

For UNIX Systems

Nonstop Database

HiRDB Version 9

Installation and Design Guide

3000-6-452-50(E)

■ Relevant program products

List of program products:

For Red Hat Enterprise Linux 6 (64-bit x86_64) operating systems:

P-9W62-4592 HiRDB Server Version 9 09-50

P-9W62-2D92 HiRDB/Run Time Version 9(64) 09-50

P-9S62-2B92 HiRDB/Run Time Version 9 09-50

P-F9W62-11925 HiRDB Non Recover Front End Server Version 9 09-00

P-F9W62-11926 HiRDB Advanced High Availability Version 9 09-00

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

■ Trademarks

HITACHI, JP1, uCosminexus, HiRDB, DocumentBroker, HA Monitor are either trademarks or registered trademarks of Hitachi, Ltd. in Japan and other countries.

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

AMD is a trademark of Advanced Micro Devices, Inc.

IBM, AIX are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, AIX 5L are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, DataStage, MetaBroker, MetaStage and QualityStage are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, DB2 are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, HACMP are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, OS/390 are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, PowerHA are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

Itanium is a trademark of Intel Corporation in the United States and other countries.

Microsoft and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Microsoft Access is a registered trademark of Microsoft Corporation in the U.S. and other countries.

Microsoft Office and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Motif is a registered trademark of the Open Software Foundation, Inc.

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

ODBC is Microsoft's strategic interface for accessing databases.

OLE is the name of a software product developed by Microsoft Corporation and the acronym for Object Linking and Embedding.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

VERITAS is a trademark or registered trademark of Symantec Corporation in the U.S. and other countries.

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

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

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

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

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

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

Other product and company names mentioned in this document may be the trademarks of their respective owners. Throughout this document Hitachi has attempted to distinguish trademarks from descriptive terms by writing the name with the capitalization used by the manufacturer, or by writing the name with initial capital letters. Hitachi cannot attest to the accuracy of this information. Use of a trademark in this document should not be regarded as affecting the validity of the trademark.

■ 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.

No part of this material may be reproduced in any form or by any means without permission in writing from the publisher.

Printed in Japan.

■ Issued

Oct. 2015: 3000-6-452-50(E)

■ Copyright

All Rights Reserved. Copyright (C) 2012, 2015, Hitachi, Ltd.

Preface

This manual describes the system definitions for HiRDB Version 9, a nonstop database server program product. For details about prerequisite information that is not provided here, see the manual *HiRDB Version 9 Description* (3020-6-450).

■ Intended readers

This manual is intended for users who will build and/or operate a relational database system using HiRDB Version 9 (hereafter referred to as HiRDB).

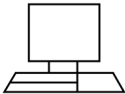


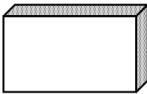
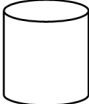


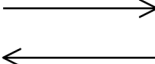
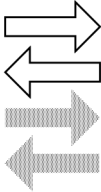


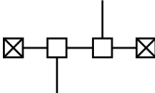
Readers of this manual must have:

- A basic knowledge of managing UNIX or Linux systems
- A basic knowledge of SQL

Please read the *HiRDB Version 9 Description* manual before reading this manual.

■ Conventions: Diagrams

This manual uses the following conventions in diagrams:

- | | | | |
|---|---|---|---|
| • Workstation or personal computer | • I/O operation | • Screen display | • Program or server |
|  |  |  |  |
| • File or magnetic disk | • Magnetic tape | • Communication line | • Flow of control |
|  |  |  |  |
| • Flow of data | • Work procedure | • Network | • LAN |
|  |  |  |  |

■ Conventions: Fonts and symbols

The following table explains the text formatting conventions used in this manual:

Font	Convention
Bold	<p>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:</p> <ul style="list-style-type: none">• From the File menu, choose Open.• Click the Cancel button.• In the Enter name entry box, type your name.

Font	Convention
<i>Italics</i>	<p><i>Italics</i> are used to indicate a placeholder for some actual text to be provided by the user or system. For example:</p> <ul style="list-style-type: none"> Write the command as follows: <code>copy source-file target-file</code> The following message appears: A file was not found. (file = <i>file-name</i>) <p><i>Italics</i> are also used for emphasis. For example:</p> <ul style="list-style-type: none"> Do <i>not</i> delete the configuration file.
Code font	<p>A code font indicates text that the user enters without change, or text (such as messages) output by the system. For example:</p> <ul style="list-style-type: none"> At the prompt, enter <code>dir</code>. Use the <code>send</code> command to send mail. The following message is displayed: <code>The password is incorrect.</code>

The following table explains the symbols used in this manual:

Symbol	Convention
	<p>In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR. For example: A B C means A, or B, or C.</p>
[]	<p>In syntax explanations, square brackets indicate that the enclosed item or items are optional. For example: [A] means that you can specify A or nothing. [B C] means that you can specify B, or C, or nothing.</p>
...	<p>In coding, an ellipsis (...) indicates that one or more lines of coding are not shown for purposes of brevity.</p> <p>In syntax explanations, an ellipsis indicates that the immediately preceding item can be repeated as many times as necessary. For example: A, B, B, ... means that, after you specify A, B, you can specify B as many times as necessary.</p>
()	<p>Parentheses indicate the range of items to which the vertical bar () or ellipsis (...) is applicable.</p>

■ 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*.

■ Important notes on this manual

The following facilities are explained, but they are not supported:

- Distributed database facility
- Standby-less system switchover (1:1) facility
- Standby-less system switchover (effects distributed) facility
- HiRDB External Data Access facility
- Inner replica facility
- Updatable online reorganization
- Sun Java System Directory Server linkage facility
- Simple setup tool
- Extended syslog facility
- Rapid batch facility
- Memory database facility
- Linkage with JP1/NETM/Audit

The following products and option program products are explained, but they are not supported:

- HiRDB Disaster Recovery Light Edition
- uCosminexus Grid Processing Server
- HiRDB Text Search Plug-in
- HiRDB XML Extension
- TP1/Server Base
- JP1/PFM-Agent Option for HiRDB
- JP1/VERITAS NetBackup Agent for HiRDB License
- HiRDB Dataextractor
- XDM/RD
- HiRDB SQL Tuning Advisor
- COBOL2002
- HiRDB Control Manager - Console
- HiRDB Control Manager - Server

Contents

1	Overview of HiRDB System Construction	1
1.1	Overview of behavior when operands are omitted	2
1.2	System construction procedures	3
1.2.1	System construction procedure for installing a new HiRDB	3
1.2.2	Setting up a HiRDB environment	3
1.2.3	Environment setup for linking to other products	4
1.3	Organization of HiRDB directories and files	6
1.3.1	Initial files that are created	6
1.3.2	Files that consistently increase in size	10
1.4	Upgrading HiRDB	30
1.4.1	Before upgrading	30
1.4.2	Replacing an existing version with the new version	35
1.4.3	Installing a new version while retaining the old version	37
1.4.4	Upgrading the HiRDB plug-ins	37
1.4.5	Using Java stored procedures and functions	37
1.4.6	In the event of an upgrading error	38
1.4.7	Restoring an earlier version of HiRDB	40
1.5	Updating to HiRDB update version	42
1.5.1	Updating HiRDB	42
1.5.2	Prerequisites	43
1.5.3	Update procedure	43
1.5.4	Update procedure when the system switchover facility is used	44
1.5.5	Cautions	45
1.5.6	Operation considerations	45
1.5.7	Related product limitations and considerations	46
1.5.8	Operation when an error occurs during installation	46
1.6	Using JP1/Software Distribution to execute online distribution of HiRDB	48
1.6.1	Online distribution procedure	48
1.6.2	Preparations on JP1/Software Distribution Workstation	49
1.6.3	Notes about online distribution	52
1.7	Migrating to 64-bit mode HiRDB	53
1.7.1	Considerations when migrating to 64-bit mode	53
1.7.2	How to migrate to 64-bit mode	53
1.7.3	In the event of an SQL object migration error	55
1.7.4	In the event of a 64-bit-mode migration error (restoring the old version)	56
2	Installation	57
2.1	Pre-installation procedure	58

2.1.1	Checking and modifying OS parameters	58
2.1.2	Registering the HiRDB administrator	58
2.1.3	Setting up a HiRDB group	59
2.1.4	Creating the installation directory	59
2.1.5	Registering host names	59
2.2	HiRDB installation procedure	61
2.2.1	Installing HiRDB	61
2.2.2	Installing option program product	61
2.2.3	Installing plug-ins	62
2.3	Post-installation procedures	63
2.3.1	Creating the HiRDB directory	63
2.3.2	Creating a work file output directory	65
2.3.3	Registering HiRDB and option program products in the operating system	66
2.3.4	Setting environment variables	68
2.3.5	Setting a remote shell execution environment	70
2.3.6	Notes about background execution of HiRDB operation commands	70
2.3.7	Preparing to create the HiRDB file system area	70
2.4	Uninstallation of HiRDB	75

3

Setting Up an Environment Using the Simple Setup Tool	77
3.1 Overview of the simple setup tool	78

4

Setting Up an Environment Using Commands	79
4.1 Overview of environment setup using commands	80
4.2 Creating the HiRDB system definitions	82
4.2.1 Creating HiRDB system definitions (HiRDB single server configuration)	82
4.2.2 Creating HiRDB system definitions (HiRDB parallel server configuration)	84
4.2.3 Sharing HiRDB system definition files (HiRDB parallel server configuration)	87
4.2.4 Modifying HiRDB system definitions (excluding UAP environment definitions)	89
4.2.5 Modifying a UAP environment definition	90
4.3 Creating HiRDB file system areas	92
4.3.1 Types of HiRDB file system areas	92
4.3.2 Using character special files	92
4.3.3 Creating a large file	93
4.3.4 Example 1 (creating a HiRDB file system area for RDAREAs)	93
4.3.5 Example 2 (creating a HiRDB file system area for system files)	93
4.3.6 Example 3 (creating a HiRDB file system area for work table files)	94
4.3.7 Example 4 (creating a HiRDB file system area for utilities)	94
4.3.8 Example 5 (creating a HiRDB file system area for list RDAREAs)	95
4.4 Creating system files	96
4.4.1 Creating system log files	96

4.4.2	Creating synchronization point dump files	96
4.4.3	Creating status files	97
4.4.4	Example of system file creation (HiRDB single server configuration)	97
4.4.5	Example of system file creation (HiRDB parallel server configuration)	100
4.5	Creating system RDAREAs	107
4.5.1	Basics	107
4.5.2	Example 1 (HiRDB single server configuration)	107
4.5.3	Example 2 (HiRDB parallel server configuration)	108
4.6	Starting HiRDB for the first time	110
4.7	Creating user RDAREAs	111
4.7.1	Basics	111
4.7.2	Example 1 (HiRDB single server configuration)	111
4.7.3	Example 2 (HiRDB parallel server configuration)	112
4.8	Creating user LOB RDAREAs	114
4.8.1	Basics	114
4.8.2	Example 1 (HiRDB single server configuration)	114
4.8.3	Example 2 (HiRDB parallel server configuration)	115
4.9	Creating data dictionary LOB RDAREAs	117
4.9.1	Basics	117
4.9.2	Example 1 (HiRDB single server configuration)	117
4.9.3	Example 2 (HiRDB parallel server configuration)	118
4.10	Creating list RDAREAs	120
4.10.1	Basics	120
4.10.2	Example 1 (HiRDB single server configuration)	120
4.10.3	Example 2 (HiRDB parallel server configuration)	121

5

Setting Up the Plug-in Environment	123
5.1 Overview of plug-in environment setup	124
5.1.1 Environment setup procedure	124
5.1.2 Notes on using plug-ins	128
5.2 Upgrading plug-ins	130
5.3 Deleting plug-ins	132

6

Creating Databases	135
6.1 Overview of database creation	136
6.1.1 Preparing for database creation	136
6.1.2 Database creation procedure	137
6.1.3 Database update log acquisition mode	137
6.1.4 Notes on data storage for a table for which an index with the unique attribute has been defined	140
6.1.5 Loading a large amount of data (data loading with the synchronization point specification)	140
6.1.6 Loading data into a row-partitioned table (using the parallel loading facility)	141

