkstatus`	Command for changing a chunk status
7	`$INSTDIR/bin/adbchgnodetype`	Command for changing an HADB server's node type
8	`$INSTDIR/bin/adbchgsqltrc`	Command for starting and stopping output of SQL trace information
9	`$INSTDIR/bin/adbchgsrvmode`	Command for changing the HADB server operation mode
10	`$INSTDIR/bin/adbclientdefmang`	Command for centrally managing client definitions
11	`$INSTDIR/bin/adbcolumnize`	Command for managing the updated-row columnizing facility
12	`$INSTDIR/bin/adbconvertaudittrailfile`	Command for converting audit trail files
13	`$INSTDIR/bin/adbdbstatus`	DB status analysis command
14	`$INSTDIR/bin/adbexport`	Data export command
15	`$INSTDIR/bin/adbgetcst`	Cost information collection command
16	`$INSTDIR/bin/adbidxrebuild`	Index rebuild command
17	`$INSTDIR/bin/adbimport`	Data import command
18	`$INSTDIR/bin/adbinfoget`	Troubleshooting information collection command
19	`$INSTDIR/bin/adbinfosweep`	Troubleshooting information deletion command
20	`$INSTDIR/bin/adbinit`	Database initialization command
21	`$INSTDIR/bin/adbls`	HADB server status display command
22	`$INSTDIR/bin/adbmergechunk`	Command for merging chunks
23	`$INSTDIR/bin/adbmodarea`	DB area addition/modification command
24	`$INSTDIR/bin/adbmodbuff`	Command for changing a buffer

No.	Directory name or file name	Description
25	$INSTDIR/bin/adbmonitor	Server-monitoring command
26	$INSTDIR/bin/adbreorgsystemdata	Command for reorganizing system tables
27	$INSTDIR/bin/adbshmdump	Shared memory dump command
28	$INSTDIR/bin/adbsrvd	Server daemon
29	$INSTDIR/bin/adbstart	HADB server startup command
30	$INSTDIR/bin/adbstat	Statistical analysis command
31	$INSTDIR/bin/adbstop	HADB server termination command
32	$INSTDIR/bin/adbsyndict	Command for managing synonym dictionaries
33	$INSTDIR/bin/adbsystoru	Command for getting ITRU troubleshooting information
34	$INSTDIR/bin/adbtdmpget	Used by the system
35	$INSTDIR/bin/adbtdmpview	
36	$INSTDIR/bin/adbterrno	
37	$INSTDIR/bin/adbtgetdef	
38	$INSTDIR/bin/adbtgetsqltrc	
39	$INSTDIR/bin/adbthdlinfo	
40	$INSTDIR/bin/adbtmsglog	
41	$INSTDIR/bin/adbtstat	
42	$INSTDIR/bin/adbtsyslog	
43	$INSTDIR/bin/adbtmk2	
44	$INSTDIR/bin/adbunarchivechunk	Command for unarchiving chunks
45	$INSTDIR/bin/linaphx64	Used by the system
46	$INSTDIR/bin/patrol_sby_exe	HA Monitor slave monitor command (Start)
47	$INSTDIR/bin/patrol_sby_term	HA Monitor slave monitor command (Stop)
48	$INSTDIR/bin/stop_sby	HA Monitor slave stop command
49	$INSTDIR/client	Directory that stores commands and libraries used by HADB clients
50	$INSTDIR/client/bin	Directory that stores various executable commands used by HADB clients
51	$INSTDIR/client/bin/adbsql	SQL execution command
52	$INSTDIR/client/lib	Directory that stores various types of libraries used by HADB clients
53	$INSTDIR/client/lib/libadbclt.so	Client library
54	$INSTDIR/client/lib/adbjdbc8.jar	JDBC driver (JRE version 8)
55	$INSTDIR/conf	Directory that stores definition files
56	$INSTDIR/include	Directory that stores user-provided header files
57	$INSTDIR/include/adbcli.h	User-provided header files
58	$INSTDIR/include/adbtypes.h	

No.	Directory name or file name	Description
59	$INSTDIR/include/adbcnv.h	
60	$INSTDIR/include/adbodb.h	
61	$INSTDIR/lib	Directory that stores various types of libraries
62	$INSTDIR/lib/libadb.so	Database access control library
63	$INSTDIR/lib/libadbrpcs.so	Communication library
64	$INSTDIR/lib/libadbsort.so	Sort library
65	$INSTDIR/lib/libtapa.so	Used by the system
66	$INSTDIR/lib/adbmsg.cat	Message catalog file
67	$INSTDIR/lib/sysdef	Directory that stores definition analysis information files
68	$INSTDIR/lib/sysdef/adb.def	Definition analysis information files
69	$INSTDIR/lib/sysdef/adbarchivechunk.def	
70	$INSTDIR/lib/sysdef/adbclientdefmang.def	
71	$INSTDIR/lib/sysdef/adbclt.def	
72	$INSTDIR/lib/sysdef/adbexport.def	
73	$INSTDIR/lib/sysdef/adbgetcst.def	
74	$INSTDIR/lib/sysdef/adbidxrebuild.def	
75	$INSTDIR/lib/sysdef/adbimport.def	
76	$INSTDIR/lib/sysdef/adbimport_clm.def	
77	$INSTDIR/lib/sysdef/adbimport_clm_csv.def	
78	$INSTDIR/lib/sysdef/adbinit.def	
79	$INSTDIR/lib/sysdef/adbmergechunk.def	
80	$INSTDIR/lib/sysdef/adbmodarea.def	
81	$INSTDIR/lib/sysdef/adbmodbuff.def	
82	$INSTDIR/lib/sysdef/adbreorgsystemdata.def	
83	$INSTDIR/lib/sysdef/adbunarchivechunk.def	
84	$INSTDIR/sample	Directory that stores sample application programs and definition file templates
85	$INSTDIR/sample/cli_sample1.c	Sample application program (for CLI functions)
86	$INSTDIR/sample/create_sampledb.sh	Shell script for creating the table SAMPLE
87	$INSTDIR/sample/reorg_column_sample.sh	Shell script for reorganizing a column store table
88	$INSTDIR/sample/SAMPLE.txt	Data to be stored in the table SAMPLE
89	$INSTDIR/sample/SAMPLE_table.sql	File in which definition SQL statements for defining the table SAMPLE are coded
90	$INSTDIR/sample/Sample1.java	Sample application program (for JDBC)
91	$INSTDIR/sample/odbc_sample1.c	Sample application program (for ODBC)

No.	Directory name or file name	Description
92	$INSTDIR/sample/conf	Directory that stores a definition file template
93	$INSTDIR/sample/conf/adbarchivechunk.opt	Template file for the archive chunk option
94	$INSTDIR/sample/conf/adbclientdefmang.def	Template file for client-managing definitions
95	$INSTDIR/sample/conf/adbexport.opt	Export option template file
96	$INSTDIR/sample/conf/adbgetcst.opt	Template file for the cost-information collection option
97	$INSTDIR/sample/conf/adbidxrebuild.opt	Template file for the index rebuild option
98	$INSTDIR/sample/conf/adbimport.opt	Import option template file
99	$INSTDIR/sample/conf/adbinit.opt	Initialization option template file
100	$INSTDIR/sample/conf/adbmergechunk.opt	Template file for the merge chunk option
101	$INSTDIR/sample/conf/adbmodarea.opt	Template file for the DB area addition/modification option
102	$INSTDIR/sample/conf/adbmodbuff.opt	Template file for the buffer-modifying option
103	$INSTDIR/sample/conf/adbunarchivechunk.opt	Template file for the unarchive chunk option
104	$INSTDIR/sample/conf/client.def	Client definition template file
105	$INSTDIR/sample/conf/server.def	Server definition template file
106	$INSTDIR/sample/scripts	Directory that stores sample files for HA Monitor commands
107	$INSTDIR/sample/scripts/multinode.env	Environment variable definition file for HA Monitor commands (multi-node function)
108	$INSTDIR/sample/scripts/multinode_user.sh	Sample shell file for HA Monitor user commands (multi-node function)
109	$INSTDIR/sample/scripts/multinode_act.sh	Sample shell file for the HA Monitor startup command (multi-node function)
110	$INSTDIR/sample/scripts/multinode_term.sh	Sample shell file for the HA Monitor termination command (multi-node function) (for host reset)
111	$INSTDIR/sample/scripts/multinode_term_scsi.sh	Sample shell file for the HA Monitor termination command (multi-node function) (SCSI reservation for shared disk)
112	$INSTDIR/sample/scripts/multinode_patrol.sh	Sample shell file for the HA Monitor monitoring command (multi-node function)
113	$INSTDIR/sample/scripts/coldstandby.env	Environment variable definition file for HA Monitor commands (cold standby configuration)
114	$INSTDIR/sample/scripts/coldstandby_act.sh	Sample shell file for the HA Monitor startup command (cold standby configuration)
115	$INSTDIR/sample/scripts/coldstandby_term.sh	Sample shell file for the HA Monitor termination command (cold standby configuration)
116	$INSTDIR/sample/scripts/coldstandby_patrol.sh	Sample shell file for the HA Monitor monitoring command (cold standby configuration)
117	$INSTDIR/sample/jp1audit	Directory that stores the sample file of the definition file for JP1/Audit.
118	$INSTDIR/sample/jp1audit/admjevlog_HADB.conf	Sample file of the definition file for operational behavior for JP1/Audit
119	$INSTDIR/sample/jp1audit/HADB.conf	Sample file of the definition file for product behavior for JP1/Audit

No.	Directory name or file name	Description
120	$INSTDIR/spool	Directory that stores execution result logs

A.2 Configuration of the server directory (for operation)

The following table shows the configuration of the server directory used in the operation of the HADB server.

Table A-2: Configuration of the server directory (for operation)

No.	Directory name or file name	Description	Timing of creation	Timing of deletion
1	$ADBDIR/bin	Directory that stores various executable commands	When HADB server is installed	When HADB server is uninstalled
2	$ADBDIR/bin/adbarchivechunk	Command for archiving chunks		
3	$ADBDIR/bin/adbaudittrail	Command for managing the audit trail facility		
4	$ADBDIR/bin/adbcancel	Cancel command		
5	$ADBDIR/bin/adbchgchunkcomment	Command for entering, changing, and deleting comments for chunks		
6	$ADBDIR/bin/adbchgchunkstatus	Command for changing a chunk status		
7	$ADBDIR/bin/adbchgnodetype	Command for changing an HADB server's node type		
8	$ADBDIR/bin/adbchgsqltrc	Command for starting and stopping output of SQL trace information		
9	$ADBDIR/bin/adbchgsrvmode	Command for changing the HADB server operation mode		
10	$ADBDIR/bin/adbclientdefmang	Command for centrally managing client definitions		
11	$ADBDIR/bin/adbcolumnize	Command for managing the updated-row columnizing facility		
12	$ADBDIR/bin/adbconvertaudittrailfile	Command for converting audit trail files		
13	$ADBDIR/bin/adbdbstatus	DB status analysis command		
14	$ADBDIR/bin/adbexport	Data export command		
15	$ADBDIR/bin/adbgetcst	Cost information collection command		
16	$ADBDIR/bin/adbidxrebuild	Index rebuild command		
17	$ADBDIR/bin/adbimport	Data import command		

No.	Directory name or file name	Description	Timing of creation	Timing of deletion
18	$ADBDIR/bin/adbinfoget	Troubleshooting information collection command		
19	$ADBDIR/bin/adbinfosweep	Troubleshooting information deletion command		
20	$ADBDIR/bin/adbinit	Database initialization command		
21	$ADBDIR/bin/adbls	HADB server status display command		
22	$ADBDIR/bin/adbmergechunk	Command for merging chunks		
23	$ADBDIR/bin/adbmodarea	DB area addition/ modification command		
24	$ADBDIR/bin/adbmodbuff	Command for changing a buffer		
25	$ADBDIR/bin/adbmonitor	Server-monitoring command		
26	$ADBDIR/bin/ adbreorgsystemdata	Command for reorganizing system tables		
27	$ADBDIR/bin/adbshmdump	Shared memory dump command		
28	$ADBDIR/bin/adbsrvd	Server daemon		
29	$ADBDIR/bin/adbstart	HADB server startup command		
30	$ADBDIR/bin/adbstat	Statistical analysis command		
31	$ADBDIR/bin/adbstop	HADB server termination command		
32	$ADBDIR/bin/adbsyndict	Command for managing synonym dictionaries		
33	$ADBDIR/bin/adbsystoru	Command for getting ITRU troubleshooting information		
34	$ADBDIR/bin/adbtdmpget	Used by the system		
35	$ADBDIR/bin/adbtdmpview			
36	$ADBDIR/bin/adbterrno			
37	$ADBDIR/bin/adbtgetdef			
38	$ADBDIR/bin/adbtgetsqltrc			
39	$ADBDIR/bin/adbthdlinfo			
40	$ADBDIR/bin/adbtmsglog			
41	$ADBDIR/bin/adbtstat			
42	$ADBDIR/bin/adbtsyslog			

No.	Directory name or file name	Description	Timing of creation	Timing of deletion
43	$ADBDIR/bin/adbtmk2			
44	$ADBDIR/bin/adbunarchivechunk	Command for unarchiving chunks		
45	$ADBDIR/bin/linaphx64	Used by the system		
46	$ADBDIR/bin/patrol_sby_exe	HA Monitor slave monitor command (Start)		
47	$ADBDIR/bin/patrol_sby_term	HA Monitor slave monitor command (Stop)		
48	$ADBDIR/bin/stop_sby	HA Monitor slave stop command		
49	$ADBDIR/client	Directory that stores commands and libraries used by HADB clients		
50	$ADBDIR/client/bin	Directory that stores various executable commands used by HADB clients		
51	$ADBDIR/client/bin/adbsql	SQL execution command		
52	$ADBDIR/client/lib	Directory that stores various types of libraries used by HADB clients		
53	$ADBDIR/client/lib/libadbclt.so	Client library		
54	$ADBDIR/client/lib/adbjdbc8.jar	JDBC driver (JRE version 8)		
55	$ADBDIR/conf	Directory that stores definition files		
56	$ADBDIR/include	Directory that stores user-provided header files		
57	$ADBDIR/include/adbcli.h	User-provided header files		
58	$ADBDIR/include/adbtypes.h			
59	$ADBDIR/include/adbcnv.h			
60	$ADBDIR/include/adbodb.h			
61	$ADBDIR/lib	Directory that stores various types of libraries		
62	$ADBDIR/lib/libadb.so	Database access control library		
63	$ADBDIR/lib/libadbrpcs.so	Communication library		
64	$ADBDIR/lib/libadbsort.so	Sort library		
65	$ADBDIR/lib/libtapa.so	Used by the system		
66	$ADBDIR/lib/adbmsg.cat	Message catalog file		
67	$ADBDIR/lib/sysdef	Directory that stores definition analysis information files		

No.	Directory name or file name	Description	Timing of creation	Timing of deletion
68	$ADBDIR/lib/sysdef/adb.def	Definition analysis information files		
69	$ADBDIR/lib/sysdef/adbarchivechunk.def			
70	$ADBDIR/lib/sysdef/adbclientdefmang.def			
71	$ADBDIR/lib/sysdef/adbclt.def			
72	$ADBDIR/lib/sysdef/adbexport.def			
73	$ADBDIR/lib/sysdef/adbgetcst.def			
74	$ADBDIR/lib/sysdef/adbidxrebuild.def			
75	$ADBDIR/lib/sysdef/adbimport.def			
76	$ADBDIR/lib/sysdef/adbimport_clm.def			
77	$ADBDIR/lib/sysdef/adbimport_clm_csv.def			
78	$ADBDIR/lib/sysdef/adbinit.def			
79	$ADBDIR/lib/sysdef/adbmergechunk.def			
80	$ADBDIR/lib/sysdef/adbmodarea.def			
81	$ADBDIR/lib/sysdef/adbmodbuff.def			
82	$ADBDIR/lib/sysdef/adbreorgsystemdata.def			
83	$ADBDIR/lib/sysdef/adbunarchivechunk.def			
84	$ADBDIR/sample	Directory that stores sample application programs and definition file templates		
85	$ADBDIR/sample/cli_sample1.c	Sample application program (for CLI functions)		
86	$ADBDIR/sample/create_sampledb.sh	Shell script for creating the table SAMPLE		
87	$ADBDIR/sample/reorg_column_sample.sh	Shell script for reorganizing a column store table		
88	$ADBDIR/sample/SAMPLE.txt	Data to be stored in the table SAMPLE		

No.	Directory name or file name	Description	Timing of creation	Timing of deletion
89	$ADBDIR/sample/SAMPLE_table.sql	File in which definition SQL statements for defining the table SAMPLE are coded		
90	$ADBDIR/sample/Sample1.java	Sample application program (for JDBC)		
91	$ADBDIR/sample/odbc_sample1.c	Sample application program (for ODBC)		
92	$ADBDIR/sample/conf	Directory that stores a definition file template		
93	$ADBDIR/sample/conf/adbarchivechunk.opt	Template file for the archive chunk option		
94	$ADBDIR/sample/conf/adbclientdefmang.def	Template file for client-managing definitions		
95	$ADBDIR/sample/conf/adbexport.opt	Export option template file		
96	$ADBDIR/sample/conf/adbgetcst.opt	Template file for the cost-information collection option		
97	$ADBDIR/sample/conf/adbidxrebuild.opt	Template file for the index rebuild option		
98	$ADBDIR/sample/conf/adbimport.opt	Import option template file		
99	$ADBDIR/sample/conf/adbinit.opt	Initialization option template file		
100	$ADBDIR/sample/conf/adbmergechunk.opt	Template file for the merge chunk option		
101	$ADBDIR/sample/conf/adbmodarea.opt	Template file for the DB area addition/modification option		
102	$ADBDIR/sample/conf/adbmodbuff.opt	Template file for the buffer-modifying option		
103	$ADBDIR/sample/conf/adbunarchivechunk.opt	Template file for the unarchive chunk option		
104	$ADBDIR/sample/conf/client.def	Client definition template file		
105	$ADBDIR/sample/conf/server.def	Server definition template file		
106	$ADBDIR/sample/scripts	Directory that stores sample files for HA Monitor commands		
107	$ADBDIR/sample/scripts/multinode.env	Environment variable definition file for HA Monitor commands (multi-node function)		

No.	Directory name or file name	Description	Timing of creation	Timing of deletion
108	`$ADBDIR/sample/scripts/multinode_user.sh`	Sample shell file for HA Monitor user commands (multi-node function)		
109	`$ADBDIR/sample/scripts/multinode_act.sh`	Sample shell file for the HA Monitor startup command (multi-node function)		
110	`$ADBDIR/sample/scripts/multinode_term.sh`	Sample shell file for the HA Monitor termination command (multi-node function) (for host reset)		
111	`$ADBDIR/sample/scripts/multinode_term_scsi.sh`	Sample shell file for the HA Monitor termination command (multi-node function) (SCSI reservation for shared disk)		
112	`$ADBDIR/sample/scripts/multinode_patrol.sh`	Sample shell file for the HA Monitor monitoring command (multi-node function)		
113	`$ADBDIR/sample/scripts/coldstandby.env`	Environment variable definition file for HA Monitor commands (cold standby configuration)		
114	`$ADBDIR/sample/scripts/coldstandby_act.sh`	Sample shell file for the HA Monitor startup command (cold standby configuration)		
115	`$ADBDIR/sample/scripts/coldstandby_term.sh`	Sample shell file for the HA Monitor termination command (cold standby configuration)		
116	`$ADBDIR/sample/scripts/coldstandby_patrol.sh`	Sample shell file for the HA Monitor monitoring command (cold standby configuration)		
117	`$ADBDIR/sample/jp1audit`	Directory that stores the sample file of the definition file for JP1/Audit.		
118	`$ADBDIR/sample/jp1audit/admjevlog_HADB.conf`	Sample file of the definition file for operational behavior for JP1/Audit		
119	`$ADBDIR/sample/jp1audit/HADB.conf`	Sample file of the definition file for product behavior for JP1/Audit		
120	`$ADBDIR/spool`	Directory that stores execution result logs		
121	`$ADBDIR/spool/adbdump`*YYYYMMDDhhmmss*`.`*server-process-process-id*[#1]	HADB dump file	When HADB server is started	• When HADB server is terminated normally • When `adbinfosweep` command is executed

No.	Directory name or file name	Description	Timing of creation	Timing of deletion
122	$ADBDIR/spool/adbdumperror*YYYYMMDDhhmmssSSSSSS_TTTTTTTTTTTTTTTTTT.server-process-process-id*[#1, #2]		When an error occurs for which the KFAA60007-E message is output	When adbinfosweep command is executed
123	$ADBDIR/spool/adbmessage*XX*.log[#3]	Server message log file	When HADB server is started	
124	$ADBDIR/spool/adbsqltrc*XX*.log[#4]	SQL trace file		When HADB server is uninstalled
125	$ADBDIR/spool/adbstatlog*XX*[#3, #5]	Statistics log file		
126	$ADBDIR/spool/.defrslt	Server definition storage file		When adbinfosweep command is executed
127	$ADBDIR/spool/.adbmessage	Message log file number control file		
128	$ADBDIR/spool/.adbshmid	Shared memory ID storage file		
129	$ADBDIR/spool/.adbsemid	Semaphore set ID storage file		
130	$ADBDIR/spool/.adbsqltrc	File for managing the file numbers of SQL trace files		When HADB server is uninstalled
131	$ADBDIR/spool/.adbstatlog[#5]	Statistics log file control file		
132	$ADBDIR/spool/adbcolumnize01	File used by the updated-row columnizing facility		
133	$ADBDIR/spool/.adbcolumnize			
134	$ADBDIR/spool/adboptlog*XX*[#6]	Access path search information log file		
135	$ADBDIR/spool/.adboptlog	Management file for the access path search information log		
136	$ADBDIR/spool/.hadb_map	Used by the system		
137	$ADBDIR/spool/.hadb_ulimit			

#1

YYYYMMDDhhmmss in the file name indicates the time when the file was generated.