6.1.7 Loading data into a row-partitioned table (Creating divided-input data files)	150
6.1.8 Data loads that use the automatic numbering facility	151
6.1.9 Input data file UOC	152
6.1.10 Deleting unneeded RDAREAs	152
6.2 Creating a row-partitioned table	153
6.3 Creating a table with a LOB column	156
6.4 Creating a table containing a plug-in-provided abstract data type	159
6.4.1 The SGMLTEXT type	159
6.4.2 The XML type	162
6.5 Creating a table containing a user-defined abstract data type	176
6.5.1 Defining an abstract data type	176
6.5.2 Defining a table	179
6.5.3 Defining an index	180
6.5.4 Storing data in a table	180
6.5.5 Database update log acquisition methods	181
6.5.6 Checking the data storage status	182
6.6 Handling errors during batch index creation	183
6.6.1 When data was loaded in log acquisition mode or pre-update log acquisition mode	183
6.6.2 When data was loaded in no-log mode	185
6.7 Handling utility abnormal termination errors during data loading with the synchronization point specification	188
6.7.1 Overview of error handling procedure	188
6.7.2 Example	189

7

Linking to Other Products	191
7.1 Linking to the replication facility	192
7.1.1 Linking to HiRDB Datareplicator	192
7.1.2 Linking to HiRDB Dataextractor	192
7.2 Linking with an OLTP system	193
7.2.1 OLTP products supported for linking	193
7.2.2 HiRDB XA library	193
7.2.3 Example of HiRDB system configuration with OLTP linkage	195
7.2.4 Transaction transfer	197
7.2.5 Registering HiRDB in the transaction manager	198
7.2.6 Information to be registered in the transaction manager	200
7.2.7 Example of registering in the transaction manager	204
7.2.8 Modifying the registration information in the transaction manager	206
7.2.9 Methods for re-establishing connection between the transaction manager and HiRDB	207
7.2.10 Monitoring HiRDB using the TP1/Resource Manager Monitor facility	208
7.2.11 Notes	209
7.3 Linking to the inner replica facility	211

7.4 Linking to JP1	212
7.4.1 Reporting events to JP1/Base	212
7.4.2 Managing events by JP1/IM	214
7.4.3 Automatic job execution using JP1/AJS3 linkage	215

8

Designing a HiRDB single server configuration 217

8.1 System design for a HiRDB single server configuration	218
8.1.1 System design	218
8.1.2 System configuration	220
8.2 Designing HiRDB file system areas	221
8.2.1 Designing HiRDB file system areas for RDAREAs	221
8.2.2 Designing HiRDB file system areas for system files	222
8.2.3 Designing HiRDB file system areas for work table files	223
8.2.4 Designing HiRDB file system areas for utilities	223
8.2.5 Designing HiRDB file system areas for list RDAREAs	224
8.2.6 Maximum sizes of HiRDB file system areas	225
8.3 Designing system files	226
8.3.1 Designing system log files	226
8.3.2 Designing synchronization point dump files	230
8.3.3 Designing status files	232
8.4 Placing RDAREAs	236
8.4.1 Placing system RDAREAs	236
8.4.2 Placing data dictionary LOB RDAREAs	236
8.4.3 Placing user RDAREAs	237
8.4.4 Placing user LOB RDAREAs	237
8.4.5 Placing list RDAREAs	238

9

Designing a HiRDB parallel server configuration 239

9.1 System design for a HiRDB parallel server configuration	240
9.1.1 System design	240
9.1.2 System configuration of a HiRDB parallel server configuration	242
9.1.3 Setting up multiple front-end servers	243
9.1.4 Recovery-unnecessary front-end server	246
9.2 Designing HiRDB file system areas	250
9.2.1 Designing HiRDB file system areas for RDAREAs	250
9.2.2 Designing HiRDB file system areas for system files	251
9.2.3 Designing HiRDB file system areas for work table files	252
9.2.4 Designing HiRDB file system areas for utilities	252
9.2.5 Designing HiRDB file system areas for list RDAREAs	253
9.2.6 Maximum sizes of HiRDB file system areas	254
9.3 Designing system files	255

9.3.1 Designing system log files	255
9.3.2 Designing synchronization point dump files	259
9.3.3 Designing status files	261
9.4 Placing RDAREAs	265
9.4.1 Placing system RDAREAs	265
9.4.2 Placing data dictionary LOB RDAREAs	266
9.4.3 Placing user RDAREAs	266
9.4.4 Placing user LOB RDAREAs	267
9.4.5 Placing list RDAREAs	267
9.5 Considerations that apply to building a system with many units or servers	269
9.5.1 Considerations that apply to configuring systems	269
9.5.2 Considerations for system operation	270
9.5.3 Corrective actions to take in response to errors that occur when commands are executed	273

10 Designing a Multi-HiRDB	275
10.1 System design for a multi-HiRDB	276
10.1.1 Installing a multi-HiRDB	276
10.1.2 Setting the environment for a multi-HiRDB	276
10.2 Notes about upgrading	279

11 Designing Global Buffers and Local Buffers	281
11.1 Allocating global buffers	282
11.1.1 Allocating index global buffers	282
11.1.2 Allocating data global buffers	282
11.1.3 Allocating LOB global buffers	284
11.1.4 Global buffer allocation procedures	284
11.2 Setting the number of global buffer sectors	287
11.3 Specifying the prefetch facility	288
11.4 Specifying the asynchronous READ facility	289
11.5 Specifying deferred write processing	290
11.6 Specifying the facility for parallel writes in deferred write processing	291
11.7 Setting the commit-time reflection processing	292
11.8 Global buffer LRU management	293
11.8.1 LRU management methods	293
11.8.2 LRU management suppression settings for a UAP	294
11.8.3 Setting suppression of LRU management of binary data accessed by UAPs	295
11.9 Page access using the snapshot method	297
11.10 Global buffer pre-writing	299
11.11 Local buffers	300
11.11.1 Allocating index local buffers	300
11.11.2 Allocating data local buffers	300

11.11.3 Allocating local buffers	301
11.11.4 Considerations about local buffers	301

12

Designing Tables	303
12.1 Items to be examined during table design	304
12.2 Normalizing a table	308
12.3 Table row partitioning	312
12.3.1 Table row partitioning	312
12.3.2 Types of table row partitioning	312
12.3.3 Forms of table row partitioning	319
12.3.4 Effects of table row partitioning	320
12.3.5 Design considerations	321
12.3.6 Notes on table row partitioning	326
12.4 Table matrix partitioning	327
12.5 Defining a trigger	332
12.5.1 Application standards	332
12.5.2 Defining a trigger	333
12.5.3 Trigger considerations	336
12.5.4 Trigger management	336
12.5.5 Error recovery	339
12.6 Creating a view table	340
12.7 Specifying the FIX attribute	342
12.8 Specifying a primary key	344
12.9 Specifying a cluster key	345
12.10 Specifying the suppress option	347
12.11 Specifying the no-split option	348
12.12 Specifying a binary data column	350
12.12.1 BLOB type	351
12.12.2 BINARY type	351
12.12.3 BLOB type and BINARY type usage	352
12.13 Specifying a character set	353
12.14 Specifying the WITHOUT ROLLBACK option	354
12.15 Specifying the falsification prevention facility	356
12.15.1 Specification	356
12.15.2 Restrictions	357
12.15.3 Changing a falsification-unprevented table to a falsification prevented table	360
12.15.4 Error operation	362
12.16 Table containing a repetition column	363
12.17 Table containing an abstract data type	365
12.18 Shared tables	369
12.18.1 Effects and criteria	370

12.18.2	Definition method	371
12.18.3	Manipulating shared tables	371
12.18.4	Limitations on shared tables	372
12.18.5	Rules used to allocate back-end servers that search shared tables	372
12.18.6	Notes about execution of definition SQL statements, utilities, and operation commands	382
12.18.7	Using shared tables with a HiRDB single server configuration	383
12.19	Referential constraints	385
12.19.1	About referential constraints	385
12.19.2	Defining referential constraints	385
12.19.3	Check pending status	394
12.19.4	Data manipulation and integrity	400
12.19.5	Procedure for checking table integrity	403
12.19.6	Referential constraints and triggers	408
12.19.7	Notes about linkage with related products	410
12.20	Check constraints	412
12.20.1	About check constraints	412
12.20.2	Defining check constraints	412
12.20.3	Check pending status	413
12.20.4	Data manipulation and integrity	414
12.20.5	Procedure for checking table integrity	415
12.20.6	Notes about linkage with related products	418
12.20.7	Migrating check constraint tables to 64-bit mode	419
12.21	Compressed tables	422
12.21.1	Data compression facility	422
12.21.2	How data is compressed	423
12.21.3	How to define a compressed table	424
12.21.4	How to convert an existing table to a compressed table	424
12.21.5	How to change the definition of a compressed column (removing the compression specification for a column)	425
12.21.6	Notes about using compressed tables	426
12.21.7	How to measure the data compression rate	426
12.22	Temporary tables	428
12.22.1	Valid period of data in temporary tables	429
12.22.2	How to define temporary tables and temporary table indexes	430
12.22.3	Rules for choosing an RDAREA for storage	431
12.22.4	Processing when there are no available temporary table RDAREAs	432
12.22.5	Locking for temporary tables	434
12.22.6	Limitations on the use of temporary tables	435
13	Designing Indexes	437
13.1	Items to be examined during index design	438

13.2 Index	439
13.2.1 Creating an index	439
13.2.2 Index creation taking into account optimizing based on cost	440
13.2.3 Single-column index vs. multicolumn index	443
13.2.4 Using multiple indexes	444
13.2.5 Using an index with an exceptional key value set	444
13.2.6 Effects on performance of the number of indexes	445
13.3 Index row partitioning	446
13.4 Plug-in index	451
13.5 Plug-in index row partitioning	452

14

Designing RDAREAs	457
14.1 Items to be examined during RDAREA design	458
14.2 Segments	461
14.2.1 Determining the segment size	461
14.2.2 Setting the percentage of free pages in a segment	462
14.2.3 Allocating and releasing segments	463
14.3 Pages	464
14.3.1 Determining the page length	464
14.3.2 Setting the percentage of unused space in a page	465
14.3.3 Allocating and releasing pages	466
14.4 Designing list RDAREAs	468
14.5 Free space reuse facility	470
14.5.1 Data storage search modes	470
14.5.2 Free space reuse facility	470
14.5.3 Effects and applicability	472
14.5.4 Considerations	473
14.5.5 Environment settings	473
14.5.6 Checking execution status	475
14.5.7 Notes	475
14.6 Shared RDAREAs (HiRDB parallel server configuration only)	477
14.7 Temporary table RDAREAs	480

15

Storage Requirements for HiRDB	483
15.1 Estimating the memory size required for a HiRDB single server configuration	484
15.1.1 Memory allocation	484
15.1.2 Calculation of required memory	487
15.1.3 Formulas for shared memory used by a unit controller	494
15.1.4 Formulas for shared memory used by a single server	503
15.1.5 Formula for size of shared memory used by global buffers	507
15.1.6 Formulas for size of memory required during SQL execution	508

15.1.7 Formula for size of memory required during SQL preprocessing	514
15.1.8 Formula for size of memory required during BLOB data retrieval or updating (HiRDB single server configuration)	515
15.1.9 Formula for size of memory required during block transfer or array FETCH	516
15.1.10 Memory required by in-memory data processing	517
15.2 Estimating the memory size required for a HiRDB parallel server configuration	518
15.2.1 Memory allocation	518
15.2.2 Calculation of required memory	521
15.2.3 Formulas for shared memory used by a unit controller	529
15.2.4 Formulas for shared memory used by each server	552
15.2.5 Formula for size of shared memory used by global buffers	559
15.2.6 Formulas for size of memory required during SQL execution	562
15.2.7 Formula for size of memory required during SQL preprocessing	568
15.2.8 Formula for size of memory required during BLOB data retrieval or updating (front-end server)	569
15.2.9 Formula for size of memory required during BLOB data retrieval or updating (back-end server or dictionary server)	570
15.2.10 Formula for size of memory required during block transfer or array FETCH (front-end server)	571
15.2.11 Memory required by in-memory data processing	571

16

Determining RDAREA Size 573

16.1 Determining the size of a user RDAREA	574
16.1.1 Calculating the size of a user RDAREA	574
16.1.2 Calculating the number of table storage pages	575
16.1.3 Calculating the number of index storage pages	585
16.2 Determining the size of a data dictionary RDAREA	594
16.2.1 Determining the size of a normal data dictionary RDAREA	594
16.2.2 Determining the size of a data dictionary RDAREA for storing database state analyzed tables and database management tables	617
16.3 Determining the size of the master directory RDAREA	619
16.4 Determining the size of the data directory RDAREA	620
16.5 Determining the size of a data dictionary LOB RDAREA	621
16.6 Determining the size of a user LOB RDAREA	627
16.7 Determining the size of the registry RDAREA	628
16.8 Determining the size of the registry LOB RDAREA	630
16.9 Determining the size of the list RDAREA	631

17

Determining the Size of System Files and Audit Trail Files 633

17.1 Determining the size of system log files	634
17.1.1 Total size of system log files	634
17.1.2 Amount of system log information output during table definition	636
17.1.3 Amount of system log information output during index definition	637
17.1.4 Amount of system log information output during table data updating	639

17.1.5 Amount of system log information output during database creation by a utility	649
17.1.6 Amount of system log information that is output depending on the SQL manipulation	652
17.1.7 Amount of system log information that is output during the definition of extended system-defined scalar functions	652
17.1.8 Amount of system log information that is output during the execution of the RDAREA automatic extension facility	652
17.1.9 Amount of system log information output when the PURGE TABLE statement is executed	653
17.1.10 Amount of system log information output when the free page release utility (pdreclaim) is executed	653
17.1.11 Amount of system log information that is output during execution of the facility for predicting reorganization time	654
17.1.12 Amount of system log information output during an updatable backup hold	655
17.2 Determining the size of synchronization point dump files	657
17.3 Determining the size of status files	658
17.4 Determining audit trail file capacity	664

18

Determining Work Table File Size	665
18.1 Overview of work table files	666
18.2 Determining the size of a HiRDB file system area (pdfmkfs -n command)	668
18.2.1 Size of a work table file used by an SQL statement	668
18.2.2 Size of a work table file used by a utility	673
18.3 Determining the maximum number of files (pdfmkfs -l command)	675
18.4 Determining the maximum number of extensions (pdfmkfs -e command)	676

19

Storage Requirements for Utility Execution	677
19.1 Determining the file sizes required for utility execution	678
19.1.1 File sizes required for the execution of the database load utility (pdload)	678
19.1.2 File sizes required for the execution of the database reorganization utility (pdrorg)	680
19.1.3 File sizes required for the execution of the statistics analysis utility (pdstedit)	685
19.1.4 File sizes required for the execution of the database condition analysis utility (pddbstd)	688
19.1.5 File sizes required for the execution of the database copy utility (pdcopy)	688
19.1.6 File sizes required for the execution of the dictionary import/export utility (pdexp)	691
19.1.7 File sizes required for the execution of the optimizing information collection utility (pdgetcst)	692
19.1.8 File sizes required for the execution of the access path display utility (pdvwopt)	692
19.1.9 File sizes required for execution of the rebalancing utility (pdrbal)	693
19.1.10 File sizes required for execution of the integrity check utility (pdconstck)	694
19.1.11 File sizes required for execution of parallel loading (pdparaload)	695
19.1.12 Buffer size used to determine the size of the work file for sorting	695
19.2 Determining the memory size required for utility execution	697
19.2.1 Memory size required for the execution of the database initialization utility (pdinit)	697
19.2.2 Memory size required for the execution of the database definition utility (pddef)	698
19.2.3 Memory size required for the execution of the database load utility (pdload)	698
19.2.4 Memory size required for the execution of the database reorganization utility (pdrorg)	701

19.2.5 Memory size required for the execution of the database structure modification utility (pdmod)	703
19.2.6 Memory size required for the execution of the statistics analysis utility (pdstedit)	704
19.2.7 Memory size required for the execution of the database condition analysis utility (pddbstd)	705
19.2.8 Memory size required for the execution of optimizing the information collection utility (pdgetcst)	705
19.2.9 Memory size required for the execution of the database copy utility (pdcopy)	706
19.2.10 Memory size required for the execution of the database recovery utility (pdrstr)	707
19.2.11 Memory size required for the execution of the dictionary import/export utility (pdexp)	710
19.2.12 Memory size required for the execution of the access path display utility (pdvwopt)	711
19.2.13 Memory size required for the execution of the rebalancing utility (pdrbal)	711
19.2.14 Memory size required for execution of the free page release utility (pdreclaim) and global buffer residence utility (pdpgbfn)	715
19.2.15 Memory size required for execution of the integrity check utility (pdconstck)	715
19.2.16 Memory size required for the execution of parallel loading (pdparaload)	716

20 Specifying OS Parameters 719

20.1 Estimating HP-UX OS parameter values	720
20.2 Estimating Solaris OS parameter values	725
20.3 Estimating AIX OS parameter values	728
20.4 Estimating Linux kernel parameter values	732
20.5 Estimating the sizes of message queues and semaphores	736
20.6 Listen queue specified values	739

21 Sample Files 741

21.1 Overview of sample files	742
21.1.1 Names of sample files	742
21.2 System configuration and table definition information	745
21.3 Use of the sample files	749
21.3.1 Creating the configuration files	749
21.3.2 HiRDB file system area names and user-created file names used with sample database	753

22 Communication Between HiRDB Servers and HiRDB Clients 759

22.1 Connecting HiRDB clients to a HiRDB server	760
22.1.1 Connection to a HiRDB server with an FQDN specified	760
22.1.2 Using the multi-connection address facility to connect to a HiRDB server	761
22.2 Settings for a DNS server to manage IP addresses	766
22.3 Settings when a firewall and NAT are installed	768
22.3.1 When a firewall is installed on the HiRDB single server configuration side	768
22.3.2 When a firewall and NAT are installed on the HiRDB single server configuration side	769
22.3.3 When a firewall is installed on the HiRDB parallel server configuration side	770
22.3.4 When a firewall and NAT are installed on the HiRDB parallel server configuration side	771
22.4 Port numbers used by HiRDB	773

22.4.1 Estimating the number of ports that a unit will use	773
22.4.2 Notes	773
22.4.3 Calculation examples	774
22.4.4 Ways to avoid a shortage of ports	774
22.5 Port numbers specified in HiRDB	776
22.5.1 List of port numbers specified in HiRDB	776
22.5.2 Specifying port numbers	776
22.5.3 Notes on port number duplication	779
22.6 HiRDB reserved port facility	781
22.6.1 Estimation of the HiRDB reserved port facility	781

Appendixes 783

A. HiRDB Maximum and Minimum Values	784
A.1 Maximum and minimum values for the system configuration	784
A.2 Maximum and minimum values for databases	785
A.3 Maximum and minimum values for HiRDB file names	787
B. Processes Started by HiRDB	788
B.1 Processes started by a HiRDB single server configuration	788
B.2 Processes started by a HiRDB parallel server configuration	792
C. Questions and Answers	799

Index 807

1

Overview of HiRDB System Construction

This chapter provides an overview of the HiRDB system construction procedure, HiRDB file organization, and upgrading procedure.

1.1 Overview of behavior when operands are omitted

With each version and revision, HiRDB revises the default values that are assumed when you omit from the HiRDB system definition operands, utility options, and SQL options. To handle these changes to default values, HiRDB versions 09-50 and later offer two modes for the default behavior when operands are omitted: *recommended mode*, which assumes the recommended default values, and *compatibility mode*, which uses the default values for a specific version of HiRDB. Normally, to build a more secure system, we suggest that you apply the recommended mode, which greatly reduces the number of operands that must be specified.

For details about the advantages and disadvantages of changing the default values when you upgrade from a version earlier than 09-50, see *1.4.1(8) Checking the default values of operands in the HiRDB system definition* and *1.4.1(9) Checking other default values*. After checking the default values, if you decide to focus on compatibility with the previous version, you can apply the compatibility mode to make the default values the same as in the previous version. However, in this case all of the operands will have the previous version's default values, so we recommend specifying the recommended values on a per-operand basis.

If you want to update to the HiRDB update version, apply the operand default behavior that is already in effect.

The behavior when operands are omitted can be selected during setup of HiRDB. The behavior can also be changed using the command `pdsetenv`.

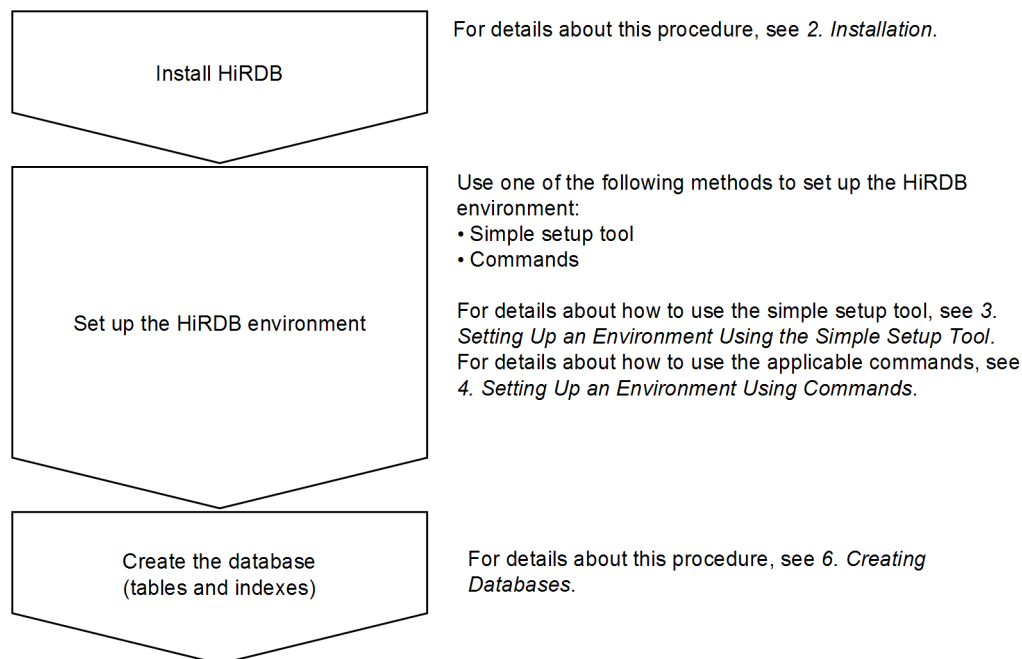
1.2 System construction procedures

This section discusses the system construction procedure for installing a new HiRDB.

1.2.1 System construction procedure for installing a new HiRDB

The following figure illustrates the system construction procedure for a new installation of HiRDB.

Figure 1–1: System construction procedure for installing a new HiRDB



For recommendations and notes on using HiRDB in continuous 24-hour operation, see the *HiRDB Version 9 System Operation Guide*.

1.2.2 Setting up a HiRDB environment

The HiRDB administrator uses one of the following methods to set up a HiRDB environment:

- The simple setup tool
- Commands

Tip

If you are setting up a HiRDB environment for the first time, we recommend that you use the simple setup tool.

The following table lists the advantages and disadvantages of each environment setup method.

Table 1–1: Advantages and disadvantages of each environment setup method

Setup method	Overview	Advantages	Disadvantages
Simple setup tool	Enter the HiRDB environment setup data according to the displayed windows. The HiRDB environment is set up based on the entered data.	Easier than using other methods to set up the HiRDB environment. If you use the simple setup tool to perform a standard setup or wizard setup, you can set up a test environment. You can also use it	The HiRDB system construction is limited to the range of configurations that can be set up using the simple setup tool.

Setup method	Overview	Advantages	Disadvantages
	For details about how to use the simple setup tool, see <i>Chapter 3. Setting Up an Environment Using the Simple Setup Tool</i> .	to change existing system definition settings. ^{#1}	
Commands ^{#2}	Use HiRDB commands to set up the HiRDB environment. For details about how to use commands, see <i>Chapter 4. Setting Up an Environment Using Commands</i> .	HiRDB commands allow you to tailor the HiRDB system configuration to your needs.	You will require certain knowledge to set up the HiRDB environment. Specifically, you need to understand the facilities and settings described in this manual. In addition, environment setup using commands is more difficult than other methods.

#1

The values generated by the simple setup tool are based on a HiRDB test environment. When you apply values to an actual environment, determine appropriate values and specify them instead of using the values generated by the simple setup tool.

#2

Try a simple installation before you construct the production system. Using the sample files to execute a series of HiRDB construction procedures on a test system makes it easier to create an actual production system.

! Important note

With the simple setup tool method, a plug-in environment cannot be set up.