#2

SSSSSS in the file name indicates the microsecond portion of the time when the HADB dump file was generated. *TTTTTTTTTTTTTTTTTT* indicates the thread ID of the real thread in which the HADB server's internal conflict error was detected.

#3

XX in the file name is a sequential number between 01 and 04.

#4

XX in the file name is a sequential number between 01 and 08.

#5

If the adb_sta_log_path operand in the server definition is specified, the data is output to the directory specified in the adb_sta_log_path operand in the server definition.

For details about the adb_sta_log_path operand in the server definition, see the description of the adb_sta_log_path operand in 7.2.7 Operands related to statistical information (set format).

#6

XX in the file name is a sequential number that is either 01 or 02.

A.3 DB directory configuration

The following table shows the configuration of the DB directory that is created when the adbinit command is used to initialize the database.

Table A-3: DB directory configuration

No.	Directory name and file name	Description	Timing of creation	Timing of deletion
1	$DBDIR/ADBMST	Master directory DB area file	When adbinit command is executed	When adbinit command is executed (with existing file present)
2	$DBDIR/ADBDIC	Dictionary DB area file		
3	$DBDIR/ADBSTBL	System-table DB area file		
4	$DBDIR/ADBWRK	Work table DB area file		
5	$DBDIR/DBAREA[#1]	Data DB area file	• When adbinit command is executed • When adbmodarea command is executed (to add a data DB area) • When adbmodarea command is executed (to expand a data DB area)	• When adbinit command is executed (with existing file present) • When adbmodarea command is executed (to delete a data DB area)
6	$DBDIR/ADBSYS	System directory	When adbinit command is executed	When adbinit command is executed (with existing directory present)
7	$DBDIR/ADBSYS/ADBSLG	System log file directory		
8	$DBDIR/ADBSYS/ADBSTS	Status file directory		
9	$DBDIR/ADBSYS/ADBUTL	Command status file directory		
10	$DBDIR/ADBWORK	Work directory		
11	$DBDIR/SPOOL	Error information (core files) output destination directory[#2]		
12	$DBDIR/SPOOL/core.*server-process-process-ID*	Error information (core files)	When HADB server is terminated abnormally	When adbinfosweep command is executed

#1

The name specified by the user in the adbinit or adbmodarea command

#2

If the adb_core_path operand in the server definition is specified, error information (core files) is output to the directory specified in the adb_core_path operand instead of to the $DBDIR/SPOOL directory.

B. Dictionary Tables

This appendix lists the information stored in dictionary tables and the B-tree indexes defined for dictionary tables, and it explains how to search for information in dictionary tables.

B.1 Dictionary table overview

Dictionary tables store table and index definition information as well as DB area information.

HADB provides two types of dictionary tables, base tables and viewed tables.

- Dictionary table (base table)

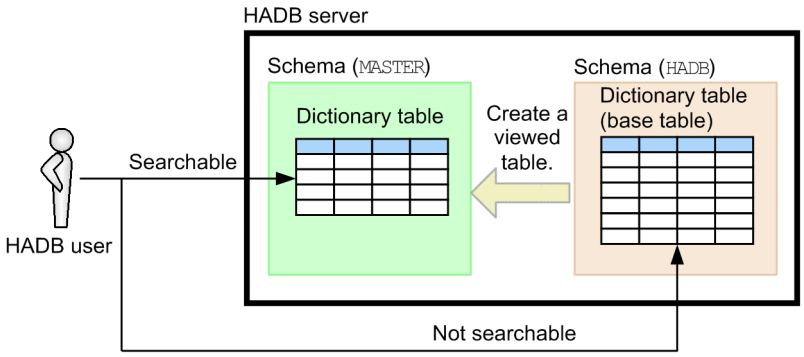
  In this manual, a dictionary table that is a base table is called a *dictionary table (base table)*. The schema name of a dictionary table (base table) is `HADB`.

- Dictionary table (viewed table)

  In this manual, a dictionary table that is a viewed table is called a *dictionary table*. The schema name of a dictionary table is `MASTER`.

The following figure shows the relationship between a dictionary table (base table) and a dictionary table.

Figure B-1: Relationship between a dictionary table (base table) and a dictionary table



Explanation:

Definition information for tables and indexes as well as DB area information is stored in dictionary tables (base tables). A viewed table that is created from a dictionary table (base table) for search purposes becomes a dictionary table.

HADB users can search dictionary tables, but not dictionary tables (base tables).

An HADB user can check definition information of tables and indexes and DB area information by using the `SELECT` statement to search dictionary tables. When you search dictionary tables, specify `MASTER` as the schema name.

(1) List of dictionary tables

The following table lists the dictionary tables.

Table B-1: List of dictionary tables

No.	Dictionary table name	Information stored	Table ID of the dictionary table (base table) that corresponds to the dictionary table	Number of subqueries included in the view definition#	Existence of external reference
1	SQL_TABLES	Stores table definition information of base tables, viewed tables, dictionary tables (base tables), system tables (base tables), dictionary tables, and system tables. Each row stores information for one table.	0x00020001	1	Y
2	SQL_COLUMNS	Stores column definition information of base tables, viewed tables, dictionary tables (base tables), system tables (base tables), dictionary tables, and system tables. Each row stores information for one column.	0x00020002	1	Y
3	SQL_DIV_TABLE	Stores information in locations such as DB areas that store base tables, dictionary tables (base tables), and system tables (base tables). Each row stores information for one table.	0x00020004	1	Y
4	SQL_INDEXES	Stores index definition information for base tables. This table also stores B-tree index definition information for dictionary tables (base tables) and system tables (base tables). Each row stores information for one index.	0x00020003	1	Y
5	SQL_DIV_INDEX	Stores information about DB areas that store indexes of base tables. This table also stores information about DB areas that store B-tree indexes for dictionary tables (base tables) and system tables (base tables). Each row stores information for one index.	0x00020005	1	Y
6	SQL_DBAREAS	Stores DB area definition information. Each row stores information for one DB area.	0x00020006	0	N
7	SQL_SCHEMATA	Stores schema-related definition information. Each row stores information for one schema.	0x00020007	1	Y
8	SQL_VIEWS	Stores viewed table definition information. Each row stores information for one viewed table.	0x00020008	1	Y

No.	Dictionary table name	Information stored	Table ID of the dictionary table (base table) that corresponds to the dictionary table	Number of subqueries included in the view definition#	Existence of external reference
9	SQL_VIEW_TABLE_USAGE	Stores the definition information of the underlying tables of a viewed table. Each row stores information for one underlying table.	0x00020009	1	Y
10	SQL_VIEW_OBJECT	Stores information that is used by the system. There is no dictionary table. There is only a dictionary table (base table).	0x0002000A	--	--
11	SQL_DEFINE_SOURCE	Stores SQL statement information. Each row stores information for one SQL text.	0x0002000B	2	Y
12	SQL_DEFINE_ENVIRONMENT	Stores the environment information that was in effect during view definition or when changing the view definition. Each row stores the environment information of one viewed table.	0x0002000C	2	Y
13	SQL_USERS	Stores HADB user information. Each row stores information on one HADB user.	0x0002000D	1	N
14	SQL_TABLE_CONSTRAINTS	Stores constraint information related to base tables. Each row stores information on one constraint.	0x0002000E	1	Y
15	SQL_INDEX_COLINF	Stores information related to indexed columns. Each row stores information on one indexed column.	0x0002000F	1	Y
16	SQL_KEY_COLUMN_USAGE	Stores information related to columns comprising the primary key and foreign keys. Each row stores information on one column comprising the primary key and foreign keys.	0x00020010	1	Y
17	SQL_REFERENTIAL_CONSTRAINTS	Stores information related to referential constraints. Each row stores information on one referential constraint.	0x00020011	1	Y
18	SQL_TABLE_PRIVILEGES	Stores information related to access privileges for tables. Each row stores one table's worth of information on one authorization identifier granted by a privilege grantor.	0x00020012	1	Y
19	SQL_AUDITS	Stores information related to audit target definitions defined	0x00020013	0	N

No.	Dictionary table name	Information stored	Table ID of the dictionary table (base table) that corresponds to the dictionary table	Number of subqueries included in the view definition#	Existence of external reference
		by the CREATE AUDIT statement. Each row stores information for one audit target definition.			

Legend:

Y: The dictionary table's viewed table includes a subquery that performs an external reference. Therefore, you cannot specify the corresponding dictionary table in a multiset value expression subquery.

N: The dictionary table's viewed table does not include a subquery that performs an external reference.

--: Not applicable.

#

If you specified the corresponding dictionary table in a SQL statement subquery, the number that must be added as the number of nests in that subquery is displayed.

Note

- A dictionary table also stores the table definition information and index definition information of the dictionary table.
- The data types of the dictionary tables follow HADB's SQL specification.

(2) Times at which a dictionary table is created

A dictionary table is automatically created at the following times:

- When the HADB server is started for the first time following the completion of database initialization
- When the HADB server version is upgraded

(3) Scope of information in dictionary tables that can be referenced by HADB users

The scope of information in dictionary tables that can be referenced by HADB users varies according to the privileges they have. The following table shows the dictionary table information that an HADB user can reference.

Note that even for viewed tables specified as an underlying table of the dictionary table, the information a user can reference varies according to the users' privileges. Therefore, even when searching the same viewed table, the HADB user who defined the viewed table and users granted access privileges for that viewed table might not be able to view the same information.

Table B-2: Dictionary table information that an HADB user can reference

Dictionary table name	Information being referenced	Privilege of HADB user who references dictionary table			
		DBA privilege	Audit admin privilege	Access privilege	CONNECT privilege
SQL_TABLES	Their own	Y	Y	Y	Y

Dictionary table name	Information being referenced	Privilege of HADB user who references dictionary table			
		DBA privilege	Audit admin privilege	Access privilege	CONNECT privilege
	Other user	Y	Y	C[#1]	N
	MASTER	Y	Y	Y	Y
SQL_COLUMNS	Their own	Y	Y	Y	Y
	Other user	Y	N	C[#1]	N
	MASTER	Y	Y	Y	Y
SQL_DIV_TABLE	Their own	Y	Y	Y	Y
	Other user	Y	N	C[#1]	N
	MASTER	--	--	--	--
SQL_INDEXES	Their own	Y	Y	Y	Y
	Other user	Y	Y	C[#1]	N
	MASTER	Y	Y	Y	Y
SQL_DIV_INDEX	Their own	Y	Y	Y	Y
	Other user	Y	N	C[#1]	N
	MASTER	--	--	--	--
SQL_DBAREAS	--	Y	Y	Y	Y
SQL_SCHEMATA	Their own	Y	Y	Y	Y
	Other user	Y	Y	C[#1]	N
	MASTER	Y	Y	Y	Y
SQL_VIEWS	Their own	Y	Y	Y	Y
	Other user	Y	N	C[#1, #3]	N
	MASTER	Y	Y	Y	Y
SQL_VIEW_TABLE_USAGE	Their own	Y	Y	Y	Y
	Other user	Y	Y	C[#1, #3]	N
	MASTER	Y	Y	Y	Y
SQL_DEFINE_SOURCE	Their own	Y	Y	Y	Y
	Other user	Y	N	C[#1, #3]	N
	MASTER	--	--	--	--
SQL_DEFINE_ENVIRONMENT	Their own	Y	Y	Y	Y
	Other user	Y	N	C[#1, #3]	N
	MASTER	Y	Y	Y	Y
SQL_USERS	Their own	Y	Y	--	Y
	Other user	Y	Y	--	N
	MASTER	--	--	--	--
SQL_TABLE_CONSTRAINTS	Their own	Y	Y	Y	Y

Dictionary table name	Information being referenced	Privilege of HADB user who references dictionary table			
		DBA privilege	Audit admin privilege	Access privilege	CONNECT privilege
	Other user	Y	N	C#1	N
	MASTER	--	--	--	--
SQL_INDEX_COLINF	Their own	Y	Y	Y	Y
	Other user	Y	N	C#1	N
	MASTER	--	--	--	--
SQL_KEY_COLUMN_USAGE	Their own	Y	Y	Y	Y
	Other user	Y	N	C#1	N
	MASTER	--	--	--	--
SQL_REFERENTIAL_CONSTRAINTS	Their own	Y	Y	Y	Y
	Other user	Y	N	C#1	N
	MASTER	--	--	--	--
SQL_TABLE_PRIVILEGES	Their own	C#2	C#2	C#2	C#2
	Other user	C#2	N	C#1, #2	N
	MASTER	--	--	--	--
SQL_AUDITS	Their own	N	Y	N	N
	Other user	N	Y	N	N
	MASTER	N	Y	N	N

Legend:

Their own: Information related to a schema or schema object for which the HADB user is the owner (his or her own HADB user information in the case of SQL_USERS).

Other user: Information related to a schema or schema object that another HADB user owns (the HADB user information of other HADB users in the case of SQL_USERS).

MASTER: Information related to dictionary tables and system tables.

Y: Can be referenced.

C: Can be referenced but conditions apply.

N: Cannot be referenced.

--: Not applicable.

#1

Users can reference information related to schema objects for which they have the access privilege, and the schema of those schema objects.

#2

When the target schema object is a viewed table that has been invalidated, users cannot retrieve information from SQL_TABLE_PRIVILEGES about access privileges for the invalidated viewed table.

#3

When the target schema object is a viewed table that has been invalidated, users cannot reference definition information related to the invalidated viewed table.

B.2 Content of SQL_TABLES

SQL_TABLES stores table definition information of base tables, viewed tables, dictionary tables, dictionary tables (base tables), system tables, and system tables (base tables). Each row stores information for one table.

The following table describes the content of SQL_TABLES.

Table B-3: Content of SQL_TABLES

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier
3	TABLE_TYPE	CHAR(1)	Table type: • 'R': Base table • 'V': Viewed table
4	TABLE_ID	SMALLINT	Table ID: • From 0x00020001: Dictionary tables (base tables) • From 0x000200C9: System tables (base tables) • From 0x00020191: Base tables defined by HADB users • From 0x00040001: Dictionary tables and system tables • From 0x00040401: Viewed tables defined by HADB users
5	N_COLS	SMALLINT	Number of columns in the table
6	N_INDEX	SMALLINT	Number of indexes defined for the table (sum of all indexes)
7	CREATE_TIME	TIMESTAMP	Datetime stamp indicating when the table was defined.
8	N_NOTNULL	SMALLINT	Number of columns for which the NOT NULL constraint is specified
9	FREE_AREA	SMALLINT	Percentage of unused area in the table pages
10	FIX_TABLE	CHAR(1)	Whether this is a FIX table: • 'F': FIX table • 'N': Not a FIX table
11	ROW_LENGTH	SMALLINT	FIX table row length. For dictionary tables (base tables), system tables (base tables), or tables that are not FIX tables, the null value is stored.
12	IS_BRANCH_ALL	CHAR(1)	Whether the BRANCH ALL table option is specified: • 'Y': A table with BRANCH ALL specified • 'N': A table without BRANCH ALL specified
13	N_RANGE_INDEX	SMALLINT	Number of range indexes defined for a table.
14	IS_CHUNK	CHAR(1)	Specification of CHUNK chunk specification • 'Y': Specified (multi-chunk table) • 'N': Not specified (single chunk table)

No.	Column name	Data type	Stored information
15	N_CHUNK_RESERVED	SMALLINT	Maximum number of chunks to be created This value stores a CHUNK specification value. In the case of single chunk tables and viewed tables, a null value is stored.
16	N_DEFAULT_COLUMN	SMALLINT	Number of columns for which the DEFAULT clause is specified Stores the number of columns for which the DEFAULT clause was specified when a base table was defined. For viewed tables, dictionary tables (base tables), system tables (base tables), and base tables without the DEFAULT clause specified, the null value is stored.
17	N_PRIMARY_KEY_COLUMN	SMALLINT	Number of columns comprising the primary key Stores the number of columns comprising the primary key that was defined when a base table was defined. For viewed tables, dictionary tables (base tables), system tables (base tables), and base tables without the primary key specified, the null value is stored.
18	N_FOREIGN_KEY	SMALLINT	Number of foreign keys Store the number of foreign keys defined for a base table. The null value is stored for the following tables: • Viewed table • Dictionary table (base table) • System table (base table) • Base table for which no foreign key is defined
19	N_REFERENCING_KEY	SMALLINT	Number of foreign keys that reference a primary key Stores the number of foreign keys that reference a primary key defined for this base table. The null value is stored for the following tables: • Viewed table • Dictionary table (base table) • System table (base table) • Base table that has no foreign key that references a primary key • Base table for which no primary key is defined
20	N_FOREIGN_KEY_COLUMN	SMALLINT	Number of columns comprising foreign keys Stores the total number of columns comprising the foreign keys defined for the base table. The null value is stored for the following tables: • Viewed table • Dictionary table (base table) • System table (base table) • Base table for which no foreign key is defined
21	N_TEXT_INDEX	SMALLINT	Number of text indexes defined for the table. The null value is stored for the following tables: • Viewed table • Dictionary table (base table) • System table (base table) • Table for which no text index is defined
22	IS_ARCHIVABLE	CHAR(1)	Is there a chunk archive specification?

No.	Column name	Data type	Stored information
			• 'Y' There is a chunk archive specification (it is an archivable multi-chunk table) • Null value There is no chunk archive specification (it is a base table other than an archivable multi-chunk table) The null value is stored for the following tables: • Viewed table • Dictionary table (base table) • System table (base table)
23	ARCHIVE_DIRECTORY_PATH	VARCHAR(400)	Absolute path of the archive directory The null value is stored for the following tables: • Viewed table • Dictionary table (base table) • System table (base table) • Base table other than an archivable multi-chunk table
24	RECREATE_TIME	TIMESTAMP	Execution date and time of the ALTER VIEW statement The time stamp of the latest ALTER VIEW statement execution for the viewed table is stored. The null value is stored for the following tables: • Base tables • Viewed tables for which an ALTER VIEW statement has not been executed
25	STORAGE_FORMAT	VARCHAR(32)	Table-data storage format • 'COLUMN' Column store format • 'ROW' Row store format In the case of a viewed table, a null value is stored.

B.3 Content of SQL_COLUMNS

SQL_COLUMNS stores column definition information of base tables, viewed tables, dictionary tables, dictionary tables (base tables), system tables, system tables (base tables). Each row stores information for one column.

The following table describes the content of SQL_COLUMNS.

Table B-4: Content of SQL_COLUMNS

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier
3	COLUMN_NAME	VARCHAR(100)	Column name
4	COLUMN_ID	SMALLINT	Column ID
5	TABLE_ID	SMALLINT	Table ID: • From 0x00020001: Dictionary tables (base tables)

No.	Column name	Data type	Stored information
			• From 0x000200C9: System tables (base tables) • From 0x00020191: Base tables defined by HADB users • From 0x00040001: Dictionary tables and system tables • From 0x00040401: Viewed tables defined by HADB users
6	DATA_TYPE_CODE	SMALLINT	Column data type: • 113 (0x71): DATE • 121 (0x79): TIME • 125 (0x7D): TIMESTAMP • 145 (0x91): VARBINARY • 149 (0x95): BINARY • 193 (0xC1): VARCHAR • 197 (0xC5): CHAR • 225 (0xE1): DOUBLE PRECISION • 229 (0xE5): DECIMAL • 241 (0xF1): INTEGER • 245 (0xF5): SMALLINT
7	DATA_LENGTH	SMALLINT	Column definition length: **When the column data type is one of those listed below** Contains actual column definition length, in bytes. • INTEGER • SMALLINT • CHAR • VARCHAR • DATE • DOUBLE PRECISION • BINARY • VARBINARY **When the column data type is TIME(*p*)** Stores the value determined using the following formula: $3 + \uparrow p \div 2 \uparrow$ *p*: 0, 3, 6, 9, or 12 **When the column data type is TIMESTAMP(*p*)** Stores the value determined using the following formula: $7 + \uparrow p \div 2 \uparrow$ *p*: 0, 3, 6, 9, or 12 **If the column data type is DECIMAL:** The first byte stores the scaling and the second byte stores the precision. If the selection expression of a CREATE VIEW statement's query expression body includes a value expression for which the division result of the arithmetic operation is the DECIMAL data type, the following two values might not match: • The scaling value stored here

No.	Column name	Data type	Stored information
			• The scaling value of the derived column of the internal table derived from the equivalent exchange of a viewed table, when searching a viewed table. For details, see *When searching a viewed table* in *Notes applying when the data type of the division result is DECIMAL* in the manual *HADB SQL Reference*.
8	IS_NULLABLE	CHAR(1)	Whether the NOT NULL constraint was specified for the column: • 'Y' Null values are permitted (NOT NULL constraint was not specified) • 'N' Null values are not permitted (NOT NULL constraint was specified) In the case of a FIX table, all columns default to 'N'.
9	COLUMN_OFFSET	SMALLINT	Column offset (a number indicating the position of the byte, counted from the beginning of the table, in which the column begins). The null value is stored for a table that is not a FIX table (table for which the BRANCH ALL option is not specified).
10	BRANCH	CHAR(1)	BRANCH specification value in column definition: • 'Y': YES • 'N': NO • 'A': AUTO • 'O': VARCHAR-type or VARBINARY-type column without BRANCH specified For columns whose type is not VARCHAR or VARBINARY, the null value is stored. The null value is also stored for columns in a column store table.
11	DEFAULT_VALUE	VARCHAR(32000)	Default value specified in the DEFAULT clause Stores the default value that was specified in the DEFAULT clause when a base table was defined. • Literal Stores a literal if a literal was specified. For details, see Table B-5: Value that is stored in the DEFAULT_VALUE column if a literal is specified. • CURRENT_DATE Stores this value when CURRENT_DATE is specified. • CURRENT_TIME(*p*) Stores this value when CURRENT_TIME(*p*) is specified. *p* shows this value in fractional seconds precision. When *p* = 0, CURRENT_TIME is stored. When *p* = 3, 6, 9, or 12, CURRENT_TIME(*p*) is stored. • CURRENT_TIMESTAMP(*p*) Stores this value when CURRENT_TIMESTAMP(*p*) is specified. *p* shows this value in fractional seconds precision. When *p* = 0, CURRENT_TIMESTAMP is stored.