1.2.3 Environment setup for linking to other products

This section discusses the environment setup using related products.

(1) Using the replication facility

To use the replication facility, you need HiRDB Datareplicator and HiRDB Dataextractor. For details about how to set up an environment for the replication facility, see *7.1 Linking to the replication facility*.

(2) Linking to OLTP

For details about the environment setup procedure to link your HiRDB to OLTP, see *7.2 Linking with an OLTP system*.

(3) Using the system switchover facility

To use the system switchover facility, cluster software is required. The specific cluster software to be used depends on the platform. For details about cluster software and the system switchover facility, see the *HiRDB Version 9 System Operation Guide*.

(4) Using the inner replica facility

To use the inner replica facility, HiRDB Staticizer Option is required. For details about the environment setup procedure, see *7.3 Linking to the inner replica facility*.

(5) Using Real Time SAN Replication (disaster recovery)

To use Real Time SAN Replication with the log-only synchronous method, HiRDB Disaster Recovery Light Edition is required.

For details about Real Time SAN Replication, see the *HiRDB Version 9 Disaster Recovery System Configuration and Operation Guide*.

(6) When linking to JP1

You can manage HiRDB events as JP1 events by linking your system to JP1. When JP1 is linked, you can achieve centralized management of system events and use events to start jobs automatically. For details about linkage with JP1, see *7.4 Linking to JP1*.

1.3 Organization of HiRDB directories and files

1.3.1 Initial files that are created

(1) Directories and files that the HiRDB administrator creates

The following table lists the directories and files that the HiRDB administrator creates.

Table 1–2: Organization of directories and files that the HiRDB administrator creates

File or directory name	Description
\$PDDIR/conf/pdsys	File for storing system common definitions
\$PDDIR/conf/pdutsys	File for storing unit control information definitions
\$PDDIR/conf/pdsvrc	File for storing server common definitions
\$PDDIR/conf/ <i>server-name</i>	File for storing each server definition
\$PDDIR/conf/pduapenv	Directory for storing UAP environment definitions
\$PDDIR/conf/chgconf	Directory for storing system reconfiguration definition files

(2) Directories and files that HiRDB creates

The following table lists the directories and files that HiRDB creates.

Table 1–3: Directories and files that HiRDB creates

File or directory name	Description
\$PDDIR/bin	Directory for storing HiRDB commands and utilities
\$PDDIR/lib	Directory for storing HiRDB's shared libraries and message text files
\$PDDIR/lib/sysconf	Directory for storing a file that is used to analyze HiRDB system definitions
\$PDDIR/lib/sysdef	
\$PDDIR/lib/sysdef_r	
\$PDDIR/lib/sysdef_v94	
\$PDDIR/lib/servers	Directory for storing HiRDB server and XDS executable files and libraries
\$PDDIR/lib/save	Directory for storing information when the <code>pdmemsv</code> command is used to share the libraries
\$PDDIR/lib/jvm	Directory for storing libraries that are required to execute the POSIX library HiRDB (created for 32-bit mode and 64-bit mode HP-UX 11.0)
\$PDDIR/lib/chinese	Directory for storing EUC Chinese character code parser libraries
\$PDDIR/lib/chinese-gb18030	Directory for storing Chinese character code parser libraries
\$PDDIR/lib/lang-c	Directory for storing single-byte character code parser libraries
\$PDDIR/lib/sjis	Directory for storing Shift-JIS kanji code parser libraries
\$PDDIR/lib/ujis	Directory for storing EUC Japanese kanji code parser libraries
\$PDDIR/lib/utf-8	Directory for storing Unicode (UTF-8) parser libraries
\$PDDIR/lib/utf-8_ivs	Directory for storing Unicode (IVS-supported UTF-8) parser libraries

File or directory name	Description
\$PDDIR/client/lib	Directory for storing HiRDB client's libraries
\$PDDIR/client/lib10_20	Directory for storing HP-UX 10.20 HiRDB client's libraries
\$PDDIR/client/utl	Directory for storing HiRDB client's commands and utilities
\$PDDIR/include	Directory for storing the header information that is used during UAP creation
\$PDDIR/spool	Directory for storing HiRDB work files
\$PDDIR/spool/save ^{#1}	Directory for storing saved core files and server process abnormal termination information file
\$PDDIR/spool/pdshmdump ^{#1}	Directory for storing shared memory dump files
\$PDDIR/spool/pdlckinf ^{#1}	Directory for storing deadlock time-out information files
\$PDDIR/spool/pdsysdump ^{#1}	Directory for storing simple dump files common to the system
\$PDDIR/spool/pdsdsdump ^{#1}	Directory for storing simple dump files for a single server
\$PDDIR/spool/pdfesdump ^{#1}	Directory for storing simple dump files for a front-end server
\$PDDIR/spool/pddicdump ^{#1}	Directory for storing simple dump files for a dictionary server
\$PDDIR/spool/pdbesdump ^{#1}	Directory for storing simple dump files for a back-end server
\$PDDIR/spool/pdstj1, pdstj2	Statistics log files
\$PDDIR/spool/pdlog1, pdlog2	Message log files
\$PDDIR/spool/pdjnlinf	Directory for storing system log information output files
\$PDDIR/spool/pdjnlinf/errinf	Directory for storing system log error information output files
\$PDDIR/spool/pdscdqi1, pdscdqi2	Files for storing HiRDB's internal schedule queue information
\$PDDIR/spool/oslmqid	File for storing message queue IDs
\$PDDIR/spool/oslsmid	File for storing semaphore IDs
\$PDDIR/spool/pdprcsts	prc status file
\$PDDIR/spool/.pdatmode	Status files for startup and termination
\$PDDIR/spool/.pdipcid	Files used for managing semaphore IDs
\$PDDIR/spool/.pdommenv	Files for storing shared memory information
\$PDDIR/spool/cmdlog/cmdlog1, cmdlog2	Files containing the historical information about the executed commands
\$PDDIR/spool/errlog/errlog1, errlog2	Files containing the historical information about internal HiRDB operation
\$PDDIR/spool/olkfifs	Directory for storing pipe files for use with thread locking ^{#7}
\$PDDIR/spool/olkrsfs	Directory for storing pipe files for use with thread suspend/resume ^{#8}
\$PDDIR/spool/oslcnt1	File for managing the number of pipe files
\$PDDIR/spool/cnctusrinf	File for storing the connected user information if there is any user still connected to HiRDB during the execution of normal termination or a planned termination command
\$PDDIR/spool/cnctusrdtl	File for storing the execution result of <code>pdls -d act</code> , <code>pdls -d prc</code> , and <code>pdls -d trn</code> commands if there is any user still connected to HiRDB during the execution of normal termination or a planned termination command

1. Overview of HiRDB System Construction

File or directory name	Description
\$PDDIR/spool/pdsqldump ^{#1}	Directory for storing access path information files
\$PDDIR/spool/pdtrninf	Directory for outputting transaction information file when using realtime SAN replication
\$PDDIR/spool/pdprf	Directory for outputting PRF trace files
\$PDDIR/spool/pduaperr	Directory for SQL error report files
\$PDDIR/spool/pdcwwrn	Directory for SQL runtime warning information files
\$PDDIR/spool/utlrpt	Directory for outputting processing performance information files
<i>any name</i> ^{#2}	RPC trace file
/dev/HiRDB/pth/ ^{#3}	Directory for storing communication information files
\$PDDIR/tmp ^{#4}	HiRDB's internal work directory
\$PDDIR/tmp/pdommenv	File for storing shared memory information
\$PDDIR/tmp/home/ <i>HiRDB-managed-ID-directory</i>	Current work directory
\$PDDIR/conf	Directory for storing HiRDB system definition files
\$PDDIR/conf/backconf	Directory for storing the pre-reconfiguration HiRDB system definition when executing the system reconfiguration command
\$PDDIR/conf/Inittab	/etc/inittab save directory
\$PDDIR/.dbenv	Directory for storing HiRDB database environment information files
\$PDCLTPATH/pdsq11.trc, pdsq12.trc ^{#5}	Files for storing trace information for SQL that is executed by a UAP
\$PDCLTPATH/pderr1.trc, pderr2.trc ^{#5}	Files for storing information about communication errors between a UAP and a server
\$PDDIR/plugin	Directory that integrates all HiRDB plug-in directories
\$PDDIR/plugin/.sys	HiRDB's internal work directory
\$PDDIR/plugin/lib	Directory for storing plug-in libraries
\$PDDIR/plugin/ <i>plug-in-name</i>	Plug-in directory
\$PDDIR/plugin/ <i>plug-in-name</i> /.sys	HiRDB's internal work directory
\$PDDIR/plugin/ <i>plug-in-name</i> /bin	Directory for storing plug-in commands
\$PDDIR/plugin/ <i>plug-in-name</i> /etc	Directory for storing the common files that are required by all plug-ins
\$PDDIR/plugin/ <i>plug-in-name</i> /conf	Directory for storing plug-in configuration files
\$PDDIR/jre ^{#6}	Java execution environment
\$PDDIR/renew	Directory used during updating to the HiRDB update version
\$PDDIR/renew_bak	Directory for backing up the operating HiRDB when updating to the HiRDB update version
\$PDDIR/.pdlogprgid	File for managing the syslog program ID
■ For HP-UX /sbin/init.d/ /sbin/rc1.d/	Directory for storing script files that run when the operating system starts up or shuts down. File name:

File or directory name	Description
<ul style="list-style-type: none"> ■ For Solaris /etc/init.d/ /etc/rc0.d/ /etc/rc1.d/ ■ For Linux /etc/init.d/ /etc/rc0.d/ /etc/rc1.d/ /etc/rc6.d/ 	<ul style="list-style-type: none"> ■ For HP-UX or Solaris HiRDB single server configuration is <code>HiRDB_S</code> or <code>K090HiRDB_S</code> HiRDB parallel server configuration is <code>HiRDB_P</code> or <code>K090HiRDB_P</code> ■ For Linux HiRDB single server configuration is <code>HiRDB_S</code>, <code>K09HiRDB_S</code>, or <code>S91HiRDB_S</code> HiRDB parallel server configuration is <code>HiRDB_P</code>, <code>K09HiRDB_P</code>, or <code>S91HiRDB_P</code>
<code>\$PDDIR/pdistup</code>	Directory containing the simple setup tool
<code>\$PDDIR/spool/tmp</code>	Directory for storing temporary work files

#1: Because HiRDB uses this directory to output troubleshooting information, it may keep increasing in size. You should use the `pdcsPOOL` command periodically to delete the contents.

You use the operands listed below for periodic deletion of troubleshooting information. For details about these operands, see the manual *HiRDB Version 9 System Definition*:

- `pd_spool_cleanup_interval`
- `pd_spool_cleanup_interval_level`
- `pd_spool_cleanup`
- `pd_spool_cleanup_level`

#2: To specify the filename, use the `pd_rpc_trace_name` operand.

#3: The files in this directory can be deleted when all HiRDB servers in this server machine have stopped.

#4: This directory is used internally by HiRDB. Do not create directories or files in this directory. Do not specify this directory for use by HiRDB to create files (for example, for the `pd_rpc_trace_name` operand). This directory is deleted and re-created each time the unit starts.

#5: Two copies of this file are output to the directory specified with `PDCLTPATH`. If `PDCLTPATH` is omitted, the files are output to the current directory used to start the UAP (in the case of a UAP started from OpenTP1, `$DCDIR/tmp/home/directory-with-server-name-xx`).

The names for the files to be created depends on whether X/Open-compliance API (`TX_` function) was used. When the `TX_` function is used, the files are created with the following names:

- `pdsqLxxxx-1.trc`, `pdsqLxxxx-2.trc`
- `pderrxxxx-1.trc`, `pderrxxxx-2.trc`

Legend:

xxxx: Process ID during UAP execution

Be aware that it is possible for as many files to be output as there are server processes during UAP execution, since the process ID serves as the file name.

#6: This directory is created when the version is earlier than 07-03. When version 07-03 or later is used, this directory is not created because JRE is not bundled with the package.

#7: The number of pipe files created in this directory is the *value of the `pd_max_server_process` operand* $\times 2 + 100$.

#8: Shown below are the formulas for approximating the number of pipe files to be created in this directory:

- HiRDB single server configuration
Value of `pd_max_server_process` + 127 + value of `pd_max_users` $\times 4$
- HiRDB parallel server configuration
*Value of `pd_max_server_process` + 127 + *a**

α : Use one of the following formulas, as applicable:

Front-end server: $\text{value of } pd_max_users \times 2$

Dictionary server: $\text{value of } pd_max_dic_process \times 35$

Back-end server: $\text{value of } pd_max_bes_process \times 35$

Include in the total the appropriate value for the servers in the unit.

1.3.2 Files that consistently increase in size

Files that consistently increase in size when HiRDB is used are listed in the table below by information type.

In the table below, note that an asterisk (*) can be any alphanumeric character. Standard path names are given for files or directory names. These might vary by system.

The earliest version that supports a given file is listed as the supporting version. For example, if the supporting version is the initial version, then all subsequent versions also support that file.

The legend for the table below is as follows:

Y: The maximum size can be limited using options.

N: The maximum size cannot be limited.

S: HiRDB single server configuration

P: HiRDB parallel server configuration

DK: HiRDB/Developer's Kit

RT: HiRDB/Run Time

(1) Simple dump

No.	File or directory name	Description	Approx. file size	File count	Can the max. size be limited?	Outputting software	Supporting version
1	\$PDDIR/spool/pdfesdump/*	<p>This is a simple dump file for a front-end server. It is generated when there is a pdfes process segmentation error or when an abort occurs.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the pdcspool command.</p> <p>The unit will shut down if there is insufficient space in \$PDDIR.</p>	Several MB (undetermined)	<p>Directory: The system loops through three generations of directories: \$PDDIR/spool/pdfesdump, pdfesdump1, and pdfesdump2</p> <p>File: Unlimited within directory</p>	N	P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can the max. size be limited?	Outputting software	Supporting version
2	\$PDDIR/spool/pddicdump/*	<p>This is a simple dump file for a dictionary server. It is generated when there is a pddic process segmentation error, an abort occurs or the dictionary RDAREA is closed.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the pdcspool command.</p> <p>The unit will shut down if there is insufficient space in \$PDDIR.</p>	Several MB (undetermined)	<p>Directory:</p> <p>The system loops through three generations of directories: \$PDDIR/spool/pddicdump, pddicdump1, pddicdump2</p> <p>File:</p> <p>Unlimited within directory</p>	N	P	Initial
3	\$PDDIR/spool/pdbesdump/*	<p>This is a simple dump file for a back-end server. It is generated when there is a pdbes process segmentation error, an abort occurs or the user RDAREA is closed.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the pdcspool command.</p> <p>The unit will shut down if there is insufficient space in \$PDDIR.</p>	Several MB (undetermined)	<p>Directory:</p> <p>The system loops through three generations of directories: \$PDDIR/spool/pdbesdump, pdbesdump1, and pdbesdump2</p> <p>File:</p> <p>Unlimited within directory</p>	N	P	Initial

1. Overview of HiRDB System Construction

No.	File or directory name	Description	Approx. file size	File count	Can the max. size be limited?	Outputting software	Supporting version
4	\$PDDIR/spool/pdsdsdump/*	<p>This is a simple dump file for a single server. It is generated when there is a pdsds process segmentation error, an abort occurs or the user RDAREA is closed.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the pdcspool command.</p> <p>The unit will shut down if there is insufficient space in \$PDDIR.</p>	Several MB (undetermined)	<p>Directory: The system loops through three generations of directories: \$PDDIR/spool/pdsdsdump, pdsdsdump1, and pdsdsdump2</p> <p>File: Unlimited within directory</p>	N	S	Initial
5	\$PDDIR/spool/pdsysdump/*	<p>This is a simple dump file shared by the system. It is generated when a process that controls the HiRDB system terminates abnormally.</p> <p>When the server process is XDS, it is generated in the following cases:</p> <ul style="list-style-type: none"> XDS is forcibly terminated by a pdxdsstop -f command. Group system switchover is performed by HA monitor's monswap command. HiRDB is forcibly terminated by a pdstop -f command. XDS terminates abnormally. <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval 	Several MB to several dozen MB (undetermined)	<p>Directory: The system loops through three generations of directories: \$PDDIR/spool/pdsysdump, pdsysdump1, and pdsysdump2</p> <p>File: Unlimited within directory</p>	N	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can the max. size be limited?	Outputting software	Supporting version
		<ul style="list-style-type: none"> pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the <code>pdcsPOOL</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>					

(2) Error information

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	<code>\$PDDIR/spool/pdlckinf/*</code>	<p>This is a deadlock timeout information file. It is generated when a lock error occurs.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the <code>pdcsPOOL</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>	Several KB	Unlimited	N	S, P	Initial
2	<code>\$PDDIR/spool/pdtrninf/*</code>	<p>This is a transaction information file. It is generated using realtime SAN replication.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval 	Several KB	Unlimited	N	S, P	07-01

1. Overview of HiRDB System Construction

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
		<ul style="list-style-type: none"> pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the <code>pdcspool</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>					
3	<code>\$PDDIR/spool/pdjnlinf/*</code>	<p>This is the status information file for the facility for monitoring the free space remaining for system log files. When the system log file free area falls to a value below the warning value, a file is generated.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the <code>pdcspool</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>	Approx. 2,729 to 3,521 bytes	One file per server	N	S, P	07-00
4	<code>\$PDDIR/spool/pdjnlinf/errinf/*</code>	<p>This is a file for outputting system log error information. It is generated at reruns and when a system log read error occurs.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level 	The max. value is the value specified in the <code>pd_log_max_data_size</code> operand.	Number of log generations (finite number)	N	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
		<ul style="list-style-type: none"> pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the <code>pdcspace</code> command. The file is also deleted when the <code>pdstop</code> command is executed (with no options, the <code>-i</code> option, or the <code>dbdestroy</code> option).</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDBDIR</code>.</p>					
5	/tmp/pdskipsq1_server-name_process-ID	<p>This is the reflection SQL skip information file when the <code>pdorend</code> command is executed. It is generated when an SQL skip target error occurs in reflection processing when the <code>pdorend</code> command is executed.</p> <p>Use the OS's <code>rm</code> command (or some other method) to manually delete the file.</p>	$((400 + ((1,500 \times \text{mapping-key-structure-column-count}) \times 2) [+ 50 + (\sum (50 + \uparrow(\text{updated-member-count}^{\#1} \div 9) \uparrow \times 80))^{\#2}) \times \text{skipped-SQL-query-count}) \times 1.2$	Number of servers in RDAREA on which the <code>pdorend</code> command was run (for each process ID of the command and in server units)	N	S, P	07-01
6	\$PDCLTPATH/pderr1.trc, pderr2.trc	This is a client error information file. It is generated when a UAP is executed. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Value specified in client environment variable <code>PDUAPERLOG</code> (default is 4,096 bytes)	Two files, between which output is cycled.	Y	S, P, DK, RT	Initial
7	\$PDCLTPATH/pderrxxx-1.trc, pderrxxx-2.trc (xxxx: Process ID when UAP executes)	This is a client error information file (when an API (TX_ function) that conforms to X/Open is used). It is generated when a UAP is executed. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Value specified in client environment variable <code>PDUAPERLOG</code> (default is 4,096 bytes)	Two files created for each UAP process ID, between which output is cycled.	Y	S, P, DK, RT	07-01

#1

Details about the number of updated members are as follows.

- UPDATE SET (specifies members): Number of members to be updated
- UPDATE ADD: 1
- UPDATE DELETE: Number of members to be deleted

#2

When `mvcelmwarn=ignore` is specified in the control statement and a repetition column exists, the values in square brackets are added. Σ : Adds amounts for the number of repetition columns.

(3) Tuning information

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	\$PDDIR/spool/pdsqldump/*	<p>This is an access path information file. When 1 or greater is specified in the client environment variable PDVWOPTMODE, this file is generated when a SQL statement is executed (for each preprocessing).</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level <p>To delete manually: Execute the pdcspool command.</p> <p>The unit will shut down if there is insufficient space in \$PDDIR.</p>	(Several KB to several hundred KB) × SQL statement count within transaction	For each transaction	N	S, P	04-03

(4) Troubleshooting information

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	\$PDDIR/spool/save/abcode.*	<p>This is server process abnormal termination information (primarily abort codes). It is generated when a server process terminates abnormally.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> pd_spool_cleanup_interval pd_spool_cleanup_interval_level pd_spool_cleanup pd_spool_cleanup_level 	<p>In 32-bit mode: 40 bytes</p> <p>In 64-bit mode: 72 bytes</p>	Unlimited	N	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
		<p>To delete manually: Execute the <code>pdcspool</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>					
2	<code>\$PDDIR/spool/save/*</code>	<p>This is server process abnormal termination information (code file). It is generated when a server process terminates abnormally.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> <code>pd_spool_cleanup_interval</code> <code>pd_spool_cleanup_interval_level</code> <code>pd_spool_cleanup</code> <code>pd_spool_cleanup_level</code> <p>To delete manually: Execute the <code>pdcspool</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>	<p>Several MB to several dozen MB</p> <p>Note: In the following cases, the attached shared memory dump is also included in the core, so it can grow to several GB.</p> <ul style="list-style-type: none"> In AIX, the <code>EXTSHM</code> environment variable is not specified. The Linux version is earlier than 5.2 and the HiRDB version is earlier than 08-04. 	<p>Three generations of looped files named <code>shutdown-server-name[1-3]</code> (If the name of the server is unknown, this might be a number rather than a name.)</p>	N	S, P	Initial
3	<code>\$PDDIR/spool/save/*.deb</code>	<p>This is server process abnormal termination information. It is generated when a server process terminates abnormally.</p> <p>When the server process is XDS, it is generated in the following cases:</p> <ul style="list-style-type: none"> XDS is forcibly terminated by a <code>pdxdstop -f</code> command. Group system switchover is performed by HA monitor's <code>monswap</code> command. HiRDB is forcibly terminated by a <code>pdstop -f</code> command. XDS terminates abnormally. 	<p>Several MB to several dozen MB</p>	<p>Three generations of looped files named <code>shutdown-server-name[1-3].deb</code> (If the name of the server is unknown, this might be a number rather than a name.)</p>	N	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
		<p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> • <code>pd_spool_cleanup_interval</code> • <code>pd_spool_cleanup_interval_level</code> • <code>pd_spool_cleanup</code> • <code>pd_spool_cleanup_level</code> <p>To delete manually: Execute the <code>pdcspool</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>					
4	<code>\$PDDIR/spool/save/*.ext.deb</code>	<p>This is server process abnormal termination information. It is generated when a server process terminates abnormally, and a backtrace is output.</p> <p>When the server process is XDS, it is generated in the following cases:</p> <ul style="list-style-type: none"> • XDS is forcibly terminated by a <code>pdxdstop -f</code> command. • Group system switchover is performed by HA monitor's <code>monswap</code> command. • HiRDB is forcibly terminated by a <code>pdstop -f</code> command. • XDS terminates abnormally. <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> • <code>pd_spool_cleanup_interval</code> • <code>pd_spool_cleanup_interval_level</code> • <code>pd_spool_cleanup</code> • <code>pd_spool_cleanup_level</code> 	Several dozen KB	<p>Three generations of looped files named <i>shutdown-server-name</i>[1-3].deb[#]</p> <p>(If the name of the server is unknown, this might be a number rather than a name.)</p>	N	S, P	09-03

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
		<p>To delete manually: Execute the <code>pdcsPOOL</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>					
5	<code>\$PDDIR/spool/save/command-name*.txt</code>	<p>This is troubleshooting information for the <code>pdload</code>, <code>pdrcorg</code>, <code>pdreclaim</code>, and <code>pdpgbfon</code> commands. It is generated when a command times out.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> <code>pd_spool_cleanup_interval</code> <code>pd_spool_cleanup_interval_level</code> <code>pd_spool_cleanup</code> <code>pd_spool_cleanup_level</code> <p>To delete manually: Execute the <code>pdcsPOOL</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>	Several KB to several dozen KB (Varies with total number of HiRDB processes executed at that time and number of resources waiting for lock release)	One file for each command process ID	N	S, P	06-02
6	<code>\$PDDIR/pdshmdump/*</code>	<p>This is a shared memory dump file. It is generated when a server process or unit terminates abnormally.</p> <p>To delete automatically: Specify the following operands:</p> <ul style="list-style-type: none"> <code>pd_spool_cleanup_interval</code> <code>pd_spool_cleanup_interval_level</code> <code>pd_spool_cleanup</code> <code>pd_spool_cleanup_level</code> <p>To delete manually: Execute the <code>pdcsPOOL</code> command.</p> <p>The unit will shut down if there is insufficient space in <code>\$PDDIR</code>.</p>	Several hundred MB to several GB (about the same size as the shared memory segment whose attribute of the process that uses shared memory displayed with the <code>pdls -d mem</code> command is MANAGER)	The system loops through two generations of files: <code>shmdump</code> and <code>shmdump.old</code>	N	S, P	Initial

1. Overview of HiRDB System Construction

#

When the `pdcancel -d` command is used to forcibly terminate a server process, an instance of the following file is generated for each process that is canceled.