No.	Column name	Data type	Stored information
			When *p* = 3, 6, 9, or 12, CURRENT_TIMESTAMP(*p*) is stored. • CURRENT_USER Stores this value when CURRENT_USER is specified. • Null value Stores this value when NULL is specified. For columns of viewed tables, columns of dictionary tables (base tables), columns of system tables (base tables), and columns of base tables without the DEFAULT clause specified, the null value is stored.
12	IS_DEFAULT_COLUMN	CHAR(1)	Whether the DEFAULT clause is specified • 'Y' Column for which the DEFAULT clause was specified when a base table was defined • Null value Column for which the DEFAULT clause was not specified when a base table was defined The null value is stored also for columns of viewed tables, columns of dictionary tables (base tables), and columns of system tables (base tables).
13	IS_PRIMARY_KEY_COLUMN	CHAR(1)	Whether a column comprises the primary key • 'Y' Column that comprises the primary key • Null value Column that does not comprise the primary key The null value is stored also for columns of viewed tables, columns of dictionary tables (base tables), and columns of system tables (base tables).
14	PRIMARY_KEY_COLUMN_SEQUENCE_NUMBER	SMALLINT	Primary key configuration sequence Stores the sequence of columns comprising the primary key in ascending order, starting with 1. The null value is stored for columns of viewed tables, columns of dictionary tables (base tables), columns of system tables (base tables), and columns of base tables that do not comprise the primary key.
15	N_FOREIGN_KEY_COLUMN	SMALLINT	Number of foreign keys that use this column Stores the number of foreign keys that use this column. The null value is stored for the following columns: • Column of a viewed table • Column of a dictionary table (base table) • Column of a system table (base table) • Column of a base table not comprising a foreign key
16	IS_ARCHIVE_RANGE_COLUMN	CHAR(1)	Is archive range column? • 'Y' Is archive range column • Null value Is not archive range column The null value is stored for the following columns: • Column of a viewed table • Column of a dictionary table (base table)

No.	Column name	Data type	Stored information
			• Column of a system table (base table)
17	COMPRESSION_TYPE	VARCHAR(32)	Value specified for COMPRESSION TYPE in column definition Stores the details specified for the compression-type specification when defining a column store table. The null value is stored for the following columns: • Column of a viewed table • Column of a dictionary table (base table) • Column of a system table (base table) • Column of a row store table

The following table describes the value that is stored in the DEFAULT_VALUE column when a literal is specified in the DEFAULT clause.

Table B-5: Value that is stored in the DEFAULT_VALUE column if a literal is specified

No.	Data type assumed for the default value		Value that is stored in the DEFAULT_VALUE column	
			Data length (bytes)	Example of character format for the value that is stored
1	Numeric data	INTEGER	Length of the specified default value that has been converted to character format	12345
2		SMALLINT		12345
3		DECIMAL		12.345
4		DOUBLE PRECISION		-1234567890.1234567E-123
5	Character string data	CHAR	*Length of the specified default value that has been converted to character format* + 2	'ABCD'
6		VARCHAR		
7	Datetime data	DATE	16	DATE'*YYYY-MM-DD*'
8		TIME(*p*)	• If *p* is zero, 14 • If *p* is not zero, 15 + *p*	TIME'*hh*:*mm*:*ss*.*nn...n*'[#1, #2]
9		TIMESTAMP(*p*)	• If *p* is zero, 30 • If *p* is not zero, 31 + *p*	TIMESTAMP'*YYYY-MM-DD*Δ*hh*:*mm*:*ss*.*nn....n*'[#1,#2]
10	Binary data	BINARY	• Binary data in binary format, *Data length of the specified default value* × 8 + 3 • Binary data in hexadecimal format, *Data length of the specified default value* × 2 + 3	• Binary data in binary format B'01010101' • Binary data in hexadecimal format X'010203'
11		VARBINARY		

Legend:

p: Fractional seconds precision (0, 3, 6, 9, or 12)

YYYY: Year

MM: Month

DD: Date

Δ: Space

hh: Hour

mm: Minute

ss: Second

nn...n: Fraction of a second

#1

If the fractional seconds precision of the column for which the DEFAULT clause is specified is zero, the period (.) between the second value (*ss*) and the fraction of a second (*nn...n*) is omitted.

#2

If the fractional seconds precision of the default value is shorter than the number of digits in the fraction of a second (*nn...n*) for the column for which the DEFAULT clause is specified, trailing zeros are padded to fill the length.

If the fractional seconds precision of the default value is greater than the number of digits in the fraction of a second (*nn...n*) for the column for which the DEFAULT clause is specified, the value is truncated.

B.4 Content of SQL_DIV_TABLE

SQL_DIV_TABLE stores information on locations such as DB areas that store base tables, dictionary tables (base tables), and system tables (base tables). Each row stores information for one table.

The following table describes the content of SQL_DIV_TABLE.

Table B-6: Content of SQL_DIV_TABLE

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier
3	TABLE_ID	SMALLINT	Table ID: • From 0x00020001: Dictionary tables (base tables) • From 0x000200C9: System tables (base tables) • From 0x00020191: Base tables defined by HADB users
4	DBAREA_ID	SMALLINT	DB area ID of the DB area that stores the table

B.5 Content of SQL_INDEXES

SQL_INDEXES stores index definition information for base tables. It also stores B-tree index definition information for dictionary tables (base tables) and system tables (base tables). Each row stores information for one index.

The following table describes the content of SQL_INDEXES.

Table B-7: Content of SQL_INDEXES

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name

No.	Column name	Data type	Stored information
2	TABLE_NAME	VARCHAR(100)	Table identifier of the table for which the index is defined
3	INDEX_NAME	VARCHAR(100)	Index identifier In the case of an index that corresponds to the primary key, an index identifier having the same name as the constraint name is stored.
4	INDEX_ID	SMALLINT	Index ID: • From 0x00030001: Indexes of dictionary tables (base tables) • From 0x00030191: Indexes of system tables (base tables) • From 0x00030321: Indexes of base tables defined by HADB users
5	TABLE_ID	SMALLINT	Table ID of the table for which the index is defined: • From 0x00020001: Dictionary table (base table) • From 0x000200C9: System table (base table) • From 0x00020191: Base tables defined by HADB users
6	COLUMN_COUNT	SMALLINT	Number of indexed columns
7	COLUMN_ID_LIST	VARCHAR(32)	List of indexed column IDs#
8	CREATE_TIME	TIMESTAMP	Datetime stamp for when the index was defined
9	UNIQUE_TYPE	CHAR(1)	B-tree index type (whether the index is unique): • 'Y': Unique index • 'N': Non-unique index
10	FREE_AREA	SMALLINT	Percentage of unused page area in an index. For a range index, the null value is stored.
11	INDEX_KEY_EXPAND	CHAR(1)	Whether the sum total of the sizes of the columns comprising the index is at least 256 bytes: • 'Y' The sum total of the sizes of the columns comprising the index is at least 256 bytes. • 'N' The sum total of the sizes of the columns comprising the index is 255 bytes or less.
12	INDEX_TYPE	CHAR(1)	Index type: • 'B': B-tree index • 'T': Text index • 'R': Range index
13	IS_PRIMARY_KEY	CHAR(1)	Whether an index corresponds to the primary key • 'Y' Index that corresponds to the primary key • Null value Index that does not correspond to the primary key The null value is also stored for indexes of dictionary tables (base tables) and indexes of system tables (base tables).
14	IS_CHUNK_SKIP	CHAR(1)	Whether a range index can skip chunks • 'Y' Range index that can skip chunks

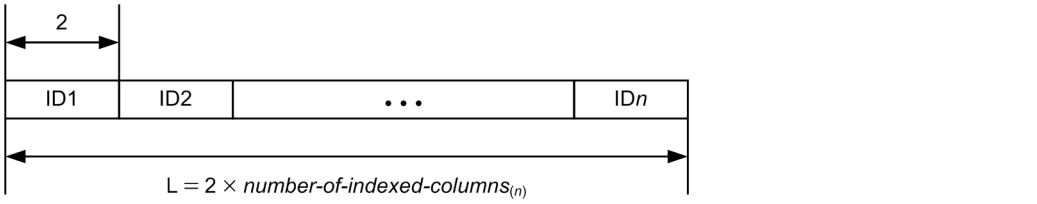
No.	Column name	Data type	Stored information
			• Null value Range index that cannot skip chunks The null value is stored for the following indexes: • Indexes other than range indexes • Range indexes that have been defined by a version earlier than 02-02
15	IS_EXCLUDE_NULL_VALUES	CHAR(1)	Whether null-value exclusion is specified • 'Y' Index for which null-value exclusion is specified • Null value Index for which null-value exclusion is not specified
16	IS_ARCHIVE_RANGE	CHAR(1)	Is range index defined automatically by HADB server? • 'Y' Range index defined for the archive range column automatically by HADB server • Null value Range index that does not meet the conditions of 'Y' However, if an archivable multi-chunk table is changed to a regular multi-chunk table, a null value is stored even for range indexes automatically defined by the HADB server. A null value is also stored for the following indexes: • Indexes of dictionary tables (base tables) • Indexes of system tables (base tables)
17	IS_TEXT_CORRECTION_RULE	CHAR(1)	Is there a notation-correction-search text-index specification? • 'Y' Text index that has a notation-correction-search text-index specification • Null value Text index that does not have a notation-correction-search text-index specification A null value is also stored for the following indexes: • Indexes of dictionary tables (base tables) • Indexes of system tables (base tables) • Indexes other than text indexes
18	IS_TEXT_WORDCONTEXT	CHAR(1)	Whether there is a text-index-word-context search specification • 'Y' Text index that has a text-index-word-context search specification • Null value Text index that does not have a text-index-word-context search specification A null value is also stored for the following indexes: • Indexes of dictionary tables (base tables) • Indexes of system tables (base tables) • Indexes other than text indexes
19	TEXT_DELIMITER_TYPE	CHAR(1)	Type of text-index delimiter specification • 'D'

No.	Column name	Data type	Stored information
			Text index for which DEFAULT is specified as the text-index delimiter specification (including those whose text-index delimiter specification was omitted causing DEFAULT to be assumed) • 'A' Text index for which ALL is specified as the text-index delimiter specification A null value is stored for the following indexes: • Text index for which no text-index-word-context search specification is specified • Indexes of dictionary tables (base tables) • Indexes of system tables (base tables) • Indexes other than text indexes

#

The following figure shows the format of the indexed column ID list.

Figure B-2: Format of the indexed column ID list



IDm = { When ascending order is specified, or for a range index or text index, the m^{th} column ID is stored.
When descending order is specified, the m^{th} column ID × (-1) is stored. (Units: bytes)

ID1, ID2, ..., ID*n* are stored in the order in which they were specified when the indexes were defined.

B.6 Content of SQL_DIV_INDEX

SQL_DIV_INDEX stores information about locations such as DB areas that store indexes of base tables. It also stores information about locations such as DB areas that store B-tree indexes for dictionary tables (base tables) and system tables (base tables). Each row stores information for one index.

The following table describes the content of SQL_DIV_INDEX.

Table B-8: Content of SQL_DIV_INDEX

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier of the table for which the index is defined
3	INDEX_NAME	VARCHAR(100)	Index identifier
4	INDEX_ID	SMALLINT	Index ID: • From 0x00030001: Indexes of dictionary tables (base tables) • From 0x00030191: Indexes of system tables (base tables)

No.	Column name	Data type	Stored information
			• From 0x00030321: Indexes of base tables defined by HADB users
5	DBAREA_ID	SMALLINT	DB area ID of the DB area that stores the indexes

B.7 Content of SQL_DBAREAS

SQL_DBAREAS stores DB area definition information. Each row stores information for one DB area.

The following table describes the content of SQL_DBAREAS.

Table B-9: Content of SQL_DBAREAS

No.	Column name	Data type	Stored information
1	DBAREA_NAME	VARCHAR(100)	DB area name
2	DBAREA_ID	SMALLINT	DB area ID
3	DBAREA_TYPE	CHAR(4)	DB area type: • 'MAST': Master directory DB area • 'DICT': Dictionary DB area • 'DATA': Data DB area • 'WORK': Work table DB area • 'STBL': System-table DB area
4	PAGE_SIZE	SMALLINT	DB area page size (bytes) For the work table DB area, the page size specified for the adb_init_wrk_page_size operand, which is an initialization option in the adbinit command, is stored. Note that the page size that was specified in the adbinit command is stored even if the page size has been changed by using the adb_dbarea_wrk_page_size operand in the server definition.
5	SEGMENT_SIZE	SMALLINT	DB area segment size (bytes)
6	N_TABLE	SMALLINT	Number of base tables stored in DB areas For the master directory DB area and work table DB area, 0 is stored.
7	N_INDEX	SMALLINT	Number of indexes stored in the DB area. (sum of all indexes) For the master directory DB area and work table DB area, 0 is stored.

B.8 Content of SQL_SCHEMATA

SQL_SCHEMATA stores schema-related definition information. Each row stores information for one schema.

The following table describes the content of SQL_SCHEMATA.

Table B-10: Content of SQL_SCHEMATA

No.	Column name	Data type	Stored information
1	SCHEMA_OWNER	VARCHAR(100)	Schema owner
2	SCHEMA_DEFINER	VARCHAR(100)	Authorization identifier of the HADB user who defined the schema
3	SCHEMA_NAME	VARCHAR(100)	Schema name
4	CREATE_TIME	TIMESTAMP	Datetime stamp for when the schema was defined

B.9 Content of SQL_VIEWS

SQL_VIEWS stores viewed table definition information. Each row stores information for one viewed table.

The following table describes the content of SQL_VIEWS.

Table B-11: Content of SQL_VIEWS

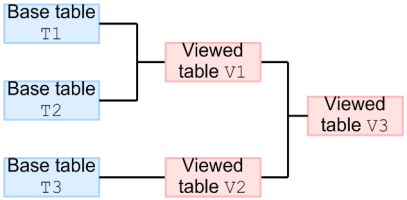
No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier
3	VIEW_ID	SMALLINT	View ID
4	IS_UPDATABLE	CHAR(1)	Updatability of viewed table 'Y': Updatable viewed table 'N': Read-only viewed table
5	VIEW_OBJECT_SIZE	SMALLINT	View object size
6	DEFINE_SOURCE_ID	INTEGER	SQL statement ID
7	IS_INVALID	CHAR(1)	Information on whether a viewed table has been invalidated • Y Viewed table is invalidated The null value is stored if the viewed table is not invalidated.
8	VIEW_LEVEL	SMALLINT	View level

B.10 Content of SQL_VIEW_TABLE_USAGE

SQL_VIEW_TABLE_USAGE stores information about the underlying tables of a viewed table. Each row stores information for one underlying table.

If the defined underlying table of the viewed table is a viewed table, that viewed table's underlying tables' information is also stored. All information is stored up until the viewed table's underlying tables become base tables. The following figure shows an example.

Figure B-3: Example of information stored when an underlying table is a viewed table



Explanation

The underlying tables of the viewed table V3 are viewed table V1 and viewed table V2. Therefore, information on the viewed table V3's underlying tables stored in SQL_VIEW_TABLE_USAGE consists of information not only for viewed tables V1 and V2, but also for base tables T1, T2, and T3.

The following table describes the content of SQL_VIEW_TABLE_USAGE.

Table B-12: Content of SQL_VIEW_TABLE_USAGE

No.	Column name	Data type	Stored information
1	VIEW_SCHEMA	VARCHAR(100)	Schema name of the viewed table
2	VIEW_NAME	VARCHAR(100)	Table identifier of the viewed table
3	VIEW_ID	SMALLINT	View ID
4	TABLE_SCHEMA	VARCHAR(100)	Schema name of the underlying table
5	TABLE_NAME	VARCHAR(100)	Table identifier of the underlying table
6	TABLE_TYPE	CHAR(1)	Type of the underlying table • 'R': Base table • 'V': Viewed table
7	IS_DIRECT	CHAR(1)	Information on whether the table is defined as a directly underlying table • 'Y': Directly underlying table • 'N': Not a directly underlying table

B.11 Content of SQL_DEFINE_SOURCE

SQL_DEFINE_SOURCE stores SQL statement information. Each row stores information for one SQL text.

The following table describes the content of SQL_DEFINE_SOURCE.

Table B-13: Content of SQL_DEFINE_SOURCE

No.	Column name	Data type	Stored information
1	SOURCE_ID	INTEGER	SQL statement ID
2	SOURCE_TYPE	CHAR(1)	SQL statement type: • 'V': View definition
3	DEFINE_SOURCE	VARCHAR(64000)	SQL statement SQL statement is stored in the format in which it was input.
4	SOURCE_SIZE	SMALLINT	Actual length of SQL statement

Note

When the dictionary table SQL_DEFINE_SOURCE is searched, only the information on the viewed table defined by the HADB user is output as the search result.

Even if you perform a search by specifying HADB or MASTER for the schema name, the search result will be 0 (no error will occur).

B.12 Content of SQL_DEFINE_ENVIRONMENT

SQL_DEFINE_ENVIRONMENT stores the environment information that was in effect when making or changing a view definition. Each row stores the environment information of one viewed table.

The following table describes the content of SQL_DEFINE_ENVIRONMENT.

Table B-14: Content of SQL_DEFINE_ENVIRONMENT

No.	Column name	Data type	Stored information
1	RESOURCE_TYPE	CHAR(1)	Definition information type: • 'V': Viewed table
2	RESOURCE_ID	SMALLINT	Definition information ID • View ID
3	DELETE_RESERVED_WORD	VARCHAR(32000)	Reserved words that had been unregistered when the CREATE VIEW statement was executed#. If no reserved words were unregistered, the null value is stored.
4	DEFINE_VR	CHAR(8)	Version that was in effect during definition Stored in the following format: '*VV-RR*' • *VV*: Version number • *RR*: Revision number
5	RECREATE_VR	CHAR(8)	The version of the HADB server when the ALTER VIEW statement was executed Version information for the HADB server that was in effect when an ALTER VIEW statement was executed most recently for the viewed table is stored in the following format: '*VV-RR*' • *VV*: Version number • *RR*: Revision number The null value is stored if no ALTER VIEW statements have been executed for the viewed table.

#

The DELETE_RESERVED_WORD column stores the reserved words to be unregistered as defined in the reserved word unregistration function environment in the order they were specified in the server definition, delimited by the comma (,) and without spaces.

The following is a storage example in which the reserved words ACTION and CATALOG are specified in the reserved word unregistration function environment to be unregistered and in which two viewed tables are defined.

Example: Search of SQL_DEFINE_ENVIRONMENT

```
SELECT * FROM "MASTER"."SQL_DEFINE_ENVIRONMENT"
```

Search result

RESOURCE_TYPE	RESOURCE_ID	DELETE_RESERVED_WORD	DEFINE_VR	RECREATE_VR
'V'	262385	ACTION,CATALOG	01-01	NULL
'V'	262386	ACTION,CATALOG	01-01	NULL

B.13 Content of SQL_USERS

SQL_USERS stores HADB user information. Each row stores information on one HADB user.

The following table describes the content of SQL_USERS.

Table B-15: Content of SQL_USERS

No.	Column name	Data type	Stored information
1	USER_NAME	VARCHAR(100)	HADB user's authorization identifier
2	USER_ID	SMALLINT	HADB user's user ID
3	DBA_PRIVILEGE	CHAR(1)	Whether the user has the DBA privilege 'Y': Has the DBA privilege. 'N': Does not have the DBA privilege.
4	CONNECT_PRIVILEGE	CHAR(1)	Whether the user has the CONNECT privilege 'Y': Has the CONNECT privilege. 'N': Does not have the CONNECT privilege.
5	SCHEMA_PRIVILEGE	CHAR(1)	Whether the user has the schema definition privilege 'Y': Has the schema definition privilege. 'N': Does not have the schema definition privilege.
6	CREATE_TIME	TIMESTAMP	HADB user creation date and time
7	AUDIT_ADMIN_PRIVILEGE	CHAR(1)	Whether the user has the audit admin privilege 'Y': Has the audit admin privilege 'N': Does not have the audit admin privilege
8	AUDIT_VIEWER_PRIVILEGE	CHAR(1)	Whether the user has the audit viewer privilege 'Y': Has the audit viewer privilege 'N': Does not have the audit viewer privilege

B.14 Content of SQL_TABLE_CONSTRAINTS

SQL_TABLE_CONSTRAINTS stores constraint information related to base tables. Each row stores information on one constraint.

The following table describes the content of SQL_TABLE_CONSTRAINTS.