`$PDDIR/spool/save/server-name.process-ID.cancel.ext.deb`

(5) Special files for communication

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	<code>/dev/HiRDB/pt h/*</code>	<p>This is a special file for communication. It is generated at HiRDB startup, HiRDB server startup, command execution, and UAP connection.</p> <p>It is automatically deleted when the HiRDB server process terminates normally.</p> <p>Use the OS's <code>rm</code> command (or some other method) to manually delete the file (it can be deleted with all the HiRDB servers of the server machine stopped).</p> <p>The unit cannot start if there is insufficient space in <code>\$PDDIR</code>.</p>	0	Maximum of 196,605	N	S, P	Initial

(6) Temporary files for index creation

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	<code>directory-specified-by-pd_plugin_ixmk_dir/index-name.RDAREA-name</code> (if directory specified in <code>pd_plugin_ixmk_dir</code>)	<p>This is a plug-in index information file. It is generated when an SQL statement is executed.</p> <p>It is automatically deleted when batch creation of the plug-in index terminates normally or when re-creation of the plug-in index is executed.</p> <p>Use the OS's <code>rm</code> command (or some other method) to manually delete the file.</p>	See the <i>Notes</i> section in <i>Delayed batch creation of a plug-in index in the HiRDB Version 9 System Operation Guide</i> .	$plugin\ count \times index\ storage\ RDAREA-count$ (number of updated RDAREAs)	N	S, P	05-06

(7) Temporary work files for operation commands

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	<code>\$PDDIR/tmp/CMr*</code>	This is a temporary operation command file. It is generated when an	$MAX(1,024\ bytes \times global-buffer-count, 512)$	One file for each command process ID	Y	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
		operation command is executed. It is automatically deleted when operation command execution terminates.	bytes \times <i>RDAREA-count</i>)				
2	\$PDDIR/tmp/CMS*	This is a temporary operation command file. It is generated when an operation command is executed. It is automatically deleted when operation command execution terminates.	128 bytes \times <i>RDAREA-count</i>	One file for each command process ID	Y	S, P	Initial
3	\$PDDIR/tmp/CMb*	This is a differential information file for the <i>pdbufls</i> command. It is generated when the <i>pdbufls</i> command (specifying <i>-k sts</i> or <i>no</i> option) is executed. It is automatically deleted when HiRDB starts. Use the OS's <i>rm</i> command (or some other method) to manually delete the file while the <i>pdbufls</i> command is not executing.	16 bytes + 124 bytes \times <i>global-buffer-count</i>	One file for each command process ID	Y	S, P	Initial
4	\$PDDIR/tmp/pdcmd*	This is the operation command results file. It is generated when an operation command is executed. It is automatically deleted when command execution terminates. This file is only used by a HiRDB parallel server configuration when the dictionary server and system manager are different nodes.	MAX(1,024 bytes \times <i>global buffer count</i> , 512 bytes \times <i>RDAREA count</i>)	One file for each command process ID	Y	P	Initial
5	/tmp/pddefrev.exp.*(the <i>pd_tmp_directory</i> operand is ignored)	This is a temporary <i>pddefrev</i> command work file. It is generated when the <i>pddefrev</i> command is executed. It is automatically deleted when <i>pddefrev</i> command execution terminates. Use the OS's <i>rm</i> command (or some other method) to manually delete the file.	<i>number of resources to export</i> \times 70 bytes	One file for each command process ID	N	S, P	04-01
6	\$PDDIR/spool/tmp/*	This is a temporary file for an operation command. It is generated when an operation command is executed. It is deleted	A maximum of 1,024 bytes	One file for each command process ID	N	S, P	09-04

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
		automatically when the operation command execution terminates.					

(8) Utility results files

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	\$TMPDIR/ pdcp1*, / tmp/ pdcp1*, <i>directory-specified-by-the-p-option</i> / pdcp1* (In 08-02 or later, this file is stored in the pd_tmp_directory directory if the pd_tmp_directory operand is specified.)	This is a pdcopy results file. It is generated when pdcopy is executed. Use the OS's rm command (or some other method) to manually delete the file.	3,500 (bytes) × <i>RDAREA count</i>	One file with a different file name each time for pdcopy with no -p option	Y	S, P	Initial
2	\$TMPDIR/ pdrs1*, <i>directory-specified-by-the-w-option</i> / pdrs1* (In 08-02 or later, this file is stored in the pd_tmp_directory directory if the pd_tmp_directory operand is specified.)	This is a temporary pdrstr results file (temporary file for storing rollback logs when recovering log specifications). It is generated when pdrstr is executed. It is deleted automatically when rollback terminates. Use the OS's rm command (or some other method) to manually delete the file.	Depends on rollback log size	Number of servers associated with the pdrstr target RDAREA	N	S, P	Initial
3	\$TMPDIR/ pdrs2*, / tmp/ pdrs2*, <i>directory-specified-by-the-p-option</i> / pdrs2* (In 08-02 or	This is a pdrstr results file. It is generated when pdrstr is executed. Use the OS's rm command (or some other method) to manually delete the file.	3,500 (bytes) × <i>RDAREA count</i>	One file with a different file name each time for pdrstr with no -p option	Y	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
	later, this file is stored in the <code>pd_tmp_directory</code> directory if the <code>pd_tmp_directory</code> operand is specified.)						
4	<code>\$TMPDIR/pdrs4</code> , <code>/tmp/pdrs4</code> (In 08-02 or later, this file is stored in the <code>pd_tmp_directory</code> directory if the <code>pd_tmp_directory</code> operand is specified.)	This is a temporary <code>pdrstr</code> results file (temporary file for outputting messages to console). It is generated when <code>pdrstr</code> is executed. It is deleted automatically when <code>pdrstr</code> execution terminates. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	256 bytes	One (command execution node only)	Y	S, P	Initial
5	<code>/tmp/REPORT*</code> (in 08-02 or later, this file is stored under the <code>pd_tmp_directory</code> directory if the <code>pd_tmp_directory</code> operand is specified)	This is a <code>pdrbal</code> results file. It is generated when <code>pdrbal</code> is executed. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	See 19.1.9 <i>File sizes required for execution of the rebalancing utility (pdrbal)</i> .	One file with a different file name each time no report control statement is specified	N	S, P	06-00
6	<code>/tmp/CONSTCK-REPORT*</code> (in 08-02 or later, this file is stored under the <code>pd_tmp_directory</code> directory if the <code>pd_tmp_directory</code> operand is specified)	This is a <code>pdconstck</code> results file. It is generated when <code>pdconstck</code> starts executing. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	See 19.1.10 <i>File sizes required for execution of the integrity check utility (pdconstck)</i> .	One file for each command process ID	N	S, P	07-03

(9) Temporary work file for utilities

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	\$TMPDIR/ pdcop3*, / tmp/ pdcop3* (In 08-02 or later, this file is stored in the pd_tmp_d irectory directory if the pd_tmp_d irectory operand is specified.)	This is a temporary pdcopy results file (for outputting messages to the console). It is generated when pdcopy is executed. It is automatically deleted when pdcopy execution terminates. Use the OS's rm command (or some other method) to manually delete the file.	256 bytes	One (command execution node only)	Y	S, P	Initial
2	\$TMPDIR/ pdcop4*, / tmp/ pdcop4* (In 08-02 or later, this file is stored in the pd_tmp_d irectory directory if the pd_tmp_d irectory operand is specified.)	This is a temporary pdcopy results file (for storing messages). It is generated when pdcopy execution starts. It is automatically deleted when pdcopy execution terminates. Use the OS's rm command (or some other method) to manually delete the file.	256 bytes \times <i>pdcopy target</i> <i>RDAREA count</i>	Number of servers associated with the target RDAREA of <i>pdcopy</i> + 2	Y	S, P	Initial
3	\$TMPDIR/ pdrstr5*, / tmp/ pdrstr5* (In 08-02 or later, this file is stored in the pd_tmp_d irectory directory if the pd_tmp_d irectory operand is specified.)	This is a temporary pdrstr results file (for storing messages). It is generated when pdrstr execution starts. It is automatically deleted when pdrstr execution terminates. Use the OS's rm command (or some other method) to manually delete the file.	256 bytes \times <i>pdrstr target</i> <i>RDAREA count</i>	Number of servers associated with the target <i>RDAREA</i> + 2. However, add an additional 1 when the number of target servers is two or more, or when you are inputting a log that exists somewhere other than the recovery target unit.	Y	S, P	Initial
4	/tmp/ ERROR* (in 08-02 or later, this file is stored under the pd_tmp_d irectory directory if	This is a results file for pdload input data storage information (error information file). It is generated when pdload is executed. Use the OS's rm command (or some other method) to manually delete the file.	See 19.1.1 <i>File sizes required for the execution of the database load utility (pdload).</i>	If no error operand is specified in the source statement, one file, with a different file name each time	N	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
	the <code>pd_tmp_directory</code> operand is specified)						
5	<code>/usr/tmp</code> <code>/*, /var/tmp/*, /tmp/*</code> (in 08-02 or later, this file is stored under the <code>pd_tmp_directory</code> if the <code>pd_tmp_directory</code> operand is specified)	This is a temporary file for creating the <code>pdload</code> error information file. It is generated when <code>pdload</code> is executed. Use the OS's <code>rm</code> command (or some other method) to manually delete the file. This file is only used with a HiRDB parallel server configuration.	See 19.1.1 <i>File sizes required for the execution of the database load utility (pdload)</i> .	Number of servers that store the table to be loaded with data	N	S, P	Initial
6	<code>/tmp/LOBMID*</code> (in 08-02 or later, this file is stored under the <code>pd_tmp_directory</code> if the <code>pd_tmp_directory</code> operand is specified)	This is a work file for <code>pdload</code> BLOB column loads. It is generated when <code>pdload</code> execution starts. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	See 19.1.1 <i>File sizes required for the execution of the database load utility (pdload)</i> .	When something other than <code>d</code> is specified for the <code>-k</code> option, the <code>lobmid</code> statement is omitted, and a table that has a BLOB column is loaded with data, one file with a different file name each time	N	S, P	03-00
7	<code>/tmp/INDEX*</code> , <code>idxwork-control-statement-specification-directory/INDEX*</code> (in 08-02 or later, this file is stored under the <code>pd_tmp_directory</code> if the <code>pd_tmp_directory</code> operand is specified)	This is an index information file for <code>pdload</code> , <code>pdrorg</code> , and <code>pdrbal</code> . It is generated when <code>pdload</code> , <code>pdrorg</code> , or <code>pdrbal</code> executes. It is deleted automatically when index loading finishes. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	See the following: <i>19.1.1 File sizes required for the execution of the database load utility (pdload)</i> , <i>19.1.2 File sizes required for the execution of the database reorganization utility (pdrorg)</i> , <i>19.1.9 File sizes required for the execution of the rebalancing utility (pdrbal)</i>	When index batch creation mode is selected, the number of files is <code>index count × index storage RDAREA count</code> , with a different file name each time.	N	S, P	Initial
8	<code>/tmp/rs*</code> , <code>sort-control-statement-specification -</code>	This is a sorting work file for <code>pdload</code> , <code>pdrorg</code> , and <code>pdrbal</code> . It is generated	See the following: <i>19.1.1 File sizes required for the</i>	When index batch creation mode is selected, the number of files	N	S, P	Initial