Table B-16: Content of SQL_TABLE_CONSTRAINTS

No.	Column name	Data type	Stored information
1	CONSTRAINT_SCHEMA	VARCHAR(100)	Schema name that includes a constraint
2	CONSTRAINT_NAME	VARCHAR(100)	Constraint name
3	CONSTRAINT_TYPE	CHAR(1)	Constraint type • 'P': Primary key • 'F': Foreign key
4	TABLE_SCHEMA	VARCHAR(100)	Schema name of a table for which a constraint is defined
5	TABLE_NAME	VARCHAR(100)	Table identifier of a table for which a constraint is defined

B.15 Content of SQL_INDEX_COLINF

SQL_INDEX_COLINF stores information related to indexed columns. Each row stores information on one indexed column.

The following table describes the content of SQL_INDEX_COLINF.

Table B-17: Content of SQL_INDEX_COLINF

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier of a table for which an index is defined
3	INDEX_NAME	VARCHAR(100)	Index identifier
4	INDEX_ID	SMALLINT	Index ID
5	INDEX_ORDER	SMALLINT	Order of the columns that comprise an index Stores the order of columns that comprise an index in ascending order, starting with 1.
6	COLUMN_NAME	VARCHAR(100)	Indexed column name
7	ASC_DESC	CHAR(1)	Order of B-tree index key values • 'A' ASC is specified (the order is ascending). • 'D' DESC is specified (the order is descending). The null value is stored for indexes other than B-tree indexes.

B.16 Content of SQL_KEY_COLUMN_USAGE

SQL_KEY_COLUMN_USAGE stores information related to columns that comprise primary keys and foreign keys. Each row stores information on one column comprising a primary key and foreign keys.

The following table describes the content of SQL_KEY_COLUMN_USAGE.

Table B-18: Content of SQL_KEY_COLUMN_USAGE

No.	Column name	Data type	Stored information
1	CONSTRAINT_SCHEMA	VARCHAR(100)	Schema name including a constraint
2	CONSTRAINT_NAME	VARCHAR(100)	Constraint name
3	TABLE_SCHEMA	VARCHAR(100)	Schema name of the table for which the constraint is defined
4	TABLE_NAME	VARCHAR(100)	Table identifier of the table for which the constraint is defined
5	COLUMN_NAME	VARCHAR(100)	Column name of the column for which the constraint is defined
6	COLUMN_ORDER	SMALLINT	Definition order of column for which the constraint is defined

B.17 Content of SQL_REFERENTIAL_CONSTRAINTS

SQL_REFERENTIAL_CONSTRAINTS stores information related to referential constraints. Each row stores information on one referential constraint.

The following table describes the content of SQL_REFERENTIAL_CONSTRAINTS.

Table B-19: Content of SQL_REFERENTIAL_CONSTRAINTS

No.	Column name	Data type	Stored information
1	CONSTRAINT_SCHEMA	VARCHAR(100)	Schema name, including a referential constraint
2	CONSTRAINT_NAME	VARCHAR(100)	Constraint name of referential constraint
3	TABLE_SCHEMA	VARCHAR(100)	Schema name of the table for which the referential constraint is defined
4	TABLE_NAME	VARCHAR(100)	Table identifier of the table for which the referential constraint is defined
5	PRIMARY_CONSTRAINT_SCHEMA	VARCHAR(100)	Schema name, including a constraint referenced by a foreign key
6	PRIMARY_CONSTRAINT_NAME	VARCHAR(100)	Constraint name referenced by a foreign key
7	REFERENCED_TABLE_SCHEMA	VARCHAR(100)	Schema name of a referenced table
8	REFERENCED_TABLE_NAME	VARCHAR(100)	Table identifier of a referenced table
9	IS_DISABLE	CHAR(1)	Whether referential constraint check suppression is specified • 'Y' Referential constraints are not checked.

B.18 Content of SQL_TABLE_PRIVILEGES

SQL_TABLE_PRIVILEGES stores information related to access privileges for tables. Each row stores one table's worth of information on one authorization identifier granted by a privilege grantor.

The following table describes the content of SQL_TABLE_PRIVILEGES.

Table B-20: Content of SQL_TABLE_PRIVILEGES

No.	Column name	Data type	Stored information
1	GRANTOR	VARCHAR(100)	Authorization identifier of the HADB user who granted the access privilege
2	GRANTEE	VARCHAR(100)	Authorization identifier of the HADB user to whom the access privilege was granted, or 'PUBLIC'
3	GRANTEE_TYPE	CHAR(1)	Type of HADB user to whom the access privilege was granted • 'I' Authorization identifier of the HADB user or PUBLIC
4	OWNER	VARCHAR(100)	Authorization identifier of the HADB user who owns the table to which the access privilege applies
5	TABLE_SCHEMA	VARCHAR(100)	Schema name of the table to which the access privilege applies
6	TABLE_NAME	VARCHAR(100)	Table identifier of the table to which the access privilege applies
7	GRANT_TIME	TIMESTAMP	Time when the access privilege was granted Stores the time when a privilege grantor granted a table access privilege to a single authorization identifier for the first time.
8	SELECT_PRIVILEGE	CHAR(1)	Whether the HADB user has the SELECT privilege • G The HADB user has the SELECT privilege with grant option • Y The HADB user has the SELECT privilege without grant option • N The HADB user does not have the SELECT privilege
9	INSERT_PRIVILEGE	CHAR(1)	Whether the HADB user has the INSERT privilege • G The HADB user has the INSERT privilege with grant option • Y The HADB user has the INSERT privilege without grant option • N The HADB user does not have the INSERT privilege
10	UPDATE_PRIVILEGE	CHAR(1)	Whether the HADB user has the UPDATE privilege • G The HADB user has the UPDATE privilege with grant option • Y The HADB user has the UPDATE privilege without grant option • N

No.	Column name	Data type	Stored information
			The HADB user does not have the UPDATE privilege
11	DELETE_PRIVILEGE	CHAR(1)	Whether the HADB user has the DELETE privilege • G The HADB user has the DELETE privilege with grant option • Y The HADB user has the DELETE privilege without grant option • N The HADB user does not have the DELETE privilege
12	TRUNCATE_PRIVILEGE	CHAR(1)	Whether the HADB user has the TRUNCATE privilege • G The HADB user has the TRUNCATE privilege with grant option • Y The HADB user has the TRUNCATE privilege without grant option • N The HADB user does not have the TRUNCATE privilege
13	REFERENCES_PRIVILEGE	CHAR(1)	Whether the HADB user has the REFERENCES privilege • G The HADB user has the REFERENCES privilege with grant option • Y The HADB user has the REFERENCES privilege without grant option • N The HADB user does not have the REFERENCES privilege
14	IMPORT_TABLE_PRIVILEGE	CHAR(1)	Whether the HADB user has the IMPORT TABLE privilege • G The HADB user has the IMPORT TABLE privilege with grant option • Y The HADB user has the IMPORT TABLE privilege without grant option • N The HADB user does not have the IMPORT TABLE privilege
15	REBUILD_INDEX_PRIVILEGE	CHAR(1)	Whether the HADB user has the REBUILD INDEX privilege • G The HADB user has the REBUILD INDEX privilege with grant option • Y

No.	Column name	Data type	Stored information
			The HADB user has the REBUILD INDEX privilege without grant option • N The HADB user does not have the REBUILD INDEX privilege
16	GET_COSTINFO_PRIVILEGE	CHAR(1)	Whether the HADB user has the GET COSTINFO privilege • G The HADB user has the GET COSTINFO privilege with grant option • Y The HADB user has the GET COSTINFO privilege without grant option • N The HADB user does not have the GET COSTINFO privilege
17	EXPORT_TABLE_PRIVILEGE	CHAR(1)	Whether the HADB user has the EXPORT TABLE privilege • G The HADB user has the EXPORT TABLE privilege with grant option • Y The HADB user has the EXPORT TABLE privilege without grant option • N The HADB user does not have the EXPORT TABLE privilege
18	MERGE_CHUNK_PRIVILEGE	CHAR(1)	Whether the HADB user has the MERGE CHUNK privilege • G The HADB user has the MERGE CHUNK privilege with grant option • Y The HADB user has the MERGE CHUNK privilege without grant option • N The HADB user does not have the MERGE CHUNK privilege
19	CHANGE_CHUNK_COMMENT_PRIVILEGE	CHAR(1)	Whether the HADB user has the CHANGE CHUNK COMMENT privilege • G The HADB user has the CHANGE CHUNK COMMENT privilege with grant option • Y The HADB user has the CHANGE CHUNK COMMENT privilege without grant option • N The HADB user does not have the CHANGE CHUNK COMMENT privilege
20	CHANGE_CHUNK_STATUS_PRIVILEGE	CHAR(1)	Whether the HADB user has the CHANGE CHUNK STATUS privilege

No.	Column name	Data type	Stored information
			• G The HADB user has the CHANGE CHUNK STATUS privilege with grant option • Y The HADB user has the CHANGE CHUNK STATUS privilege without grant option • N The HADB user does not have the CHANGE CHUNK STATUS privilege
21	ARCHIVE_CHUNK_PRIVILEGE	CHAR(1)	Whether the HADB user has the ARCHIVE CHUNK privilege • G The HADB user has the ARCHIVE CHUNK privilege with grant option • Y The HADB user has the ARCHIVE CHUNK privilege without grant option • N The HADB user does not have the ARCHIVE CHUNK privilege
22	UNARCHIVE_CHUNK_PRIVILEGE	CHAR(1)	Whether the HADB user has the UNARCHIVE CHUNK privilege • G The HADB user has the UNARCHIVE CHUNK privilege with grant option • Y The HADB user has the UNARCHIVE CHUNK privilege without grant option • N The HADB user does not have the UNARCHIVE CHUNK privilege

Note

You cannot retrieve information about the access privileges for a viewed table if the viewed table has been invalidated.

▪ Information on access privileges for dictionary tables and system tables

Information on access privileges for dictionary tables and system tables is not stored in SQL_TABLE_PRIVILEGES. Information on access privileges for dictionary tables and system tables, is assumed to be as follows:

- SELECT_PRIVILEGE (Whether the HADB user has the SELECT privilege)
  'G': The HADB user has the SELECT privilege with grant option
- EXPORT_TABLE_PRIVILEGE (Whether the HADB user has the EXPORT TABLE privilege)
  'G': The HADB user has the EXPORT TABLE privilege with grant option
- Whether the HADB user has any access privileges other than the preceding ones
  N: No

B.19 Content of SQL_AUDITS

SQL_AUDITS stores audit target definition information defined by the CREATE AUDIT statement. Each row stores information for one audit target definition.

The following table describes the content of SQL_AUDITS:

Table B-21: Content of SQL_AUDITS

No.	Column name	Data type	Stored information
1	OPERATION_TYPE	VARCHAR(30)	Target operation type 'ANY': All audit target operations
2	EVENT_TYPE	VARCHAR(30)	Used by the system
3	OBJECT_TYPE	VARCHAR(30)	Used by the system
4	OBJECT_SCHEMA	VARCHAR(100)	Used by the system
5	OBJECT_NAME	VARCHAR(100)	Used by the system
6	USER_NAME	VARCHAR(100)	Used by the system
7	EVENT_RESULT	VARCHAR(30)	Used by the system
8	AUDIT_TYPE	VARCHAR(30)	Audit trail type 'EVENT': Final event result
9	IS_EXCLUSION_AUDIT	CHAR(1)	Used by the system

B.20 Base tables that are reserved when dictionary tables are referenced

When SQL statements are executed to reference dictionary tables, the dictionary tables (base tables) and system tables (base tables) are reserved (locked). For details about the base tables that are reserved when dictionary tables are referenced, see 2.10.5 Base tables for which a lock is obtained when dictionary tables and system tables are referenced.

B.21 B-tree indexes of dictionary tables (base tables)

The following table shows the B-tree indexes defined for dictionary tables (base tables).

Table B-22: List of B-tree indexes of dictionary tables (base tables)

No.	Dictionary table (base table)	Indexed column	B-tree index name
1	SQL_TABLES	TABLE_SCHEMA	SQLINDEXM01
2		TABLE_NAME	
3		TABLE_ID	SQLINDEXS01
4	SQL_COLUMNS	TABLE_SCHEMA	SQLINDEXM02
5		TABLE_NAME	
6		COLUMN_NAME	
7		TABLE_ID	SQLINDEXS02

No.	Dictionary table (base table)	Indexed column	B-tree index name
8	SQL_DIV_TABLE	TABLE_SCHEMA	SQLINDEXM05
9		TABLE_NAME	
10		DBAREA_ID	SQLINDEXS04
11	SQL_INDEXES	TABLE_SCHEMA	SQLINDEXM03
12		INDEX_NAME	
13		TABLE_SCHEMA	SQLINDEXM04
14		TABLE_NAME	
15		INDEX_ID	SQLINDEXS03
16	SQL_DIV_INDEX	TABLE_SCHEMA	SQLINDEXM06
17		TABLE_NAME	
18		TABLE_SCHEMA	SQLINDEXM07
19		INDEX_NAME	
20		DBAREA_ID	SQLINDEXS05
21	SQL_DBAREAS	DBAREA_NAME	SQLINDEXS06
22		DBAREA_ID	SQLINDEXS07
23		N_TABLE	SQLINDEXM08
24		DBAREA_ID	
25	SQL_SCHEMATA	SCHEMA_OWNER	SQLINDEXS08
26		SCHEMA_NAME	SQLINDEXS09
27	SQL_VIEWS	TABLE_SCHEMA	SQLINDEXM09
28		TABLE_NAME	
29		VIEW_ID	SQLINDEXS10
30	SQL_VIEW_TABLE_USAGE	VIEW_SCHEMA	SQLINDEXM10
31		VIEW_NAME	
32		TABLE_SCHEMA	SQLINDEXM11
33		TABLE_NAME	
34		VIEW_ID	SQLINDEXS11
35	SQL_VIEW_OBJECT	VIEW_ID	SQLINDEXM12
36		OBJECT_NO	
37	SQL_DEFINE_SOURCE	SOURCE_ID	SQLINDEXS12
38	SQL_DEFINE_ENVIRONMENT	RESOURCE_TYPE	SQLINDEXM13
39		RESOURCE_ID	
40	SQL_USERS	USER_NAME	SQLINDEXS13
41	SQL_TABLE_CONSTRAINTS	CONSTRAINT_SCHEMA	SQLINDEXM14
42		CONSTRAINT_NAME	

No.	Dictionary table (base table)	Indexed column	B-tree index name
43		TABLE_SCHEMA	SQLINDEXM15
44		TABLE_NAME	
45	SQL_INDEX_COLINF	TABLE_SCHEMA	SQLINDEXM16
46		INDEX_NAME	
47		TABLE_SCHEMA	SQLINDEXM17
48		TABLE_NAME	
49	SQL_KEY_COLUMN_USAGE	CONSTRAINT_SCHEMA	SQLINDEXM18
50		CONSTRAINT_NAME	
51		COLUMN_NAME	
52		TABLE_SCHEMA	SQLINDEXM19
53		TABLE_NAME	
54		COLUMN_NAME	
55	SQL_REFERENTIAL_CONSTRAINTS	CONSTRAINT_SCHEMA	SQLINDEXM20
56		CONSTRAINT_NAME	
57		TABLE_SCHEMA	SQLINDEXM21
58		TABLE_NAME	
59		PRIMARY_CONSTRAINT_SCHEMA	SQLINDEXM22
60		PRIMARY_CONSTRAINT_NAME	
61		REFERENCED_TABLE_SCHEMA	SQLINDEXM23
62		REFERENCED_TABLE_NAME	
63	SQL_TABLE_PRIVILEGES	TABLE_SCHEMA	SQLINDEXM24
64		TABLE_NAME	
65		GRANTEE	
66		GRANTOR	
67		GRANTOR	SQLINDEXS14
68		GRANTEE	SQLINDEXS15
69	SQL_AUDITS	OBJECT_TYPE	SQLINDEXM25
70		OBJECT_SCHEMA	
71		OBJECT_NAME	

B.22 Searching a dictionary table

You can use the SELECT statement to search a dictionary table. The authorization identifier for a dictionary table is MASTER.

Once you have searched a dictionary table, immediately execute the COMMIT statement to terminate the transaction. If you do not execute the COMMIT statement, the lock on the dictionary table will not be released. Note that when you terminate the adbsql command after having searched a dictionary table, the lock on the dictionary table is released and you do not need to execute a COMMIT statement.

The information that can be searched depends on whether the HADB user who is searching the dictionary table has the DBA privilege. For details, see (3) Scope of information in dictionary tables that can be referenced by HADB users in B.1 Dictionary table overview.

The follow subsections provide specification examples of using the SELECT statement to search dictionary tables.

(1) When identifying the table name from a table ID

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME"
  FROM "MASTER"."SQL_TABLES"
    WHERE "TABLE_ID"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter a table ID.

(2) Finding a list of row store tables

Use the following SELECT statement to search a dictionary table:

Specification example

```
SELECT "TABLE_SCHEMA", "TABLE_NAME"
  FROM "MASTER"."SQL_TABLES"
    WHERE "STORAGE_FORMAT" = 'ROW'
```

(3) Finding a list of column store tables

Use the following SELECT statement to search a dictionary table:

Specification example

```
SELECT "TABLE_SCHEMA", "TABLE_NAME"
  FROM "MASTER"."SQL_TABLES"
    WHERE "STORAGE_FORMAT" = 'COLUMN'
```

(4) When identifying the index name from an index ID

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME"
  FROM "MASTER"."SQL_INDEXES"
    WHERE "INDEX_ID"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter an index ID.

(5) When identifying the name of a table for which an index is defined from an index ID

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME"
  FROM "MASTER"."SQL_INDEXES"
    WHERE "INDEX_ID"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter an index ID.

(6) When identifying the name of a column for which an index is defined from an index ID

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME","COLUMN_NAME"
  FROM "MASTER"."SQL_INDEX_COLINF" WHERE "INDEX_ID" = ?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter an index ID.

(7) When identifying the index name of a range index

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME"
  FROM "MASTER"."SQL_INDEXES"
    WHERE "INDEX_TYPE"='R'
```

(8) When identifying all index names defined in a table from the table name

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME"
  FROM "MASTER"."SQL_INDEXES"
    WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request.

(9) When identifying viewed table names of all viewed tables that use tables from the table names

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "VIEW_SCHEMA","VIEW_NAME"
  FROM "MASTER"."SQL_VIEW_TABLE_USAGE"
    WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request.

(10) When identifying the table names of all defined tables from the schema names

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME","TABLE_TYPE"
  FROM "MASTER"."SQL_TABLES"
    WHERE "TABLE_SCHEMA"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter a schema name.

Note that the value of TABLE_TYPE indicates the type of the table. The value 'R' means a base table, and the value 'V' means a viewed table.

(11) When identifying the index names of all defined indexes from the schema names

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME","INDEX_TYPE"
  FROM "MASTER"."SQL_INDEXES"
    WHERE "TABLE_SCHEMA"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter a schema name.

Note that the value of INDEX_TYPE indicates the type of the index. The value 'B' means a B-tree index, the value 'T' means a text index, and the value 'R' means a range index.

(12) When identifying the name of the DB area that stores indexes from the index names

Use the following SELECT statement to search dictionary tables.

Specification example

```
SELECT "MASTER"."SQL_DBAREAS"."DBAREA_NAME"
  FROM "MASTER"."SQL_DBAREAS","MASTER"."SQL_DIV_INDEX"
    WHERE "MASTER"."SQL_DIV_INDEX"."TABLE_SCHEMA"=?
      AND "MASTER"."SQL_DIV_INDEX"."INDEX_NAME"=?
      AND "MASTER"."SQL_DBAREAS"."DBAREA_ID"="MASTER"."SQL_DIV_INDEX"."DBAREA_ID"
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and an index identifier for the second input request.

(13) When identifying the maximum number of chunks created in all multi-chunk tables

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_NAME","N_CHUNK_RESERVED"
  FROM "MASTER"."SQL_TABLES"
    WHERE "IS_CHUNK" = 'Y'
      ORDER BY "TABLE_NAME"
```

(14) When identifying the table name of a table stored in a DB area

Use the following SELECT statement to search dictionary tables.

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME" FROM "MASTER"."SQL_DIV_TABLE"
  WHERE "DBAREA_ID"=
   (SELECT "DBAREA_ID" FROM "MASTER"."SQL_DBAREAS" WHERE "DBAREA_NAME"=?)
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter a DB area name.

(15) When identifying the index name of an index stored in a DB area

Use the following SELECT statement to search dictionary tables.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME" FROM "MASTER"."SQL_DIV_INDEX"
  WHERE "DBAREA_ID"=
   (SELECT "DBAREA_ID" FROM "MASTER"."SQL_DBAREAS" WHERE "DBAREA_NAME"=?)
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter a DB area name.

(16) When determining whether a viewed table is updatable

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "IS_UPDATABLE" FROM "MASTER"."SQL_VIEWS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and enter a table identifier of the viewed table for the second input request.

If the search result is 'Y', the viewed table is updatable. If the search result is 'N', the viewed table is read-only and therefore cannot be updated.

(17) When checking the user privileges and schema operation privilege that an HADB user has

Use the following SELECT statement to search the dictionary table.

An HADB user who has the DBA privilege can check the information on the user privileges and schema operation privileges that all HADB users have. An HADB user (with the authorization identifier that was used for the current connection to the HADB server) who does not have the DBA privilege can only check the information on his or her own user privileges and schema operation privilege.

Specification example

```
SELECT "DBA_PRIVILEGE","CONNECT_PRIVILEGE","SCHEMA_PRIVILEGE"
  FROM "MASTER"."SQL_USERS" WHERE "USER_NAME"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter an HADB user authorization identifier for the input request.

If the value of a column in the search result is 'Y', the HADB user has the relevant user privilege or schema operation privilege. If the value is 'N', the HADB user does not have the relevant user privilege or schema operation privilege.

(18) When determining the index name of the index that corresponds to the primary key

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME" FROM "MASTER"."SQL_INDEXES"
  WHERE "TABLE_SCHEMA" = ? AND "TABLE_NAME" = ? AND "IS_PRIMARY_KEY" = 'Y'
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and enter a table identifier for the second input request.

(19) When checking whether the index is a range index that can skip chunks

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "IS_CHUNK_SKIP" FROM "MASTER"."SQL_INDEXES"
  WHERE "TABLE_SCHEMA" = ? AND "INDEX_NAME" = ?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and enter an index identifier for the second input request.

If the search result is 'Y', the index is a range index that can skip chunks. If the search result is the null value, the index is either a range index that cannot skip chunks or not a range index.

(20) When checking the names of the columns comprising a primary key and foreign keys

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "COLUMN_NAME"
  FROM "MASTER"."SQL_KEY_COLUMN_USAGE"
    WHERE "CONSTRAINT_SCHEMA" = ? AND "CONSTRAINT_NAME" = ?
      ORDER BY "COLUMN_ORDER" ASC
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. For the first input request, enter a schema name that includes a constraint; for the second input request, enter a constraint name.

(21) When checking foreign keys that reference a primary key

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "CONSTRAINT_SCHEMA","CONSTRAINT_NAME"
  FROM "MASTER"."SQL_REFERENTIAL_CONSTRAINTS"
    WHERE "PRIMARY_CONSTRAINT_SCHEMA" = ?
      AND "PRIMARY_CONSTRAINT_NAME" = ?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. For the first input request, enter a schema name that includes a constraint; for the second input request, enter a constraint name.

(22) When checking whether a viewed table has been invalidated

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "IS_INVALID" FROM "MASTER"."SQL_VIEWS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. For the first input request, enter the schema name of the viewed table; for the second input request, enter the table identifier of the viewed table.

If the search result is 'Y', the viewed table has been invalidated. If the search result is the null value, the viewed table has not been invalidated.

(23) When checking the access privileges for a table

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "GRANTOR",
       "SELECT_PRIVILEGE","INSERT_PRIVILEGE","UPDATE_PRIVILEGE",
       "DELETE_PRIVILEGE","TRUNCATE_PRIVILEGE",
       "REFERENCES_PRIVILEGE","IMPORT_TABLE_PRIVILEGE",
       "REBUILD_INDEX_PRIVILEGE","GET_COSTINFO_PRIVILEGE",
       "EXPORT_TABLE_PRIVILEGE","MERGE_CHUNK_PRIVILEGE",
       "CHANGE_CHUNK_COMMENT_PRIVILEGE","CHANGE_CHUNK_STATUS_PRIVILEGE",
       "ARCHIVE_CHUNK_PRIVILEGE","UNARCHIVE_CHUNK_PRIVILEGE"
  FROM "MASTER"."SQL_TABLE_PRIVILEGES"
    WHERE "GRANTEE" IN ('PUBLIC', ?)
      AND "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. For the first input request, enter the authorization identifier of the HADB user to whom the access privilege was granted; for the second input request, enter the schema name of the table. For the third input request, enter the table identifier of the table.

Under the column GRANTOR, the authorization identifier of the HADB user who granted the access privilege is output.

For other columns, the system outputs whether the HADB user has each access privilege. If the search result is 'G', the HADB user has access privileges with a grant option for the table. If the search result is 'Y', the HADB user has access privileges (without a grant option) for the table. If the search result is 'N', the HADB user does not have access privileges for the table.

(24) Finding out whether a base table is a multi-chunk table, archivable multi-chunk table, or column store table

By using the SELECT statement in the specification example to search dictionary tables, you can find out the following information:

- Whether a base table is a single-chunk table or multi-chunk table
- Whether a base table is a regular multi-chunk table or an archivable multi-chunk table
- Whether a base table is a row store table or a column store table

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME","IS_CHUNK","IS_ARCHIVABLE","STORAGE_FORMAT"
  FROM "MASTER"."SQL_TABLES"
    WHERE "TABLE_SCHEMA" = ? AND "TABLE_NAME" = ?
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier.

Example search results

```
TABLE_SCHEMA  TABLE_NAME  IS_CHUNK  IS_ARCHIVABLE  STORAGE_FORMAT
------------  ----------  --------  -------------  --------------
ADBUSER01     T1                 Y              *  COLUMN
```

In this example, the type of the base table ADBUSER01.T1 (hereafter Table T1) is displayed.

- The IS_CHUNK column shows whether Table T1 is a multi-chunk table. If 'Y' is displayed, Table T1 is a multi-chunk table. If 'N' is displayed, Table T1 is a single-chunk table.
- The IS_ARCHIVABLE column shows whether Table T1 is an archivable multi-chunk table. If 'Y' is displayed, Table T1 is an archivable multi-chunk table. If a null value is displayed, Table T1 is not an archivable multi-chunk table. In this example, * represents a null value.
- The STORAGE_FORMAT column shows whether Table T1 is a row store table or a column store table. If 'COLUMN' is displayed, Table T1 is a column store table. If 'ROW' is displayed, Table T1 is a row store table.

In this example, the output shows that Table T1 is a regular multi-chunk table and a column store table.

(25) Finding out the archive directory for an archivable multi-chunk table

Search the dictionary table by using the following SELECT statement.

Specification example

```
SELECT "ARCHIVE_DIRECTORY_PATH" FROM "MASTER"."SQL_TABLES"
  WHERE "TABLE_SCHEMA" = ? AND "TABLE_NAME" = ?
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier.

(26) Finding out the name of the archive range column in an archivable multi-chunk table

Search the dictionary table by using the following SELECT statement.

Specification example

```
SELECT "COLUMN_NAME" FROM "MASTER"."SQL_COLUMNS"
  WHERE "TABLE_SCHEMA" = ? AND "TABLE_NAME" = ? AND "IS_ARCHIVE_RANGE_COLUMN" = 'Y
'
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier.

(27) Finding out the range indexes defined for the archive range column in an archivable multi-chunk table

Search the dictionary table by using the following SELECT statement.

Specification example

```
SELECT "IX"."TABLE_SCHEMA","IX"."INDEX_NAME"
  FROM "MASTER"."SQL_INDEXES" "IX"
```

```
      ,"MASTER"."SQL_INDEX_COLINF" "IC"
      ,"MASTER"."SQL_COLUMNS" "CL"
  WHERE "IX"."TABLE_SCHEMA" = ?
    AND "IX"."TABLE_NAME" = ?
    AND "IX"."INDEX_TYPE" = 'R'
    AND "IX"."INDEX_ID" = "IC"."INDEX_ID"
    AND "IC"."TABLE_SCHEMA" = "CL"."TABLE_SCHEMA"
    AND "IC"."TABLE_NAME" = "CL"."TABLE_NAME"
    AND "IC"."COLUMN_NAME" = "CL"."COLUMN_NAME"
    AND "CL"."IS_ARCHIVE_RANGE_COLUMN" = 'Y'
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier.

(28) Finding out base table definition information

When redefining a base table, if you do not know the specifications for the CREATE TABLE statements, execute the SELECT statements in specification examples 1 and 2, and search the dictionary table.

In addition, when executing a SELECT statement by executing the adbsql command, specify the -s option. The search results of the SELECT statement can be output to standard output in CSV format.

> **Note**
>
> For information about the adbsql command, see *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

Specification example 1

```
SELECT "MT"."TABLE_SCHEMA"                          AS "schema-name"
      ,"MT"."TABLE_NAME"                            AS "table-identifier"
      ,"MT"."FREE_AREA"                             AS "unused-area-of-table-specific
ation"
      ,"MT"."FIX_TABLE"                             AS "FIX-specification"
      ,"MT"."IS_BRANCH_ALL"                         AS "BRANCH ALL-specification"
      ,"MT"."IS_CHUNK"                              AS "chunk-specification"
      ,"MT"."N_CHUNK_RESERVED"                      AS "chunk-maximum-value"
      ,"MT"."IS_ARCHIVABLE"                         AS "chunk-archive-specification"
      ,"MT"."ARCHIVE_DIRECTORY_PATH"                AS "archive-directory"
      ,"MT"."STORAGE_FORMAT"                        AS "table-data-storage-format"
      ,"MR"."DBAREA_NAME"                           AS "name-of-DB-area-storing-table
"
      ,"MC"."COLUMN_ID"                             AS "column-ID"
      ,"MC"."COLUMN_NAME"                           AS "column-name"
      ,CONVERT("MC"."DATA_TYPE_CODE", CHAR(9), '0XXXXXXX')
                                                    AS "data-type-code"
      ,CONVERT("MC"."DATA_LENGTH", CHAR(9), '0XXXXXXX')
                                                    AS "data-lenght"
      ,"MC"."IS_NULLABLE"                           AS "whether-null-value-is-permitt
ed"
      ,"MC"."BRANCH"                                AS "branch specification"
      ,"MC"."DEFAULT_VALUE"                         AS "default-value"
      ,"MC"."IS_DEFAULT_COLUMN"                     AS "DEFAULT-clause-specification"
      ,"MC"."PRIMARY_KEY_COLUMN_SEQUENCE_NUMBER" AS "primary-key-configuration-seq
uence"
      ,"MC"."IS_ARCHIVE_RANGE_COLUMN"               AS "archive-range-column-specific
ation"
      ,"MC"."COMPRESSION_TYPE"                      AS "column-data-compression-type"
```

```
FROM "MASTER"."SQL_TABLES"    "MT"
    ,"MASTER"."SQL_DIV_TABLE" "MD"
    ,"MASTER"."SQL_DBAREAS"   "MR"
    ,"MASTER"."SQL_COLUMNS"   "MC"
WHERE "MT"."TABLE_SCHEMA" = ?
  AND "MT"."TABLE_NAME" = ?
  AND "MT"."TABLE_TYPE" = 'R'
  AND "MT"."TABLE_ID" = "MD"."TABLE_ID"
  AND "MD"."DBAREA_ID" = "MR"."DBAREA_ID"
  AND "MT"."TABLE_SCHEMA" = "MC"."TABLE_SCHEMA"
  AND "MT"."TABLE_NAME" = "MC"."TABLE_NAME"
ORDER BY "MT"."TABLE_SCHEMA" ASC
        ,"MT"."TABLE_NAME"   ASC
        ,"MC"."COLUMN_ID"    ASC
```

If you performed the preceding `SELECT` statements by using the `adbsql` command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier.

Specification example 2

```
SELECT "MT"."TABLE_SCHEMA"            AS "schema-name"
      ,"MT"."TABLE_NAME"              AS "table-identifier"
      ,"MT"."CONSTRAINT_SCHEMA"       AS "schema-name-including-constraint"
      ,"MT"."CONSTRAINT_NAME"         AS "constraint-name"
      ,"MT"."CONSTRAINT_TYPE"         AS "constraint-type"
      ,"MC"."COLUMN_NAME"             AS "column-name"
      ,"MC"."COLUMN_ORDER"            AS "definition-order"
      ,"MR"."REFERENCED_TABLE_SCHEMA" AS "referenced-table-schema-name"
      ,"MR"."REFERENCED_TABLE_NAME"   AS "referenced-table-table-identifier"
      ,"MR"."IS_DISABLE"              AS "referential-constraint-check-suppression
-specification"
FROM "MASTER"."SQL_TABLE_CONSTRAINTS" "MT"
INNER JOIN "MASTER"."SQL_KEY_COLUMN_USAGE" "MC"
  ON "MT"."CONSTRAINT_SCHEMA" = "MC"."CONSTRAINT_SCHEMA"
 AND "MT"."CONSTRAINT_NAME" = "MC"."CONSTRAINT_NAME"
LEFT OUTER JOIN "MASTER"."SQL_REFERENTIAL_CONSTRAINTS" "MR"
  ON "MT"."CONSTRAINT_SCHEMA" = "MR"."CONSTRAINT_SCHEMA"
 AND "MT"."CONSTRAINT_NAME" = "MR"."CONSTRAINT_NAME"
WHERE "MT"."TABLE_SCHEMA" = ?
  AND "MT"."TABLE_NAME" = ?
ORDER BY "MT"."TABLE_SCHEMA"      ASC
        ,"MT"."TABLE_NAME"        ASC
        ,"MT"."CONSTRAINT_SCHEMA" ASC
        ,"MT"."CONSTRAINT_NAME"   ASC
        ,"MC"."COLUMN_ORDER"      ASC
```

If you performed the preceding `SELECT` statements by using the `adbsql` command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier.

(29) Finding out index definition information

When redefining indexes, if you do not know the specifications for the `CREATE INDEX` statements, search the dictionary table by using the following `SELECT` statement.

In addition, when executing a `SELECT` statement by executing the `adbsql` command, specify the `-s` option. The search results of the `SELECT` statement can be output to standard output in CSV format.

Note

For details about the `adbsql` command, see *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

Specification example

```
SELECT "MI"."TABLE_SCHEMA"            AS "schema-name"
      ,"MI"."TABLE_NAME"              AS "table-identifier"
      ,"MI"."INDEX_NAME"              AS "index-identifier"
      ,"MC"."INDEX_ORDER"             AS "indexed-column-order"
      ,"MC"."COLUMN_NAME"             AS "column-name"
      ,"MC"."ASC_DESC"                AS "ascending-or-descending"
      ,"MI"."UNIQUE_TYPE"             AS "UNIQUE"
      ,"MI"."FREE_AREA"               AS "index-unused-area-specification"
      ,"MI"."INDEX_TYPE"              AS "index-type-specification"
      ,"MI"."IS_EXCLUDE_NULL_VALUES"  AS "null-value-exclusion-specification"
      ,"MI"."IS_TEXT_CORRECTION_RULE" AS "notation-correction-search text-index-sp
ecification"
      ,"MI"."IS_TEXT_WORDCONTEXT"     AS "text-index-word-context-search-specifica
tion"
      ,"MI"."TEXT_DELIMITER_TYPE"     AS "text-index-delimiter-specification"
      ,"MR"."DBAREA_NAME"             AS "name-of-DB-area-storing-indexes"
FROM "MASTER"."SQL_INDEXES"      "MI"
    ,"MASTER"."SQL_DIV_INDEX"    "MD"
    ,"MASTER"."SQL_DBAREAS"      "MR"
    ,"MASTER"."SQL_INDEX_COLINF" "MC"
WHERE "MI"."TABLE_SCHEMA" = ?
  AND "MI"."TABLE_NAME" = ?
  AND "MI"."INDEX_ID" = "MD"."INDEX_ID"
  AND "MI"."INDEX_ID" = "MC"."INDEX_ID"
  AND "MD"."DBAREA_ID" = "MR"."DBAREA_ID"
ORDER BY "MI"."TABLE_SCHEMA" ASC
        ,"MI"."TABLE_NAME"   ASC
        ,"MI"."INDEX_NAME"   ASC
        ,"MC"."INDEX_ORDER"  ASC
```

If you performed the preceding `SELECT` statements by using the `adbsql` command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier.

(30) Investigating whether range indexes are defined in the column specified as the archive range column

When changing a regular multi-chunk table into an archivable multi-chunk table by using `ALTER TABLE` statements, you need to investigate whether range indexes are defined in the column specified as the archive range column.

When investigating whether range indexes are defined in the column specified as the archive range column, search the dictionary table by using the following `SELECT` statement.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME" FROM "MASTER"."SQL_INDEXES" "IX"
  WHERE "TABLE_SCHEMA" = ?
    AND "TABLE_NAME" = ?
    AND "INDEX_TYPE" = 'R'
    AND EXISTS(SELECT * FROM "MASTER"."SQL_INDEX_COLINF" "IC"
                 WHERE "IX"."TABLE_SCHEMA" = "IC"."TABLE_SCHEMA"
```

```
                    AND "IX"."INDEX_NAME" = "IC"."INDEX_NAME"
                    AND "IC"."COLUMN_NAME" = ?)
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters.

- The first time you are prompted enter information, enter the schema name of the regular multi-chunk table.
- The second time you are prompted enter information, enter the table identifier of the regular multi-chunk table.
- The third time you are prompted enter information, enter the name of the column to be specified as the archive range column.

(31) Finding out viewed table definition information

When redefining viewed tables, if you do not know the specifications for the CREATE VIEW statements, search the dictionary table by using the following SELECT statement.

In addition, when executing a SELECT statement by executing the adbsql command, specify the -s option. The search results of the SELECT statement can be output to standard output in CSV format.

> **Note**
>
> For details about the adbsql command, see *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

Specification example

```
SELECT "MV"."TABLE_SCHEMA"          AS "schema-name"
      ,"MV"."TABLE_NAME"            AS "table-identifier"
      ,"MS"."DEFINE_SOURCE"         AS "viewed-table-statement"
      ,"ME"."DELETE_RESERVED_WORD" AS "reserved-word-for-deletion"
      ,"ME"."DEFINE_VR"             AS "definition-version"
FROM "MASTER"."SQL_VIEWS" "MV"
    ,"MASTER"."SQL_DEFINE_SOURCE" "MS"
    ,"MASTER"."SQL_DEFINE_ENVIRONMENT" "ME"
WHERE "MV"."TABLE_SCHEMA" = ?
  AND "MV"."TABLE_NAME" = ?
  AND "MV"."DEFINE_SOURCE_ID" = "MS"."SOURCE_ID"
  AND "MV"."VIEW_ID" = "ME"."RESOURCE_ID"
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name of the viewed table. The second time, enter the table identifier of the viewed table.

Example search results

```
Schema name Table identifier View definition statement                  Term pending deletion Definition version
----------- ---------------- ---------------------------------------- --------------------- ------------------
ADBUSER01   V11              create view v11 as select * from t1                            03-03
```

The information displayed as the view definition statement is the information specified in the CREATE VIEW statements for the viewed table V11 (schema name ADBUSER01)

Note the following points when redefining a viewed table after searching the dictionary table by using the preceding SELECT statement:

- Among the search results of the SELECT statement, if ""reserved-word-for-deletion"" is not a null value, when redefining the viewed table, you might need to specify the same reserved word for deletion as the last time you defined the viewed table.
- If changing the column name by using ALTER TABLE statements cancels invalidation of the viewed table, you might need to revise the column names with regard to the definition SQL statement created based on the search results of the SELECT statement.

(32) Finding a list of dependent viewed tables, and the definition information of each of the viewed tables

To find a list of dependent viewed tables, and the definition information of each of the viewed tables, search the dictionary table by using the following SELECT statement.

In addition, when executing a SELECT statement by executing the adbsql command, specify the -s option. The search results of the SELECT statement can be output to standard output in CSV format.

> **Note**
>
> For details about the adbsql command, see *adbsql (Execute SQL Statements)* in the manual *HADB Command Reference*.

Specification example

```
SELECT "MV"."TABLE_SCHEMA"          AS "schema-name"
      ,"MV"."TABLE_NAME"            AS "table-identifier"
      ,"MV"."IS_INVALID"            AS "invalid-or-not-invalid"
      ,"MV"."VIEW_LEVEL"            AS "view-level"
      ,"MS"."DEFINE_SOURCE"         AS "view-definition-statement"
      ,"ME"."DELETE_RESERVED_WORD" AS "reserved-word-for-deletion"
      ,"ME"."DEFINE_VR"             AS "definition-version"
 FROM  "MASTER"."SQL_VIEW_TABLE_USAGE" "MVU"
     ,"MASTER"."SQL_VIEWS" "MV"
     ,"MASTER"."SQL_DEFINE_SOURCE" "MS"
     ,"MASTER"."SQL_DEFINE_ENVIRONMENT" "ME"
 WHERE "MVU"."TABLE_SCHEMA"=?
   AND "MVU"."TABLE_NAME" = ?
   AND "MV"."TABLE_SCHEMA"      = "MVU"."VIEW_SCHEMA"
   AND "MV"."TABLE_NAME"        = "MVU"."VIEW_NAME"
   AND "MV"."DEFINE_SOURCE_ID" = "MS"."SOURCE_ID"
   AND "ME"."RESOURCE_TYPE"     = 'V'
   AND "MV"."VIEW_ID"           = "ME"."RESOURCE_ID"
 ORDER BY "view-level" ASC
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the table schema name. The second time, enter the table's table identifier.

The list of dependent viewed tables, and the definition information of each of the viewed tables, is displayed.

Example search results

Schema name	Table identifier	Invalid or not invalid	View level	View definition statement	Term pending deletion	Definition version
ADBUSER01	V12		1	create view v12 as select * from t1		03-03
ADBUSER01	V11		1	create view v11 as select * from t1		03-03
ADBUSER01	V3		2	create view v3 as select * from v12,v21		03-03

The preceding viewed tables (V12, V11, and V3) are viewed tables that depend on the table specified in the dynamic parameters.

The information displayed in the viewed table statements is the information specified in the CREATE VIEW statements for each viewed table.

Note the following points when redefining a viewed table after searching the dictionary table by using the preceding SELECT statement:

- Among the search results of the SELECT statement, if ""reserved-word-for-deletion"" is not a null value, when redefining the viewed table, you might need to specify the same reserved word for deletion as the last time you defined the viewed table.
- If changing the column name by using ALTER TABLE statements cancels invalidation of the viewed table, you might need to revise the column names with regard to the definition SQL statement created based on the search results of the SELECT statement.

(33) Finding a list of dependent viewed tables

To find a list of dependent viewed tables, search the dictionary table by using the following SELECT statement.

Specification example

```
SELECT "MV"."TABLE_SCHEMA"          AS "schema-name"
      ,"MV"."TABLE_NAME"            AS "table-identifier"
      ,"MV"."IS_INVALID"            AS "invalid-or-not-invalid"
FROM  "MASTER"."SQL_VIEW_TABLE_USAGE" "MVU"
     ,"MASTER"."SQL_VIEWS" "MV"
WHERE "MVU"."TABLE_SCHEMA"=?
   AND "MVU"."TABLE_NAME" = ?
   AND "MV"."TABLE_SCHEMA"      = "MVU"."VIEW_SCHEMA"
   AND "MV"."TABLE_NAME"        = "MVU"."VIEW_NAME"
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the table schema name. The second time, enter the table's table identifier.