1. Overview of HiRDB System Construction

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
	<i>directory/rs*</i> (in 08-02 or later, this file is stored under the <i>pd_tmp_directory</i> directory if the <i>pd_tmp_directory</i> operand is specified)	when <i>pdload</i> , <i>pdrorg</i> , or <i>pdrbal</i> executes. It is automatically deleted when index loading finishes or when the process terminates. Use the OS's <i>rm</i> command (or some other method) to manually delete the file.	<i>execution of the database load utility (pdload)</i> , <i>19.1.2 File sizes required for the execution of the database reorganization utility (pdrorg)</i> , <i>19.1.9 File sizes required for execution of the rebalancing utility (pdrbal)</i> .	equals the index storage server count, with a different file name each time.			
9	<i>/tmp/*.dbst.data</i> , <i>/tmp/*.dbst.msg</i> (in 08-02 or later, this file is stored under the <i>pd_tmp_directory</i> directory if the <i>pd_tmp_directory</i> operand is specified)	This is a <i>pddbst</i> command temporary file. It is generated when <i>pddbst</i> execution starts. It is deleted automatically when <i>pddbst</i> execution terminates. Use the OS's <i>rm</i> command (or some other method) to manually delete the file.	See <i>19.1.4 File sizes required for the execution of the database condition analysis utility (pddbst)</i> .	One file for each process ID of the command.	N	S, P	Initial
10	<i>/tmp/*.syi</i> , <i>*.syo</i> , <i>*.uai</i> , <i>*.uao</i> , <i>*.sqi</i> , <i>*.sqo</i> , <i>*.pci</i> , <i>*.pco</i> , <i>*.soi</i> , <i>*.soo</i> , <i>*.doi</i> , <i>*.doo</i> , <i>*.bui</i> , <i>*.buo</i> , <i>*.fii</i> , <i>*.fio</i> , <i>*.dfi</i> , <i>*.dfo</i> , <i>*.ixi</i> , <i>*.ixo</i> , <i>*.isi</i> , <i>*.iso</i> , <i>*.cni</i> , <i>*.cno</i> , <i>*.ghi</i> , <i>*.gho</i> , <i>*.shi</i> , <i>*.sho</i> ,	This is a temporary <i>pdstedit</i> command work file. It is generated when <i>pdstedit</i> execution starts. It is deleted automatically when <i>pdstedit</i> execution terminates. Use the OS's <i>rm</i> command (or some other method) to manually delete the file.	See <i>19.1.3 File sizes required for the execution of the statistics analysis utility (pdstedit)</i> .	One file for each process ID of the command and for each information item subject to analysis.	N	S, P	Initial

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
	*.obi, *.obo, *.fsi, *.fso, *.hbi, *.hbo, <i>directory-specified-by--w-option / ***</i> (in 08-02 or later, this file is stored under the pd_tmp_directory directory if the pd_tmp_directory operand is specified)						

(10) Trace information

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	\$PDCLTPATH/ pdsq11.trc, pdsq12.trc	This is SQL trace information. It is generated at SQL statement execution. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by client environment variable <code>PDSQLTRACE</code>	Two files, between which output is cycled.	Y	S, P, DK, RT	Initial
2	\$PDCLTPATH/ pdsq1xxxx-1.trc, pdsq1xxxx-2.trc xxxx: Process ID when UAP executes	This is SQL trace information (when using X/Open-compliant API (TX function)). It is generated at SQL statement execution. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by client environment variable <code>PDSQLTRACE</code>	Two files created for each UAP process ID, between which output is cycled.	Y	S, P, DK, RT	07-01
3	\$PDCLTPATH/ pdjsqlxxx xxxx_pppp p_1.trc, pdjsqlxxx xxxx_pppp p_2.trc xxxxxxx: Connected server name	This is SQL trace information (when using Type 4 JDBC driver). It is generated at SQL statement execution. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by client environment variable <code>PDSQLTRACE</code>	Two files each created for connected server name and client reception port number, between which output is cycled.	Y	S, P, DK, RT	08-00

1. Overview of HiRDB System Construction

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
	<i>ppppp</i> : Client reception port number						
4	\$PDCLTPATH/ pdjsqlxxx xxxx_ <i>pppp</i> <i>p_1</i> .trc, pdjsqlxxx xxxx_ <i>pppp</i> <i>p_2</i> .trc xxxxxxx: Connected server name <i>ppppp</i> : Client reception port number	This is a dynamic SQL trace (when Type 4 JDBC driver is used). It is generated at SQL statement execution. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by the <code>pdtrcmgr -s</code> command	Two files each created for connected server name and client reception port number, between which output is cycled.	Y	S, P, DK, RT	08-00
5	\$PDJDBFILEDIR/ pdexc1.trc, pdexc2.trc	This is an exception trace log. It is generated when an exception occurs within a Type 4 JDBC driver. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	See <i>Exception Trace Log</i> in the <i>HiRDB Version 9 UAP Development Guide</i> .	Two files, between which output is cycled.	Y	S, P, DK, RT	08-00
6	Specified by user (in <code>setLogWriter</code> method of <code>DriverManager</code> class or <code>setLogWriter</code> method of <code>DataSource</code> interface)	This is a JDBC interface method trace. It is generated when an exception occurs within a Type 4 JDBC driver or when a <code>Connection.close</code> method is called. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified in system property <code>HiRDB_for_Java__FileSize</code> or client environment variable <code>PDJDBFILESIZE</code> . If not specified, $\uparrow 180 \times n \times m \div 1,024 \uparrow$ kilobytes <i>n</i> : Specification in user property <code>TRC_NO</code> , which is specified at connection (default is 500) <i>m</i> : Number of exceptions from connection to disconnection + 1	One file	Y	S, P, DK, RT	08-00
7	\$PDTRCPATH/ pdCHHMM SSmmm_XX X_1.trc, pdCHHMM SSmmm_XX X_2.trc	This is a dynamic SQL trace. It is generated at SQL statement execution. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by <code>pdtrcmgr -s</code> command	Two files created for each connection, between which output is cycled.	Y	S, P, DK, RT	06-00

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
	<i>HHMMSSmm</i> : CONNECT time <i>XXX</i> : CONNECT number						
8	\$PDCLTPATH/ pdrncnt1.trc, pdrncnt2.trc	This is a reconnection trace. It is generated when a connection is made automatically using the automatic connection function. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by client environment variable PDRCTRACE	Two files, between which output is cycled.	Y	S, P, DK, RT	07-01

(11) Statistical information

No.	File or directory name	Description	Approx. file size	File count	Can max. size be limited?	Outputting software	Supporting version
1	\$PDREPPATH/ pd <i>HHMMSSmmmm</i> _XXX_1.trc, pd <i>HHMMSSmmmm</i> _XXX_2.trc <i>HHMMSSmm</i> : CONNECT time <i>XXX</i> : CONNECT number	This is a UAP statistical report. It is generated at SQL statement execution. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by client environment variable PDSQLTRACE	Two files created for each connection, between which output is cycled.	Y	S, P, DK, RT	06-00
2	\$PDCLTPATH/ pdjsqlxxx.xxx_1.trc, pdjsqlxxx.xxx_2.trc xxxxxxx: Connected server name ppppp: Client reception port number	This is a UAP statistical report (when a Type 4 JDBC driver is used). It is generated at SQL statement execution. Use the OS's <code>rm</code> command (or some other method) to manually delete the file.	Specified by client environment variable PDSQLTRACE	Two files each created for the connected server name and client reception port number, between which output is cycled.	Y	S, P, DK, RT	08-00

1.4 Upgrading HiRDB

This section describes the procedure for upgrading HiRDB.

Upgrading refers to installing a later version or revision of HiRDB (for example, when *VV* or *RR* in the *VV-RR-ZZ* format HiRDB version number increases).

When upgrading a HiRDB single server configuration, also upgrade the corresponding utility special unit. Make sure that the HiRDB single server configuration and the corresponding utility special unit have matching versions of HiRDB.

When upgrading a HiRDB parallel server configuration, upgrade all units constituting the HiRDB parallel server configuration so that they have the same version of HiRDB.

! Important note

- When upgrading HiRDB, *do not uninstall the existing HiRDB*. Install the new version of HiRDB over the existing version.
- There are some notes concerning upgrading that apply when the security audit facility is used. For details about these notes, see the *HiRDB Version 9 System Operation Guide*.
- There are notes about upgrading to version 07-03 or later by using the Java stored procedures and functions. For details about the notes, see *1.4.5 Using Java stored procedures and functions*.

1.4.1 Before upgrading

Before upgrading, make sure that the steps described below are taken. Note that if you are sharing libraries in a multi-HiRDB configuration, you must cancel the library sharing and apply the following actions to all directories:

- *1.4.1(3) Checking to see whether HiRDB is online.*
- *1.4.1(5) Checking the HiRDB status.*
- *1.4.1(6) Releasing library sharing.*

(1) Checking for available space

Use the database condition analysis utility (pddbst) to see if there is enough space in the data dictionary RDAREAs. If there is not enough space, allocate sufficient space using one of the following methods:

- Reorganize the dictionary table using the database reorganization utility (pdrrorg).
- Extend the data dictionary RDAREAs using the database structure modification utility (pdmod).

This space checking is required only when you are upgrading your HiRDB; it is not necessary when you are updating to the HiRDB update version.

For the utility execution method, see the manual *HiRDB Version 9 Command Reference*.

Free space required for upgrading

Check the free space requirements indicated in the following table for the version of HiRDB being used prior to the upgrade. If there is not enough space, an insufficient space error may occur when you start HiRDB or execute the pdvtrup command after upgrading.

Table 1–4: Free space required for upgrading

Dictionary tables stored in data dictionary RDAREA	Number of free segments required in data dictionary RDAREA				
	Upgrading from 07-00 or later	Upgrading from 06-00 or later	Upgrading from 05-02 or later	Upgrading from 03-00 or later	Upgrading from 02-05 or earlier
SQL_TABLES table	1	3	3	3	3
SQL_COLUMNS table	$4 + \uparrow 5 \div S \uparrow$	$5 + \uparrow 5 \div S \uparrow$	$5 + \uparrow 15 \div S \uparrow$	$5 + \uparrow 20 \div S \uparrow$	$5 + 1 \uparrow 25 \div S \uparrow$

Dictionary tables stored in data dictionary RDAREA	Number of free segments required in data dictionary RDAREA				
	Upgrading from 07-00 or later	Upgrading from 06-00 or later	Upgrading from 05-02 or later	Upgrading from 03-00 or later	Upgrading from 02-05 or earlier
SQL_INDEXES table	1	1	4	4	4
SQL_TABLE_PRIVILEGES table	1	1	3	3	3
SQL_INDEX_COLINF table	1	1	3	3	3
SQL_VIEW_TABLE_USAGE table	2	3	4	4	4
SQL_VIEWS table	$2 + \uparrow 5 \div S \uparrow$	$3 + \uparrow 5 \div S \uparrow$	$3 + \uparrow 15 \div S \uparrow$	$3 + \uparrow 20 \div S \uparrow$	$3 + \uparrow 25 \div S \uparrow$
SQL_VIEW_DEF table	$2 + \uparrow 10 \div S \uparrow$	$3 + \uparrow 10 \div S \uparrow$	$2 + \uparrow 40 \div S \uparrow$	$2 + \uparrow 65 \div S \uparrow$	$2 + \uparrow 70 \div S \uparrow$
SQL_DIV_COLUMN table ^{#1}	--	--	--	1	--
SQL_ROUTINES table ^{#2}	--	--	$4 + \uparrow 30 \div S \uparrow$	$4 + \uparrow 30 \div S \uparrow$	--
SQL_ROUTINE_PARAMS table ^{#2}	--	--	$2 + \uparrow 20 \div S \uparrow$	$2 + \uparrow 20 \div S \uparrow$	--

Legend:

--: Not applicable

S: Segment size of data dictionary RDAREA for storing the corresponding table

#1: The table is not needed if no LOB column or data dictionary LOB RDAREA is defined.

#2: The table is not needed if no stored procedure is defined.

(2) Backing up system RDAREAs

Use the database copy utility (`pdcopy`) to back up the following RDAREAs:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs
- RDAREAs containing audit trail tables (applicable when the security audit facility is used)

Note that if you downgrade the version after you have successfully upgraded (for example, if you upgrade for test purposes and then downgrade to restore the original operations), you must first back up all RDAREAs.

To back up the RDAREAs, use the following procedure:

1. Use the `pdstop` command to terminate HiRDB normally.
2. Use the `pdstart -r` command to start HiRDB.
3. Use the database copy utility (`pdcopy`) to back up the RDAREAs. In this case, specify the reference/update-possible mode (`-M x` specified). For details about the backup procedure, see the manuals *HiRDB Version 9 System Operation Guide* or *HiRDB Version 9 Command Reference*.