The list of viewed tables that are dependent on the specified table is displayed.

Example search results

```
Schema name  Table identifier  Invalid or not invalid
-----------  ----------------  ----------------------
ADBUSER01    V12
ADBUSER01    V3
ADBUSER01    V11
```

The preceding viewed tables (V12, V3, and V11) are viewed tables that depend on the table specified in the dynamic parameters.

(34) Checking index options specified for text indexes

To find out which index options are specified for a text index, use the following SELECT statement to retrieve data from the dictionary table:

Specification example

```
SELECT "TABLE_SCHEMA"
      ,"INDEX_NAME"
      ,"INDEX_TYPE"
      ,"IS_TEXT_CORRECTION_RULE"
      ,"IS_TEXT_WORDCONTEXT"
      ,"TEXT_DELIMITER_TYPE"
  FROM "MASTER"."SQL_INDEXES"
    WHERE "TABLE_SCHEMA" = ?
      AND "INDEX_NAME" = ?
```

If you execute the preceding SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the index identifier.

(35) Checking which base tables and viewed tables are underlying tables of a viewed table

To identify the base tables and viewed tables that are the underlying tables of a viewed table, use the following SELECT statement to search a dictionary table:

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME"
  FROM "MASTER"."SQL_VIEW_TABLE_USAGE"
    WHERE "VIEW_SCHEMA"=? AND "VIEW_NAME"=? AND "IS_DIRECT"='Y'
```

If you execute the preceding SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name of the viewed table. The second time, enter the table identifier of the viewed table.

(36) Checking the authorization identifiers and audit privileges of auditors

Use the following SELECT statement to search a dictionary table.

Specification example

```
SELECT "USER_NAME","AUDIT_ADMIN_PRIVILEGE","AUDIT_VIEWER_PRIVILEGE"
  FROM "MASTER"."SQL_USERS"
    WHERE "AUDIT_ADMIN_PRIVILEGE"='Y' OR "AUDIT_VIEWER_PRIVILEGE"='Y'
```

Example search results

```
USER_NAME      AUDIT_ADMIN_PRIVILEGE  AUDIT_VIEWER_PRIVILEGE
-------------  ---------------------  ----------------------
ADBAUDITADMIN                      Y                       N
ADBAUDITOR                         N                       Y
```

A list of auditors is displayed.

The USER_NAME column displays the authorization identifier of the auditor.

If 'Y' appears in the AUDIT_ADMIN_PRIVILEGE column, that auditor has the audit admin privilege.

If 'Y' appears in the AUDIT_VIEWER_PRIVILEGE column, that auditor has the audit viewer privilege.

(37) Checking the page size of the work table DB area specified when the adbinit command was executed

Use the following SELECT statement to search the dictionary table.

Specification example

```
SELECT "PAGE_SIZE"
  FROM "MASTER"."SQL_DBAREAS"
    WHERE "DBAREA_NAME"='ADBWRK'
```

The page size of the work table DB area that was specified when the adbinit command was executed is output in the PAGE_SIZE column in bytes.

(38) Checking the tables subject to the updated-row columnizing facility (checking the column store tables for which no B-tree indexes are defined)

Use the following SELECT statement to search the dictionary table.

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME"
  FROM "MASTER"."SQL_TABLES"
    WHERE "STORAGE_FORMAT" = 'COLUMN'
      AND "TABLE_ID"
           NOT IN (
                   SELECT "TABLE_ID"
                     FROM "MASTER"."SQL_INDEXES"
                       WHERE "INDEX_TYPE" = 'B'
                   )
```

A list of tables subject to the updated-row columnizing facility (column store tables for which no B-tree indexes are defined) is displayed.

C. System Tables

This appendix lists the information stored in system tables and the B-tree indexes defined for system tables, and it explains how to search for information in system tables.

C.1 System table overview

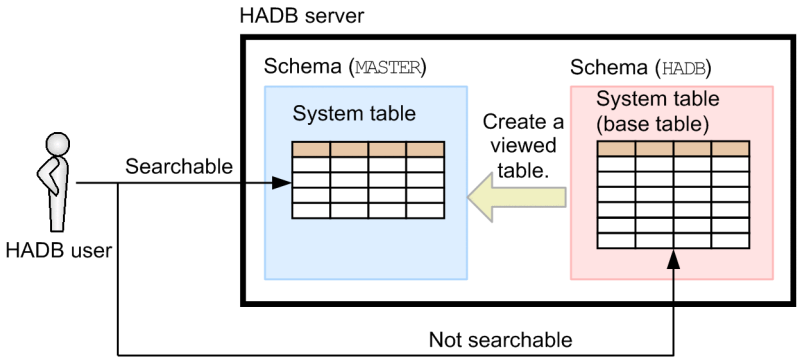
System tables store cost information for base tables and indexes, as well as chunk information for multi-chunk tables.

HADB provides two types of system tables, base tables and viewed tables.

- System table (base table)
  In this manual, a system table that is a base table is called a *system table (base table)*. The schema name of a system table (base table) is `HADB`.
- System table (viewed table)
  In this manual, a system table that is a viewed table is called a *system table*. The schema name of a system table is `MASTER`.

The following figure shows the relationship between a system table (base table) and a system table.

Figure C-1: Relationship between a system table (base table) and a system table



Explanation:

Cost information for base tables and indexes, and chunk information for multi-chunk tables, is stored in system tables (base tables). A viewed table that is created from a system table (base table) for a search purpose becomes a system table.

HADB users can search system tables. However, HADB users cannot search some system tables (base tables).

An HADB user can check information on items such as cost information collection date and time and chunk creation date and time by using the `SELECT` statement to search system tables. When searching system tables, specify `MASTER` as the schema name.

(1) List of system tables

The following table lists the system tables.

Table C-1: List of system tables

No.	System table name	Stored information	Table ID of the system table (base table) that corresponds to the system table	Number of subqueries included in the view definition#	Existence of external reference
1	STATUS_TABLES	Stores cost information related to base tables. Each row stores information for one base table.	0x000200C9	1	Y
2	STATUS_COLUMNS	Stores cost information related to columns. Each row stores information for one column.	0x000200CA	1	Y
3	STATUS_INDEXES	Stores cost information related to indexes. Each row stores information for one index. This table does not store cost information for range indexes.	0x000200CB	1	Y
4	STATUS_CHUNKS	Stores chunk information related to multi-chunk tables. Each row stores information for one chunk.	0x000200CC	1	Y
5	STATUS_SYNONYM_DICTIONARIES	Stores information related to synonym dictionaries. Each row stores information for one dictionary.	0x000200CD	0	N

Legend:

Y: The system table's viewed table includes a subquery that performs an external reference. Therefore, you cannot specify a multiset value expression subquery in the corresponding system table.

N: The system table's viewed table does not include a subquery that performs an external reference.

#

If you specified the corresponding system table in a SQL statement subquery, the number that must be added as the number of nests in that subquery is displayed.

(2) Times at which a system table is created

A system table is automatically created at the following times:

- When the HADB server is started for the first time following the completion of database initialization
- When the HADB server version is upgraded

(3) Scope of information in system tables that can be referenced by HADB users

The scope of information in system tables that can be referenced by HADB users varies according to the privileges each user has. The following table shows the system table information that an HADB user can reference.

Note that even for viewed tables specified as an underlying table of the system table, the information a user can reference varies according to the users' privileges. Therefore, even when searching the same viewed table, the HADB user who defined the viewed table and users granted access privileges for that viewed table might not be able to view the same information.

Table C-2: System table information that an HADB user can reference

System table name	Information being referenced	Privilege of HADB user who references system table			
		DBA privilege	Audit admin privilege	Access privilege	CONNECT privilege
STATUS_TABLES	Their own	Y	Y	Y	Y
	Other user	Y	N	C#	N
	MASTER	--	--	--	--
STATUS_COLUMNS	Their own	Y	Y	Y	Y
	Other user	Y	N	C#	N
	MASTER	--	--	--	--
STATUS_INDEXES	Their own	Y	Y	Y	Y
	Other user	Y	N	C#	N
	MASTER	--	--	--	--
STATUS_CHUNKS	Their own	Y	Y	Y	Y
	Other user	Y	N	C#	N
	MASTER	--	--	--	--
STATUS_SYNONYM_DICTIONARIES	Their own	Y	Y	Y	Y
	Other user	Y	N	C#	N
	MASTER	--	--	--	--

Legend:

Their own: Information related to a schema or schema object the HADB user owns.

Other user: Information related to a schema or schema object owned by another HADB user.

MASTER: Information related to dictionary tables and system tables.

Y: Can be referenced.

C: Can be referenced but conditions apply.

N: Cannot be referenced.

--: Not applicable.

#

Users can reference information related to schema objects for which they have the access privilege, and the schema of those schema objects.

C.2 Content of STATUS_TABLES

STATUS_TABLES stores cost information related to base tables. Each row stores information for one base table.

The following table describes the contents of STATUS_TABLES.

Table C-3: Content of STATUS_TABLES

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier
3	COLLECT_TIME	TIMESTAMP	Date and time cost information was collected
4	N_ROW	INTEGER	Number of rows in the processing-target table If the value in the IS_N_ROW_COMPLETE column is 'N', the number of rows for which analysis was completed at the time when collection of cost information was canceled is stored.
5	COLLECT_VR	CHAR(8)	Version of the HADB server that collected cost information#
6	IS_COMPLETE	CHAR(1)	Whether collection of the characteristics of the data stored in the processing-target table has been completed# • 'Y': Completed • 'N': Not completed Collection of cost information was canceled because the time required to collect cost information exceeded the maximum (time specified for the adb_getcst_collect_time cost information collection option).
7	IS_N_ROW_COMPLETE	CHAR(1)	Whether acquisition of the number of rows in the processing-target table has been completed# • 'Y': Completed • 'N': Not completed Collection of cost information was canceled because the time required to collect cost information exceeded the maximum (time specified for the adb_getcst_collect_time cost information collection option).

#

For a table that satisfies all of the following conditions, a null value is stored in this column of STATUS_TABLES:

- Table from which cost information was collected by an HADB server whose version is earlier than 04-03
- Table from which cost information was not collected by an HADB server whose version is 04-03 or later

C.3 Content of STATUS_COLUMNS

STATUS_COLUMNS stores cost information relating to the base table columns. One row stores information for one column.

The content of STATUS_COLUMNS is shown in the following table.

Table C-4: Content of STATUS_COLUMNS

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier

No.	Column name	Data type	Stored information
3	COLUMN_NAME	VARCHAR(100)	Column name
4	DATA_TYPE_CODE	SMALLINT	Column data type
5	DATA_LENGTH	SMALLINT	Defined length of column data For details, see *DATA_LENGTH* under Table B-4: Content of SQL_COLUMNS.
6	COLUMN_VALUE_MAX	VARCHAR(100)	Maximum column value If the value of the IS_MAXMIN_COMPLETE column is 'N', the maximum value in the processing-target column at the time when collection of cost information was canceled is stored. If no column data exists or the data existing in the column is only NULL, a null value is stored in this column.
7	COLUMN_VALUE_MIN	VARCHAR(100)	Minimum column value If the value of the IS_MAXMIN_COMPLETE column is 'N', the minimum value in the processing-target column at the time when collection of cost information was canceled is stored. If no column data exists or the data existing in the column is only NULL, a null value is stored in this column.
8	N_NULL	INTEGER	Number of null values The number of null values in the processing-target columns for which analysis was completed at the time when collection of cost information was canceled is stored in the following case: in STATUS_TABLES that corresponds to the table that includes this column, the value of the IS_COMPLETE column is 'N'.
9	IS_MAXMIN_COMPLETE	CHAR(1)	Whether collection of the maximum column value and minimum column value was completed# • 'Y': Completed • 'N': Not completed Collection of cost information was canceled because the time required to collect cost information exceeded the maximum (time specified for the adb_getcst_collect_time cost information collection option).

#

For a table that satisfies all of the following conditions, a null value is stored in this column of STATUS_COLUMNS:

- Table from which cost information was collected by an HADB server whose version is earlier than 04-03
- Table from which cost information was not collected by an HADB server whose version is 04-03 or later

C.4 Content of STATUS_INDEXES

STATUS_INDEXES stores cost information related to indexes. Each row stores information for one index.

The following table describes the contents of STATUS_INDEXES.

Table C-5: Content of STATUS_INDEXES

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier
3	INDEX_NAME	VARCHAR(100)	Index identifier
4	INDEX_TYPE	CHAR(1)	Index type: 'B': B-tree index 'T': Text index
5	COLLECT_TIME	TIMESTAMP	Date and time cost information was collected

C.5 Content of STATUS_CHUNKS

STATUS_CHUNKS stores chunk information related to multi-chunk tables. Each row stores information for one chunk.

The following table describes the contents of STATUS_CHUNKS.

Table C-6: Content of STATUS_CHUNKS

No.	Column name	Data type	Stored information
1	TABLE_SCHEMA	VARCHAR(100)	Schema name
2	TABLE_NAME	VARCHAR(100)	Table identifier
3	CHUNK_ID	INTEGER	Chunk ID
4	CREATE_TIME	TIMESTAMP	Date and time the chunk was created • If chunks have been merged, the earliest creation time associated with the merge-source chunks is stored. • For a chunk that was created in wait status by the adbimport command and was later changed to the normal status by the adbchgchunkstatus command, the date and time the adbchgchunkstatus command was executed are stored. • For a chunk that was created by using the adbchgchunkstatus command to swap the current chunk, the date and time the adbchgchunkstatus command was executed are stored. In the following case, the null value is stored: • When a chunk is created in wait status by the adbimport command
5	SWAP_TIME	TIMESTAMP	Date and time when the current chunk was swapped • If chunks have been merged, the most recent date and time information associated with the merge-source chunks when the current chunk was swapped is stored. • For a chunk that was created in wait status by the adbimport command and was later changed to the normal status by the adbchgchunkstatus command, and which

No.	Column name	Data type	Stored information
			was not swapped to become the current chunk when its status was changed to the normal status, the date and time the `adbchgchunkstatus` command was executed are stored. A null value is stored in the following cases: • A new chunk has not been created (the current chunk has not been swapped). • When chunks were merged, the current chunk was included in the merge-source chunks. • When a chunk is created in wait status by the `adbimport` command
6	CHUNK_COMMENT	VARCHAR(1024)	Comment that is set to the chunk The null value is stored if no comment is set.
7	CHUNK_STATUS	VARCHAR(32)	Chunk status • `'Normal'` Normal status • `'Pending Delete'` Delete-pending status • `'Wait'` Wait status
8	ARCHIVE_STATUS	VARCHAR(32)	Is archived chunk? • `'Archive'` Is archived chunk • Null value Is not archived chunk
9	STORAGE_FORMAT	VARCHAR(32)	Chunk-data storage format • `'COLUMN'` Column store format • `'ROW'` Row store format

C.6 Content of STATUS_SYNONYM_DICTIONARIES

STATUS_SYNONYM_DICTIONARIES stores information relating to synonym dictionaries. One row stores information for one dictionary.

The content of STATUS_SYNONYM_DICTIONARIES is shown in the following table.

Table C-7: Content of STATUS_SYNONYM_DICTIONARIES

No.	Column name	Data type	Stored information
1	SYNONYM_DICTIONARY_NAME	VARCHAR(120)	Synonym dictionary name
2	CREATE_TIME	TIMESTAMP	Creation date and time of synonym dictionary
3	BINARY_PATH	VARCHAR(510)	Path name of directory for storing synonym dictionary files
4	BINARY_FILE_NAME	VARCHAR(130)	Synonym dictionary file name

No.	Column name	Data type	Stored information
5	BINARY_FILE_NAME_IGNORECASE	VARCHAR(130)	Synonym dictionary file name corresponding to correction search (IGNORECASE specified) If CASESENSITIVE is specified in the correction search options, null values are stored. Null values are also stored when no correction options are specified.
6	BINARY_FILE_NAME_SORTCODE	VARCHAR(130)	Synonym dictionary file name corresponding to correction search (SORTCODE specified) If CASESENSITIVE is specified in the correction search options, null values are stored. Null values are also stored when no correction search options are specified.
7	CORRECTION_RULE	VARCHAR(32)	Values specified in correction search options • 'CASESENSITIVE' No specifications made for correction search • 'CORRECTIONRULE' Specifications made for correction search When no correction search options are specified, 'CASESENSITIVE' is stored.
8	N_SYNONYM_GROUP	SMALLINT	Number of synonym groups
9	SYNONYM_DICTIONARY_COMMENT	VARCHAR(1024)	Comment If a comment is not specified, a null value is stored.

C.7 Base tables that are reserved when system tables are referenced

When SQL statements are executed to reference system tables, the dictionary tables (base tables) and the system tables (base tables) are reserved (locked). For details about the base tables that are reserved when system tables are referenced, see 2.10.5 Base tables for which a lock is obtained when dictionary tables and system tables are referenced.

C.8 B-tree indexes of system tables (base tables)

The following table shows the B-tree indexes defined for system tables (base tables).

Table C-8: List of B-tree indexes of system tables (base tables)

No.	System table	Indexed column	B-tree index name
1	STATUS_TABLES	TABLE_SCHEMA	STATUSINDEXM01
2		TABLE_NAME	
3	STATUS_COLUMNS	TABLE_SCHEMA	STATUSINDEXM02
4		TABLE_NAME	
5		COLUMN_NAME	
6	STATUS_INDEXES	TABLE_SCHEMA	STATUSINDEXM03
7		INDEX_NAME	

No.	System table	Indexed column	B-tree index name
8		TABLE_SCHEMA	STATUSINDEXM04
9		TABLE_NAME	
10	STATUS_CHUNKS	TABLE_SCHEMA	STATUSINDEXM05
11		TABLE_NAME	
12		CHUNK_ID	
13	STATUS_SYNONYM_DICTIONARIES	SYNONYM_DICTIONARY_NAME	STATUSINDEXS01

C.9 Searching system tables

You use the SELECT statement to search system tables. Searching system tables enables you to check cost information and chunk information. The authorization identifier for the system tables is MASTER.

After you have searched a system table, immediately execute a COMMIT statement to terminate the transaction. If you do not execute the COMMIT statement, the lock on the system table will not be released. Note that when you terminate the adbsql command after having searched a system table, the lock on the system table is released, and therefore you do not need to execute the COMMIT statement in this case.

If a system table is searched from a transaction for which the REPEATABLE READ transaction isolation level is set, information for an old, un-updated system table might be retrieved. To prevent this, use one of the following search procedures:

- Search a system table from a transaction for which the READ COMMITTED transaction isolation level is set.
  Search a system table after specifying READ_COMMITTED in #SET TRAN_ISOLV of the adbsql subcommand.
- Execute a COMMIT statement immediately before the search.
  If you plan to search a system table from a transaction for which the REPEATABLE READ transaction isolation level is set, search the system table after executing a COMMIT statement.

Information that can be searched depends on whether the HADB user who is searching the system table has the DBA privilege. For details, see (3) Scope of information in system tables that can be referenced by HADB users in C.1 System table overview.

The following subsections provide examples of specifying the SELECT statement to search system tables.

(1) Determining the names of all base tables from which cost information was collected and the collection dates and times

Use the following SELECT statement to search the STATUS_TABLES system table.

Specification example

```
SELECT "TABLE_SCHEMA","TABLE_NAME","COLLECT_TIME"
  FROM "MASTER"."STATUS_TABLES"
```

(2) Determining the names of all indexes for which cost information was collected and the collection dates and times

Use the following SELECT statement to search the STATUS_INDEXES system table.

Specification example

```
SELECT "TABLE_SCHEMA","INDEX_NAME","COLLECT_TIME"
  FROM "MASTER"."STATUS_INDEXES"
```

(3) Checking the information about all chunks in a table based on a table name

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT "CHUNK_ID","CHUNK_COMMENT","CHUNK_STATUS","CREATE_TIME","SWAP_TIME"
  FROM "MASTER"."STATUS_CHUNKS" WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request.

(4) Identifying chunk information for a chunk ID

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT * FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=? AND "CHUNK_ID"=?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request, a table identifier for the second input request, and a chunk ID for the third input request.

(5) Identifying the chunks that were created during a specified period

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT * FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND "CREATE_TIME" BETWEEN ? AND ?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request. Enter the start date and time of the period for the third input request, and enter the end date and time of the period for the fourth input request.

The SELECT statement above targets the following chunks as search items:

- Chunks created during the specified period

- Chunks that were created as wait status chunks with the background import facility, but then were changed to the normal status by using the adbchgchunkstatus command during the specified period

However, the following chunks are not included in the search:

- Chunks created as wait status chunks with the background import facility, during the specified period

(6) Identifying the newest chunk ID based on the chunk creation date and time

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND "CREATE_TIME"=(SELECT MAX("CREATE_TIME")
        FROM "MASTER"."STATUS_CHUNKS"
          WHERE TC."TABLE_SCHEMA"="TABLE_SCHEMA"
            AND TC."TABLE_NAME"="TABLE_NAME")
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request.

For the above SELECT statement, the search targets are the chunks for which the following dates and times are the newest:

- Date and time of chunk creation
- Date and time of change to normal status chunk by using the adbchgchunkstatus command after being created as a wait status chunk with the background import facility

However, the following chunks are not included in the search:

- Wait status chunks that have never had the normal status

(7) Identifying the oldest chunk ID based on the chunk creation date and time

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS" TC
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND "CREATE_TIME"=(SELECT MIN("CREATE_TIME")
        FROM "MASTER"."STATUS_CHUNKS"
          WHERE TC."TABLE_SCHEMA"="TABLE_SCHEMA"
            AND TC."TABLE_NAME"="TABLE_NAME")
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request.

For the above SELECT statement, the search targets are the chunks for which the following dates and times are the oldest:

- Date and time of chunk creation
- Date and time of change to normal status chunk using the adbchgchunkstatus command after being created as a wait status chunk with the background import facility

However, the following chunks are not included in the search:

- Wait status chunks that have never had the normal status

(8) Identifying the chunk IDs of chunks that were created during the specified period, or the chunks that have been swapped out from current chunk

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND (("CREATE_TIME" BETWEEN ? AND ?)
      OR ("SWAP_TIME" BETWEEN ? AND ?))
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request. Enter the date and time when chunk creation started for the third input request, and enter the date and time when chunk creation ended for the fourth input request. Enter the date and time when swapping of the current chunk started for the fifth input request, and enter the date and time when swapping of the current chunk ended for the sixth input request.

The SELECT statement above targets the following chunks:

- Chunks created during the specified period
- Chunks swapped out from the current chunk during the specified period
- Chunks that were created as wait status chunks with the background import facility, but then were changed to the normal status by using the adbchgchunkstatus command during the specified period

However, the following chunks are not included in the search:

- Wait status chunks that have never had the normal status

(9) Identifying chunks whose data might have been stored on the specified date

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
     AND "CREATE_TIME" <= ?
       AND ("SWAP_TIME" IS NULL OR "SWAP_TIME" >= ?)
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request, enter a table identifier for the second input request, and enter a specific date and time for the third and fourth input requests.

The SELECT statement above targets the following chunks:

- Chunks created before the specified date and time
- Chunks created before the specified date and time that were swapped out from the current chunk after the specified date and time
- Chunks that were created as wait status chunks with the background import facility, but then were changed to the normal status by using the adbchgchunkstatus command on the specified date and time

However, the following chunks are not included in the search:

- Wait status chunks that have never had the normal status

(10) Determining the number of chunks created

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT COUNT(*) FROM MASTER.STATUS_CHUNKS
  WHERE "SCHEMA_NAME" = ? AND "TABLE_NAME" = ?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request.

(11) Searching for chunks that included a specified comment

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT "CHUNK_ID","CHUNK_COMMENT" FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"= ? AND "TABLE_NAME"= ? AND "CHUNK_COMMENT" LIKE ?
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request, enter a table identifier for the second input request, and enter the comment that was set in chunks for the third input request.

(12) Checking for delete-pending chunks

Use the following SELECT statement to search the STATUS_CHUNKS system table.

Specification example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND "CHUNK_STATUS"='Pending Delete'
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameters. Enter a schema name for the first input request and a table identifier for the second input request.

(13) Checking for chunks in normal status

Use the following SELECT statement to search the system table.

Specification example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND "CHUNK_STATUS"='Normal'
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. For the first input request, enter a schema name; for the second input request, enter a table identifier.

(14) Checking for chunks in wait status

Use the following SELECT statement to search the system table.

Specification example

```
SELECT "CHUNK_ID" FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND "CHUNK_STATUS"='Wait'
```

If you execute the above SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. For the first input request, enter a schema name; for the second input request, enter a table identifier.

(15) Finding out whether a chunk is archived

Search the system table by using the following SELECT statement.

Specification example

```
SELECT "ARCHIVE_STATUS" FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=? AND "CHUNK_ID"=?
```

If you performed the preceding SELECT statements by using the adbsql command, you will be prompted to enter the input data for the dynamic parameters. The first time you are prompted to enter data, enter the schema name. The second time, enter the table identifier, and the third time, enter the chunk ID.

If Archive is output as the execution result of the SELECT statement, that means the chunk is archived.

(16) Checking the information about the current chunk

Search the system table by using the following SELECT statement.

Specification example

```
SELECT * FROM "MASTER"."STATUS_CHUNKS"
  WHERE "TABLE_SCHEMA"=? AND "TABLE_NAME"=?
    AND "CREATE_TIME" IS NOT NULL AND "SWAP_TIME" IS NULL
```

If you execute the preceding SELECT statement in the adbsql command, the system asks you to enter data for the dynamic parameter. Enter a schema name for the first input request and a table identifier for the second input request.

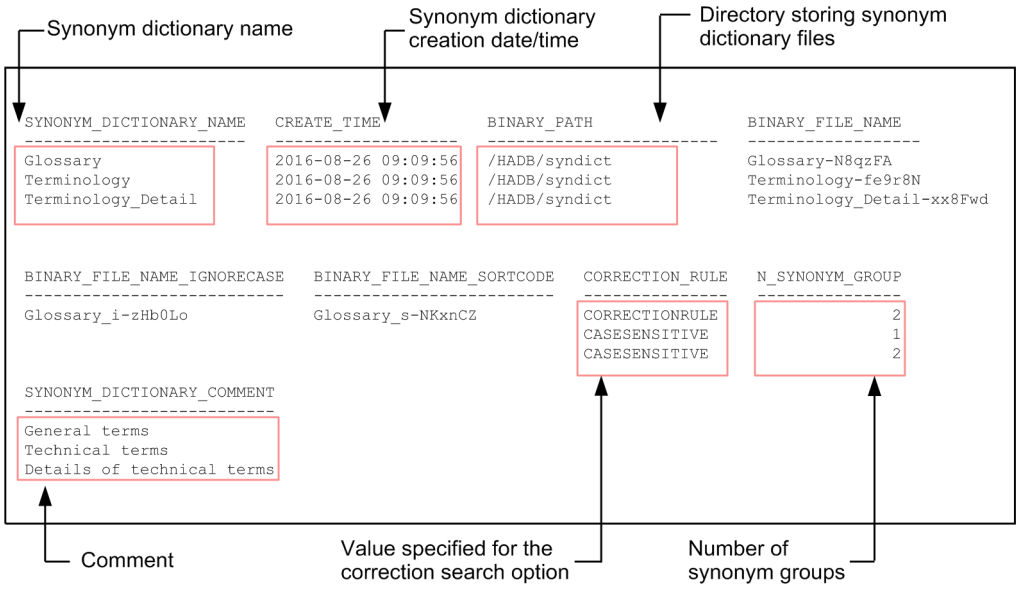
(17) Finding out information relating to all synonym dictionaries

Search the system table by using the following SELECT statement.

Specification example

```
SELECT * FROM "MASTER"."STATUS_SYNONYM_DICTIONARIES"
```

Example search results



For output items that do not have an explanation, see the explanation of the column information for `STATUS_SYNONYM_DICTIONARIES` in C.6 Content of STATUS_SYNONYM_DICTIONARIES.

Parts of the display format of the preceding search results might differ from the display format of the actual output results.

(18) Finding out information relating to specific synonym dictionaries

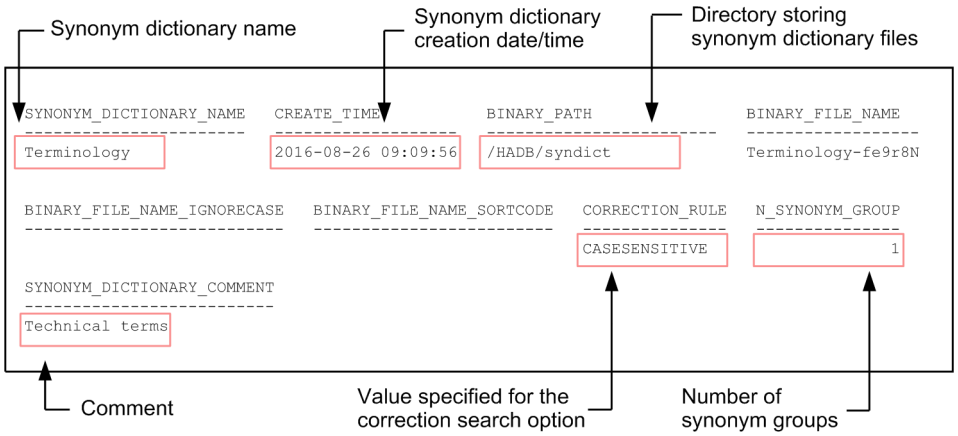
Search the system table by using the following `SELECT` statement.

Specification example

```
SELECT * FROM "MASTER"."STATUS_SYNONYM_DICTIONARIES"
  WHERE "SYNONYM_DICTIONARY_NAME"=?
```

If you performed the preceding `SELECT` statements by using the `adbsql` command, you will be prompted to enter the input data for the dynamic parameters. Enter the name of the synonym dictionary.

Example search results



For output items that do not have an explanation, see the explanation of the column information for `STATUS_SYNONYM_DICTIONARIES` in C.6 Content of STATUS_SYNONYM_DICTIONARIES.

Parts of the display format of the preceding search results might differ from the display format of the actual output results.

D. Maximum and Minimum Values in HADB

This appendix lists the maximum and minimum values relevant to the system configuration and databases.

D.1 Maximum and minimum values related to system configuration

The following table shows the maximum and minimum values relevant to the HADB system configuration.

Table D-1: Maximum and minimum values related to HADB system configuration

No.	Item	Minimum value	Maximum value	Unit
1	Total number of HADB users	1	30,000	Users
2	Total number of DB areas	5	1,024	Areas
3	Number of master directory DB areas		1	Areas
4	Number of dictionary DB areas		1	Areas
5	Number of system-table DB areas		1	Areas
6	Number of work table DB areas	1	4[#1]	Areas
7	Number of data DB areas	1	1,014	Areas
8	DB area name	1	30	Bytes
9	DB area file relative path name	1	100	Bytes
10	Size of dictionary DB area	--	16	Terabytes
11	Size of system-table DB area	--	16	Terabytes
12	Size of work table DB area	--	Work table DB area page size x 4,294,967,295	Bytes
13	Size of data DB area	--	128	Exabytes
14	Size of files in data DB area	--	128	Petabytes
15	Number of configuration files per DB area in data DB area	1	1,024	Files
16	Size of one work table	--	Work table DB area page size x 4,294,967,295	Bytes
17	Number of base tables that an HADB user can define	0	4,096	Tables
18	Number of viewed tables that an HADB user can define	0	30,000	Tables
19	Number of indexes that an HADB user can define	0	8,192	Indexes
20	Number of base tables in a DB area	0	200	Tables
21	Number of indexes per DB area	0	400	Indexes
22	Maximum number of statement handles		4,095	Handles
23	Maximum number of concurrent connections[#2]		1,024	Connections
24	Page size (excluding work table DB areas)	4,096	32,768	Bytes
25	Work table DB area page size	32,768	33,554,432	Bytes
26	Number of master log files		1	Files

No.	Item	Minimum value	Maximum value	Unit
27	Number of user log files	1	10,240	Files
28	Number of chunks in a DB area	1	30,000	Chunks
29	Number of chunks that can be created in a base table	1	30,000	Chunks
30	Number of nodes when the multi-node function is used	2	4	Nodes
31	Number of synonym dictionaries an HADB user can register	0	50	Synonym dictionaries

Legend:

--: Not applicable.

#1

This is the maximum value applied when using the multi-node function. Because one work table DB area can be created in each node, up to 4 work table DB areas can be created.

#2

When the multi-node function is used, this is the maximum number of concurrent connections for all HADB servers in the multi-node configuration.

D.2 Maximum and minimum values related to database

The following table shows the maximum and minimum values relevant to HADB databases.

Table D-2: Maximum and minimum values related to HADB database

No.	Item		Minimum value	Maximum value	Unit
1	Character string data length (definition length)	CHAR	1	32,000	Bytes
2		VARCHAR	1	64,000	Bytes
3	Numeric data length	INTEGER		8	Bytes
4		SMALLINT		4	Bytes
5		DOUBLE PRECISION		8	Bytes
6	Numeric data value range	INTEGER	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	--
7		SMALLINT	-2,147,483,648	2,147,483,647	
8	Precision of fixed-point data	DECIMAL	1	38	Digits
9	Scaling of fixed-point data	DECIMAL	0	38	Digits
10	Mantissa of floating-point numeric data	DOUBLE PRECISION		17	Digits
11	Exponent of floating-point numeric data	DOUBLE PRECISION		3	Digits

No.	Item		Minimum value	Maximum value	Unit
12	Date data value range	DATE	DATE'0001-01-01'	DATE'12/31/9999'	--
13	Time data value range	TIME	TIME'00:00:00.000000000000'	TIME'11:59:59 PM.999999999999'	--
14	Time stamp data value range	TIMESTAMP	TIMESTAMP'0001-01-01 00:00:00.000000000000'	TIMESTAMP'12/31/9999 11:59:59 PM.999999999999'	--
15	Time data fractional seconds precision	TIME	0	12	Digits
16	Fractional seconds precision of time stamp data	TIMESTAMP	0	12	Digits
17	Binary data length (defined length)	BINARY	1	32,000	Bytes
18		VARBINARY	1	32,000	Bytes
19	Range of values that can be specified for labeled duration (YEAR(S))		-9,998	9,998	--
20	Range of values that can be specified for labeled duration (MONTH(S))		-119,987	119,987	--
21	Range of values that can be specified for labeled duration (DAY(S))		-3,652,058	3,652,058	--
22	Range of values that can be specified for a labeled duration (HOUR(S))		-87,649,415	87,649,415	--
23	Range of values that can be specified for a labeled duration (MINUTE(S))		-5,258,964,959	5,258,964,959	--
24	Range of values that can be specified for a labeled duration (SECOND(S))		-315,537,897,599	315,537,897,599	--
25	Range of values that can be specified for a labeled duration (MILLISECOND(S))		-315,537,897,599,999	315,537,897,599,999	--
26	Range of values that can be specified for a labeled duration (MICROSECOND(S))		-315,537,897,599,999,999	315,537,897,599,999,999	--
27	Range of values that can be specified for a labeled duration (NANOSECOND(S))		-9,223,372,036,854,775,807	9,223,372,036,854,775,807	--
28	Range of values that can be specified for a labeled duration (PICOSECOND(S))		-9,223,372,036,854,775,807	9,223,372,036,854,775,807	--
29	Number of columns in a table		1	1,000	Columns
30	Number of indexes for a table		0	32	Indexes
31	Number of foreign keys in a table		0	255	Keys
32	Number of foreign keys that reference a primary key		0	255	Keys
33	Number of columns in a constraint		1	16	Columns
34	Number of B-tree indexed columns		1	16	Columns
35	Number of range indexed columns			1	Columns

No.	Item		Minimum value	Maximum value	Unit
36	Number of text indexed columns			1	Columns
37	Length of a CHARACTER type column for which a range index is defined		1	32	Bytes
38	Percentage of unused area in table pages		0	99	Percent
39	Percentage of unused area in index pages		0	99	Percent
40	Identifier length (authorization identifier, schema identifier, table identifier, index identifier, column name, correlation name, constraint name)		1	100	Bytes
41	SQL text length		1	16,000,000[#1]	Bytes
42	Number of tables in an SQL statement		1	2,048	Tables
43	Number of set operations	When all are UNION	0	1,023	Operations
44		When EXCEPT or INTERSECT is included	0	63	Operations
45	Number of times FULL OUTER JOIN is specified in an SQL statement		0	63	Times
46	Number of subquery and table value constructor specifications		1	1,024	Times
47	Number of sort keys		0	64	Keys
48	Number of grouping columns		0	64	Columns
49	Number of joined tables		1	64	Tables
50	Number of row value constructor specifications inside table value constructor		1	30,000	Tables
51	Number of row value constructor element specifications inside row value constructor		1	1,000	Subqueries
52	Number of selection expressions	Number of scalar subqueries	1	1	Subqueries
53		Number of table subqueries specified in the IN predicate	1	1	Subqueries
54		Number of table subqueries specified in the EXISTS predicate	1	1,000	Subqueries
55		Number of table subqueries in a derived table	1	1,000	Subqueries
56		Number of INSERT through SELECT statements	1	1,000	Statements
57		All other	1	1,000	Expressions

No.	Item	Minimum value	Maximum value	Unit
58	Number of inserted columns	1	1,000	Columns
59	Number of updated columns	1	1,000	Columns
60	Number of dynamic parameters	0	1,000	Parameters
61	Number of value expressions in an `IN` predicate	1	30,000	Expressions
62	Maximum number of rows	0	18,446,744,073,709,551,615	Rows
63	Number of scalar operations and set functions in a value expression	0	500	Operations or functions
64	Number of scalar operations and set functions in a value expression after it is expanded	0	10,000	Operations and functions
65	Number of nesting levels in a scalar function, window function, and `CASE` expression	0	15	Levels
66	Number of nesting levels in a subquery	0	32	Levels
67	Number of value expressions specified in the window partition clause of a window function	0	16	Expressions
68	Number of entities of identification numbers for the `RANDOMCURSOR` scalar functions in the SQL statement	0	1,000	Entities
69	Number of entities of identification numbers for the `RANDOMROW` scalar functions in the SQL statement	0	1,000	Entities
70	Maximum row length of work table	--	262,076[#2]	Bytes
71	Range of values specifiable as the maximum number of recursions of recursive queries	0	32,767	--
72	Number of `DISTINCT` set functions specified as set functions with different aggregated arguments in a query specification	0	64	Functions
73	Total data length of dynamic parameters in SQL statements	0	32,000,000	Bytes
74	Range of view levels of viewed tables	1	33	--

Legend:

--: Not applicable.

#1

For the `CREATE VIEW` statement, the maximum value is 64,000 bytes.

#2

If the page size of the work table DB area is 262,144 or fewer bytes, the maximum value is the page size of the work table DB area minus 68 bytes.

E. Process That Starts When the HADB Server Starts

When you execute the `adbstart` command, the HADB server's server process with the following name is started:

- `adbsrvd`

Only one server process is started, and it terminates when the `adbstop` command is executed.

A command process is started whenever an HADB command is executed. The name of each command process is the same as the name of the command.

Glossary

A

access privilege

A privilege required to access schema objects (base tables and viewed tables). Before you can use SQL statements to search the data in a schema object or use commands to manipulate a schema object, you must have access privileges for that schema object. There are several types of access privilege, including the `SELECT` privilege, `INSERT` privilege, and `IMPORT TABLE` privilege.

archivable multi-chunk table

A base table for which the background-import facility and chunk archiving function can be used.

archive directory

A directory that stores archive files.

archive file

A file that stores archived data.

archive range column

A column that is used to narrow down the search range when an archivable multi-chunk table is searched. An archive range column must be defined for each archivable multi-chunk table. A column that includes datetime data can be used as an archive range column.

archived state

The state in which archived data is stored in an archive file.

audit privilege

Audit privilege is a collective term for the following two privileges. You must have an audit privilege to use the audit trail facility.

- Audit admin privilege

  The privilege that an HADB user must have to perform operation of the audit trail facility.
- Audit viewer privilege

  The privilege that an HADB user must have to reference audit trails.

audit target definition

An audit target definition defines the combinations of events, HADB users, and schema objects for which to output audit trails. The information defined by a `CREATE AUDIT` statement is an audit target definition.

audit target event

An operation for which an audit trail is output. There are two types of audit target events, mandatory audit events and optional audit events. For mandatory audit events, an audit trail is always output if the audit trail facility is enabled. For optional audit events, auditors can choose whether to output audit trails.

audit trail facility

A facility that outputs information about activity by HADB users, such as database access and command execution, to a file as audit trail data. For example, when HADB users access a table, information about the operations they perform is output as an operation record (audit trail). This might include the time at which the users accessed the table, their authorization identifiers, the operations they performed, and the schema object on which they performed the operations. By viewing the output audit trail, an administrator can find out information such as who accessed what schema object at what time, and what operations they performed.

audit trail file

A file to which audit trail data is output that records information about activity by HADB users, such as database access and command execution.

auditor

An HADB user who has an audit privilege. An auditor is responsible for such tasks as operating the audit trail facility and using the output audit trails to audit database usage.

authorization identifier

A user ID that is allowed to connect to HADB.

B

background import

An optional facility of the `adbimport` command. Applying this facility when you execute the `adbimport` command enables you to concurrently execute a data search in a table and import data into the same table.

base row

If a branch row was created, the row that contains the information about the branching destination. If no branch row was created, one row of data is stored in a base row.

branch row

Variable-length column data (row) stored on a page other than the page on which the base row is stored.

C

centralized management of client definitions

A function that allows an administrator to centrally manage the client definition of each HADB client from the HADB server.

chunk

A unit of data that is stored in a base table by a single background-import operation. A chunk consists of base table data that was stored by background import and the index data that was created during background import. A new chunk is created each time background import is executed.

A chunk can be in any of the following statuses:

- Normal status
- Wait status

- Delete-pending status

Note that multiple chunks can be merged into a single chunk (chunk merging).

chunk archiving

To compress the data in a chunk by using the chunk archiving function and store the compressed data in an archive file.

chunk archiving function

A function that compresses the data in a chunk and stores the compressed data in an archive file.

chunk ID

An identifying number assigned to each chunk. Each chunk ID number is unique within the base table.

chunk in normal status

When a chunk is in normal status, the data contained in it can be manipulated by a data manipulation SQL statement. A chunk in normal status is created when background import is executed. You can use the `adbchgchunkstatus` command to change the status of a chunk from normal status to wait status.

chunk in wait status

When a chunk is in wait status, the data contained in it cannot be manipulated by a data manipulation SQL statement. However, a chunk in wait status can be deleted by using the `PURGE CHUNK` or `TRUNCATE TABLE` statement. A chunk in wait status is created when you specify the status of the chunk to be created while performing background import. You can use the `adbchgchunkstatus` command to change the status of a chunk from wait status to normal status. Note that a chunk in wait status does not become the current chunk.

chunk merging

The operation of merging (combining) multiple chunks into a single chunk. The `adbmergechunk` command is used to merge chunks.

Multiple chunks designated to be merged are called *merge-source chunks*, and the single chunk into which multiple chunks are merged is called the *merge-target chunk*.

chunk unarchiving

To decompress the data in an archive file by using the chunk archiving function and store the decompressed data in the data DB area. At this time, an index is also re-created.

client definition

A definition that specifies the execution environment of the HADB client.

client directory

A directory that stores a group of files related to a client process.

client-group facility

A facility for grouping multiple HADB clients together and specifying ranges for the numbers of connections and processing real threads that can be used by the group.

cold standby configuration

A system configuration in which a standby server machine (standby system) is provided in addition to the server machine (active system) that performs application processing. If the active system cannot continue application processing because of a failure, the standby system takes over as the active system and continues the application processing.

column store format

A format in which data is stored in the database at the column level. In column store format, the data in each column of a table is stored together in the database.

column store table

A table defined with column store format selected as the table-data storage format. There are two table-data storage formats: row store format and column store format. You can specify the table-data storage format when defining the table.

command status file

A file used to manage the status of commands, such as executing status and interrupted status.

common format audit trail files

Audit trail files in a format in which the information can be collected, viewed, and managed with JP1/Audit. When audit trails that have been output to audit trail files are converted by using the `adbconvertaudittrailfile` command, the converted audit trails are output to common format audit trail files.

CONNECT privilege

A user privilege needed by an HADB user in order to connect to the HADB server. This is a prerequisite user privilege for using the HADB server. Without the `CONNECT` privilege, a user cannot connect to the HADB server even if he or she has other privileges.

connection handle

An object that manages connections (between HADB clients and the HADB server).

correction search

A function that performs a text data search, ignoring the differences between uppercase and lowercase letters, between half-width and full-width characters, and between Japanese hiragana and katakana characters.

cost information

Information related to retrieval methods that is collected by the HADB server from base tables, B-tree indexes, and text indexes for the purpose of optimizing table retrieval processing. Executing the `adbgetcst` command collects cost information and stores it in a system table.

current chunk

Among the created chunks, this is the chunk to which a user is adding data. Only one current chunk exists in a base table.

The current chunk is changed in the following cases:

- When a new chunk is created using background import

- When multiple chunks are merged, including the current chunk

The current chunk might also be changed in the following case:

- When the status of a chunk is changed

D

data DB area

A DB area that stores tables and indexes.

data import

A process of storing a full table's worth of data in one step. The `adbimport` command is used to import data.

data manipulation SQL statement

An SQL statement used to manipulate table data (retrieval SQL statements and update SQL statements).

data reorganization

The process of exporting table data and then importing the exported data, for the purpose of deleting the invalid row data in the table. Data reorganization can be performed for an entire table (table reorganization) or for each chunk (chunk reorganization).

DB area

A logical area that stores tables, indexes, and their definition information. DB areas are classified into the following according to their uses:

- Master directory DB area
- Dictionary DB area
- System-table DB area
- Work table DB area
- Data DB area

DB area file

A file that makes up a DB area.

DB directory

A directory that stores a group of database-related files, such as DB area files and system log files.

DBA privilege

A user privilege required for managing HADB users and privileges (user privileges and the schema operation privilege).

definition SQL statement

A generic term for SQL statements used for the following purpose:

- Defining and deleting schemas, tables, and indexes
- Creating and deleting HADB users

- Granting privileges to HADB users
- Revoking HADB users' privileges

The SQL statements described under *Definition SQL* in the manual *HADB SQL Reference* qualify as definition SQL statements.

delete-pending chunk

An unnecessary chunk (merge-source chunk) that remains undeleted because a chunk-merging process was interrupted. An unnecessary chunk (data before reorganization) that remains undeleted because reorganization of a system table (base table) was interrupted also applies.

The data contained in this chunk cannot be manipulated by a data manipulation SQL statement. However, this chunk can be deleted by using the `PURGE CHUNK` or `TRUNCATE TABLE` statement. Note that the status of this chunk cannot be changed by using the `adbchgchunkstatus` command.

dependent privilege

An access privilege granted to an HADB user by another HADB user.

Example:

- The access privileges HADB user A has for table `X.T1` shall be called `P1`.
- The access privileges HADB user A grants to HADB user B for table `X.T1` shall be called `P2`.
- The access privileges HADB user B grants to HADB user C for table `X.T1` shall be called `P3`.

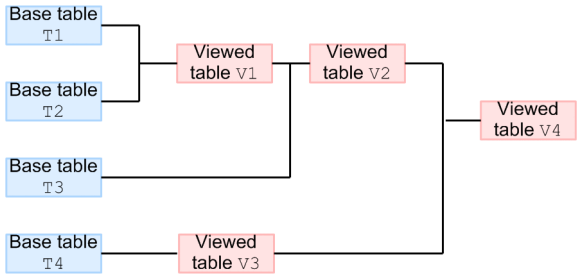
In this scenario, the dependent privileges of the access privileges `P1` are the access privileges `P2` and `P3`. The dependent privileges of the access privileges `P2` are the access privileges `P3`. The access privileges `P3` do not have any dependent privileges.

For example, suppose that the access privileges `P1` are revoked. In this case, access privileges `P2` and `P3` which are dependent privileges are also revoked. If the access privileges `P2` are revoked, the access privileges `P3` which are dependent privileges are also revoked.

dependent viewed table

A viewed table that is affected when, for example, a base table is deleted by using the `DROP TABLE` statement or when a viewed table is deleted by using the `DROP VIEW` statement. The following figure shows an example of dependent viewed tables.

Example of dependent viewed tables



Explanation

- Viewed tables `V1`, `V2`, and `V4` are dependent on base table `T1`.

- Viewed tables V1, V2, and V4 are dependent on base table T2.
- Viewed tables V2 and V4 are dependent on base table T3.
- Viewed tables V3 and V4 are dependent on base table T4.
- Viewed tables V2 and V4 are dependent on viewed table V1.
- Viewed table V4 is dependent on viewed table V2.
- Viewed table V4 is dependent on viewed table V3.
- There is no dependent viewed table for viewed table V4.

Operations other than deletion of a base table or viewed table might also affect dependent viewed tables. For details about the operations that affect dependent viewed tables, see (3) Invalidating viewed tables in 2.1.2 Viewed tables.

dictionary DB area

A DB area that stores dictionary tables and their indexes.

dictionary table

A viewed table that is created from a dictionary table (base table) for search purposes. The schema name of a dictionary table is MASTER.

dictionary table (base table)

A table that stores database definition information, such as table definitions, column definitions, and index definitions. An HADB user cannot search a dictionary table (base table). To check the information stored in a dictionary table (base table), the user must search the dictionary table.

directory for storing synonym dictionary files

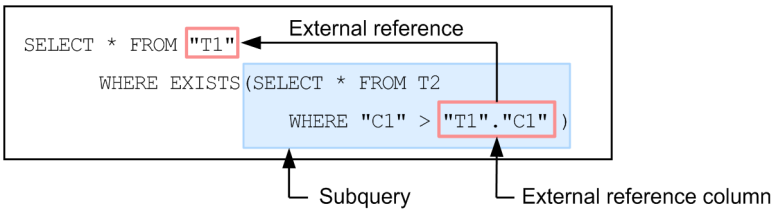
The directory in which synonym dictionary files are stored. The directory for storing synonym dictionary files is the directory specified in the adb_syndict_storage_path operand in the server definition.

E

external reference column

A column that uses a search condition in a subquery to reference a table specified in the FROM clause of an external query. Referencing a table by using a search condition in a subquery is called external reference. The following figure shows an example of an external reference column.

Example of external reference column



F

foreign key

A column (or a combination of multiple columns) that references the primary key of another base table. Primary keys and foreign keys can be used to define relationships among tables in a database.

G

global buffer

A buffer used for inputting/outputting data stored on a disk in DB areas. Global buffers are allocated in shared memory.

grant option

A privilege that allows an HADB user to grant access privileges to another HADB user.

H

HADB administrator

An OS user who manages HADB. The HADB administrator is the owner of the server directory and the DB directory, and can execute all HADB commands.

HADB administrators group

An HADB-specific group that is set in the server machine's OS. The HADB administrators group becomes the primary group to which the following OS users belong:

- HADB administrator
- OS user who belongs to the HADB administrators group

HADB administrators group OS user

See *OS user who belongs to the HADB administrators group*.

HADB client

A client-side system in HADB that operates as part of the client-server system.

HADB server

A server-side system in HADB that operates as part of the client-server system.

HADB user

A user who, by connecting to the HADB server, can take actions such as importing data, retrieving or updating data, and managing HADB users. Privileges needed to execute various types of operations can be granted to an HADB user.

hash retrieval

Processing that uses a hash table area. The following types of processing apply:

- Hash join as a table joining method
- Global hash grouping as a grouping method

- Hash execution as a method for processing subqueries
- Hash execution as a method for processing `SELECT DISTINCT`
- Hash execution as a method for processing the set operation

host reset

A function of HA Monitor that protects the data on a shared disk. You must select one of the following as the shared disk data protection method used by HA Monitor. We recommend that you select host reset.

- Host reset
- SCSI reservation for shared disk

For details about host reset and SCSI reservation for shared disk, see *Host reset* and *SCSI reservation for shared disk* in the manual *HA Monitor for Linux(R) (x86)*.

I

index rebuilding

The operation of rebuilding an index defined for a base table. The `adbidxrebuild` command is used to rebuild an index.

installation directory

A directory that is created when the HADB server and HADB clients are installed.

invalid row data

The data from a deleted row or the original data for a row that has since been updated is not deleted from the disk. This data that is present on the disk but is not subject to retrieval is called invalid row data. If you repeatedly perform operations that delete and update rows, the amount of invalid row data in the table increases. If invalid row data increases, it might affect as follows:

- Data storage efficiency drops due to the increase in the amount of invalid row data.
- Retrieval performance slows down because the number of pages to be referenced increases during retrieval processing.

L

location table

A table that is created automatically by the HADB server when the data in chunks is archived. A location table stores the following types of information:

- Chunk ID
- Archive file path
- Range of values (maximum and minimum values) in the archive range column of each archive file

Location tables are used by the HADB server and are not used by users.

One location table is created for each archivable multi-chunk table.

M

master directory

Internal system information used by HADB for managing the various types of information related to DB areas.

master directory DB area

A DB area that stores the master directory.

master node

The node containing the HADB server that controls the multi-node function. The master node corresponds to the active system in HA Monitor.

The HADB server on the master node accepts application and command connection requests and distributes processing to slave nodes with lighter loads.

Update SQL statements and commands such as the `adbimport` command are processed on the master node.

master node switchover

Action taken when a node failure occurs on the master node so that the master node is separated from the multi-node configuration and one of the slave nodes becomes the master node.

message catalog file

A message object file that stores the test of the messages that are output by HADB.

message log file

A file that stores the messages that are output by HADB.

multi-chunk table

A base table in which multiple chunks can be created. For a multi-chunk table, the background-import facility can be used. Multi-chunk tables can be categorized into two types: regular multi-chunk tables and archivable multi-chunk tables.

multi-node configuration

A system configuration consisting of multiple HADB servers that are using the multi-node function.

multi-node function

This function realizes the load distribution of retrieval SQL statement processing by coordinating multiple HADB servers.

multi-node synonym dictionary storage directory

A directory that stores the synonym dictionary files used to perform synonym searches when using the multi-node function. The multi-node synonym dictionary storage directory is the directory specified in the `adb_syndict_node_storage_path` operand in the server definition.

N

node

A node is a server machine on which an HADB server is installed, including the operating system, HADB server, and HA Monitor installed on that server machine. A node corresponds to a host in HA Monitor.

node failure

In this manual, a node failure means a failure that requires a node to be separated from a multi-node configuration. A node failure corresponds to a host failure or a server failure in HA Monitor.

node separation

A slave node in which a node failure occurred is disconnected from the multi-node configuration. This is called *node separation*.

NOT NULL constraint

A constraint that does not permit null values for column values.

O

OS user

A user who logs on to the OS and can utilize the OS functions. In HADB, the following three types of OS users operate the HADB server:

- Superuser
- HADB administrator
- OS user who belongs to the HADB administrators group

OS user who belongs to the HADB administrators group

An OS user who manages HADB separately from the HADB administrator. The OS user can access the server directory and DB directories owned by the HADB administrator. The OS user can also execute some of the HADB commands.

out-of-order execution

A database search method that is used by HADB. In out-of-order execution, a search process is divided into row units and executed in parallel.

P

page

One of the units used for storing data. It is the smallest unit for disk I/O operations. When data is read from or written into a disk, it is done in page units.

primary key

A column (or combination of columns) that uniquely identifies rows in a table. Defining a primary key allows you to maintain the uniqueness of the data in a base table. When you define a primary key, the following constraints apply to the columns comprising the primary key:

- Uniqueness constraint

- `NOT NULL` constraint

privilege

An HADB user needs privileges in order to perform various types of operations on an HADB server. HADB privileges can be roughly classified into the following three types:

- User privileges
- Schema operation privilege
- Access privilege

An HADB user can have multiple privileges. An HADB user who does not have the privileges corresponding to a certain operation cannot execute that operation.

processing real thread

A real thread that is allocated when the HADB server starts, and that is used for executing SQL statements (data import, for example).

pseudo thread

A thread that is created by HADB in order to execute, in parallel, multiple processes in a single real thread. Processes are executed in parallel by executing multiple pseudo threads in a single real thread (only one pseudo thread can actually be executing at any time).

R

range index

An index that manages the minimum and maximum values (a range) in the column of data stored in chunks and segments of a table.

real thread

A thread managed by the OS (kernel and library).

referenced table

A base table for which a primary key is defined, and which is referenced from a referencing table using a foreign key.

referencing table

A base table that defines a referential constraint and a foreign key.

referential constraint

A constraint that maintains a relationship between tables (referential consistency of data). You can define a referential constraint for a specific column (foreign key column) when you define a base table.

regular multi-chunk table

A base table for which the background-import facility, which is a basic function for multi-chunk tables, can be used. For regular multi-chunk tables, the chunk archiving function cannot be used.

reload

The process of restoring the data in unload files to a system table as part of the reorganization of a system table.

retrieval SQL statement

An SQL statement used to search table data. The `SELECT` statement qualifies as a retrieval SQL statement.

returning a node

Action taken to return a node that is separated from the multi-node configuration (due to a node failure) to the multi-node configuration (after correction of the failure). Nodes can be returned to the multi-node configuration as only slave nodes.

row store format

A format in which data is stored in the database at the row level. In row store format, one row of data is stored in the database as one record.

row store table

A table defined with row store format selected as the table-data storage format. There are two table-data storage formats: row store format and column store format. You can specify the table-data storage format when defining the table.

S

schema

A logical concept that includes tables and indexes. In HADB, each HADB user can own only one schema.

schema definition privilege

A schema operation privilege that an HADB user needs in order to define or delete schemas, tables, and indexes.

schema object

An element that can be defined in a schema. A table or index is an example of schema object.

schema operation privilege

A privilege needed for managing an HADB user's schemas. The following is the only type of schema operation privilege:

- Schema definition privilege

SCSI reservation for shared disk

A function of HA Monitor that protects the data on a shared disk. You must select one of the following as the shared disk data protection method used by HA Monitor. We recommend that you select host reset.

- Host reset
- SCSI reservation for shared disk

For details about host reset and SCSI reservation for shared disk, see *Host reset* and *SCSI reservation for shared disk* in the manual *HA Monitor for Linux(R) (x86)*.

segment

A unit for storing data that is used when allocating areas in files to tables and indexes. A segment consists of contiguous pages.

Segment size means the number of pages in a segment.

server directory

A directory that stores the files needed by the HADB server for execution. It stores a group of files related to the server process.

single-chunk table

A base table in which only one chunk can be created. For a single-chunk table, the background-import facility and chunk archiving function cannot be used.

slave node

A node that serves as the location of an HADB server that uses the multi-node function to process retrieval SQL statements and some commands. A slave node corresponds to HA Monitor's standby system.

If the master node fails, one of the slave nodes becomes the master node.

sort code

A set of character codes that are used to sort or compare characters as defined in the ISO/IEC 14651:2011 standard. Correction search uses this as the basis of notation correction.

SQL processing real thread

A processing real thread that is used when executing SQL statements.

SQL trace information

Historical information about the execution of SQL statements that is collected by SQL tracing. The SQL trace information includes the execution time of SQL statements and access path information. SQL trace information is used for the following purposes:

- Determining the causes of errors in SQL statements
- Tuning SQL statements

SQL tracing

A function for outputting historical information about the execution of SQL statements (SQL trace information) to files (SQL trace files).

statement handle

An object for managing the SQL statements to be executed.

statistics log file

A file to which part of the HADB server's operation information is output. The output information can be checked by using the `adbstat` command. The following type of information is output:

- SQL statement statistical information

status file

A file for managing the operational statuses and termination modes of the HADB server.

synonym dictionary

A dictionary in which synonyms are listed.

synonym dictionary file

A file storing the information about synonym dictionaries. The HADB server looks in the synonym dictionary file when performing synonym search.

synonym group

A group of synonyms that are targeted by synonym search.

synonym list definition file

A file that contains a list of synonyms to be registered in a synonym dictionary. When synonym search is used to search for a word, all the synonyms registered as the synonym group of that word in the synonym list definition file are also used as the search strings.

synonym search

A function that you can use when searching literatures, research papers, and other document data for a specific word. This function allows you to search also for synonyms of the word at the same time.

system log

Database update history information that is automatically acquired by HADB. The system log is used for recovering a database after an error has occurred in it.

system log file

A file to which system logs are output.

system table

A viewed table that is created from a system table (base table) for search purposes. The schema name of a system table is `MASTER`.

system table (base table)

A table for storing the following types of information:

- Cost information for base tables, B-tree indexes, and text indexes
- Chunk information for multi-chunk tables

An HADB user cannot search a system table (base table). To check the information stored in a system table (base table), the user must search the system table.

system table reorganization

The process of releasing areas of invalid row data in system tables.

system-table DB area

The DB area for storing system tables and indexes of system tables.

T

table scan

A method of retrieving base tables without using B-tree indexes or text indexes.

table-function derived table

Data in tabular format that has been derived by the following system-defined functions:

- `ADB_AUDITREAD` function
- `ADB_CSVREAD` function

text index

An index that manages information about the position of data (character strings) stored in tables. A text index can be used when character string data to be retrieved contains a specified character string.

U

underlying table

A table that becomes the base for a viewed table. The table specified in the query expression of a `CREATE VIEW` statement is the underlying table.

uniqueness constraint

A constraint that does not permit duplicate column values (or combinations of column values from multiple columns).

unload

A process of outputting, to an unload file, system table data from which invalid row data has been excluded.

unload file

A temporary file created during reorganization of a system table.

update SQL statement

An SQL statement used to add data to a table, deleting data from a table, or updating data in a table. The following SQL statements qualify as update SQL statements:

- `INSERT` statement
- `DELETE` statement
- `UPDATE` statement
- `PURGE CHUNK` statement
- `TRUNCATE TABLE` statement

updated-row columnizing facility

This is an HADB facility that automatically converts data stored in row store format in a column store table to column store format. When the `INSERT` or `UPDATE` statement is executed for a column store table, the added or updated data is stored in row store format in the column store table. If this facility is enabled, the data stored in row store format is automatically converted to column store format.

user privilege

A privilege that an HADB user needs in order to manage HADB users or connect to the HADB server. The following user privileges are available:

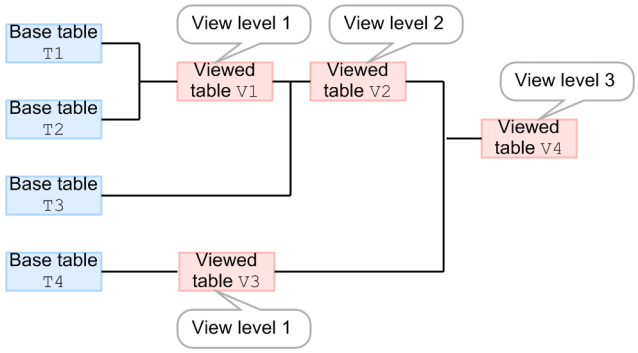
- `DBA` privilege
- `CONNECT` privilege

V

view level

A value that indicates the hierarchical depth of a defined viewed table from the base table. The following figure shows an example of view levels.

Example of view levels



Explanation

- The view level of viewed table `V1` is 1 because only base tables underlie `V1`.
- The view level of viewed table `V2` is 2 because a base table and a viewed table at view level 1 underlie `V2`.
- The view level of the viewed table `V3` is 1 because only a base table underlies `V3`.
- The view level of viewed table `V4` is 3 because the underlying tables of `V4` are at view levels 1 and 2.

viewed table

A virtual table created by defining the result of the query expression specified in the `CREATE VIEW` statement as a new table. For example, you can define the result of a set operation for multiple tables as a viewed table, or can define the specific rows or columns in a table as a viewed table.

W

word-context search

A function that quickly searches English-language text data for specific English words specified as search keys. There are two methods of word-context search as follows:

- Complete-match retrieval
- Leading-match search

work table DB area

A DB area that stores the tables (work tables) HADB creates internally when executing SQL statements.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Hitachi, Ltd.
6-6, Marunouchi 1-chome, Chiyoda-ku, Tokyo, 100-8280 Japan