(3) Checking to see whether HiRDB is online

Use the `pdls` command to see if all units are displayed as ACTIVE. If they are ACTIVE, use the `pdstop` command to terminate them normally.

(4) Terminating HiRDB normally

Before upgrading, terminate HiRDB normally. In the case of a HiRDB parallel server configuration, terminate HiRDB from the machine that contains the system manager. If HiRDB has already been terminated, check the following information to determine whether HiRDB has terminated normally:

- Message log file or syslog file

If HiRDB has not terminated normally, enter the `pdstart` command to start HiRDB and then enter the `pdstop` command to terminate it normally.

(5) Checking the HiRDB status

To check the status of a unit whose HiRDB is to be upgraded, execute the `pdls -d ust` command.

When the termination status is 4 (unit status is `STARTING` or `STOPPING`):

HiRDB is engaged in start or termination processing. Re-execute the `pdls -d ust` command after the start or termination processing is completed.

When the termination status is 8 (unit status is `PAUSE`):

Restart of the process service has been cancelled due to an error. Check the `KFPS00715-E` message and the message that has been output to the syslog file before this message, eliminate the cause of the error, and then execute the `pdrpause` command. After that, re-start the unit, and then terminate it normally with the `pdstop` command.

(6) Releasing library sharing

If you are sharing libraries with multi-HiRDB, use the `pdmemsv -d` command to release library sharing. After upgrading HiRDB, use the `pdmemsv` command to share libraries again.

(7) Stopping commands, utilities, applications, and HiRDB-linked programs

Terminate commands, utilities, and applications. Before upgrading, also stop any linked program that accesses HiRDB, such as HiRDB Datareplicator, HiRDB Dataextractor, or JP1/PFM. If any such item is running, deletion of executable files and shared libraries fails, resulting in an upgrading error.

(8) Checking the default values of operands in the HiRDB system definition

With each version and revision, HiRDB revises the default values that are assumed when you omit HiRDB system definition operands. HiRDB version 09-50 and later offers two modes for the default behavior when operands are omitted: *recommended mode*, which assumes the recommended default values, and *compatibility mode*, which uses the default values for a specific version of HiRDB. Normally, to build a more secure system, we suggest that you apply the recommended mode, which greatly reduces the number of operands that must be specified. In version 09-50 and later, to use the default values for a specific version, apply the compatibility mode, and do not specify the operand `pd_sysdef_default_option`.

- If you are upgrading from a version earlier than 09-50

For details about the advantages and disadvantages of changing the default values, see *Changes to HiRDB system definitions by version and revision* in the manual *HiRDB Version 9 System Definition*. After checking the default values, if you decide to focus on compatibility with the previous version, you can apply the compatibility mode to make the default values the same as in the previous version. However, in this case all of the operands will have the previous version's default values, so we recommend specifying the recommended values on a per-operand basis.

- If you are upgrading from a version earlier than 07-00, or if you are specifying the operand `pd_sysdef_default_option`

In addition to the changes mentioned above in *If you are upgrading from a version earlier than 09-50*, see the description about upgrading from a version earlier than 09-50 in *Operands whose default value depends on the version and operands that are no longer needed* in the manual *HiRDB Version 9 System Definitions*. Then, after taking note of the relevant advantages, consider specifying the recommended values on a per-operand basis.

(9) Checking other default values

With each version and revision, HiRDB revises the default values that are assumed when you omit utility options and SQL options.

If you are upgrading from a version earlier than 09-50, check the locations listed below regarding the advantages and disadvantages of changing the default values. After checking the default values, if you decide to focus on compatibility with the previous version, you can apply the compatibility mode to make the default values the same as in the previous version. However, in this case all of the operands will have the previous version's default values, so we recommend specifying the recommended values on a per-operand basis.

- *Options and Control Statements Whose Default Value Is Changed or Whose Specification Is No Longer Needed After Upgrading* in the manual *HiRDB Version 9 Command Reference*
- *Changes to the default interpretation of SQL syntax by version and revision* in the manual *HiRDB Version 9 SQL Reference*

(10) Checking the memory requirements

The memory requirements may increase when you upgrade the HiRDB version. You should check the HiRDB memory requirements in *Chapter 15. Storage Requirements for HiRDB*.

(11) Checking the size of status files

When you upgrade HiRDB, the size of the HiRDB status files may increase. You should check the size of the HiRDB status files as described in *17.3 Determining the size of status files*.

(12) Checking the size of synchronization point dump files

When you upgrade HiRDB, the size of the HiRDB synchronization point dump files may increase. You should check the size of the HiRDB synchronization point dump files as described in *17.2 Determining the size of synchronization point dump files*.

(13) Checking the operating system parameters

The operating system parameter values (or kernel parameter values) may change when the HiRDB version is updated. For details about how to estimate the operating system parameter values, see *20. Specifying OS Parameters*.

Note

When the following conditions are satisfied, the values of the operating system parameters will have to be increased, so the values will need to be re-evaluated:

- When upgrading from version 06-00 or earlier to version 06-01 or later
- When 31 or more system log file groups are created

(14) Checking the total number of records in the system log file

If you are upgrading, check the total number of records in the system log files in overwrite enabled status. If the following condition is not satisfied, upgrading may fail:

When upgrading from HiRDB Version 4.0 or earlier

$Total\ number\ of\ records^{\#} > 5,754,880 / system\ log\ file\ record\ length$

When upgrading from HiRDB Version 5.0 or earlier

$Total\ number\ of\ records^{\#} > 4,239,360 / system\ log\ file\ record\ length$

When upgrading from HiRDB Version 6 or earlier

$Total\ number\ of\ records^{\#} > 3,215,360 / system\ log\ file\ record\ length$

When upgrading from HiRDB Version 7 or earlier

$Total\ number\ of\ records^{\#} > 1,413,120 / system\ log\ file\ record\ length$

For a HiRDB parallel server configuration, check the total number of records in the system log files (overwrite enabled status) at the dictionary server.

#: Use one of the following methods to check the total number of records in the system log files:

- The total number of -n option values in the `pdloginit` command is the total number of records.
- Execute the `pdlogls -d sys -s server-name -e` command. The sum of the numbers of records (hexadecimal) that are output at the top of the `Recode-count` in the execution result is the total number of records.

(15) Backing up the files in the HiRDB directory

To be prepared for the possibility of upgrading errors, back up the files under the HiRDB directory (`$PDDIR/conf`). Delete the backup copy after you have checked the operation of the new version. For details about how to back up the HiRDB directory, see 2.3.1(4) *Making a backup copy of the HiRDB directory*.

(16) Upgrading option program products

If option program products were used with HiRDB before upgrading, those option program products must be upgraded to the same version as HiRDB. For details about option program products, see 2.2.2 *Installing option program product*.

(17) Checking the added reserved words

With the extension of SQL, the reserved words listed below have been added to each version of HiRDB. An SQL statement containing a reserved word that is not enclosed in double-quotation marks might result in a syntax error after upgrading.

HiRDB version	Added reserved words
06-00	GET_JAVA_STORED_ROUTINE_SOURCE, IS_USER_CONTAINED_IN_HDS_GROUP
06-01	None
06-02	BIT_AND_TEST
07-00	CONDITION, EXIT, HANDLER, TIMESTAMP_FORMAT, VARCHAR_FORMAT
07-01	FREE, LOCATOR
07-02	None
07-03	OVER
08-00	ENCRYPT
08-01	None
08-02	COUNT_FLOAT, SQLCODE_OF_LAST_CONDITION, SQLERRM_OF_LAST_CONDITION, XML, XMLAGG, XMLEXISTS,

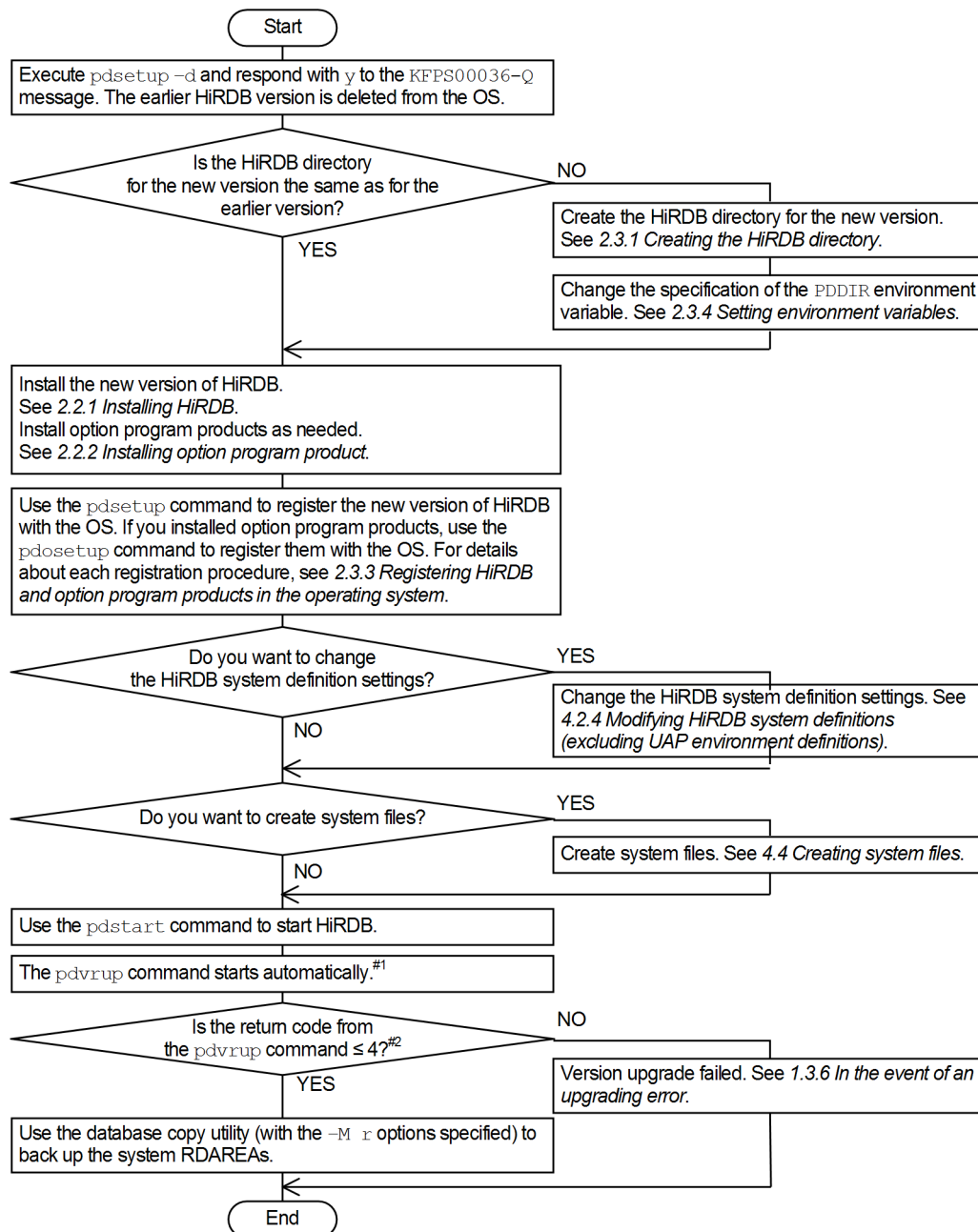
HiRDB version	Added reserved words
	XMLQUERY, XMLSERIALIZE
08-03	None
08-04	XMLPARSE
08-05	None
09-00	None
09-01	None
09-02	None
09-03	COMPRESSED

If any of the added reserved words has been used without being enclosed in double-quotation marks, take appropriate action by referencing *What to do if a name conflicts with an SQL reserved word* in the manual *HiRDB Version 9 SQL Reference*.

1.4.2 Replacing an existing version with the new version

This subsection describes the procedure for replacing an existing version of HiRDB with a more recent version.

Figure 1–2: Procedure for replacing an existing version with the new version



#1

- If you specify `pd_auto_vrup=N` in the system common definition, the `pdvrup` command does not start automatically. In this case, if the `KFPS05203-Q` message (`pdvrup` command entry request message) is issued, the HiRDB administrator must enter the `pdvrup` command.
- When you are updating to the HiRDB update version, the `pdvrup` command does not start automatically. Proceed to the next step.

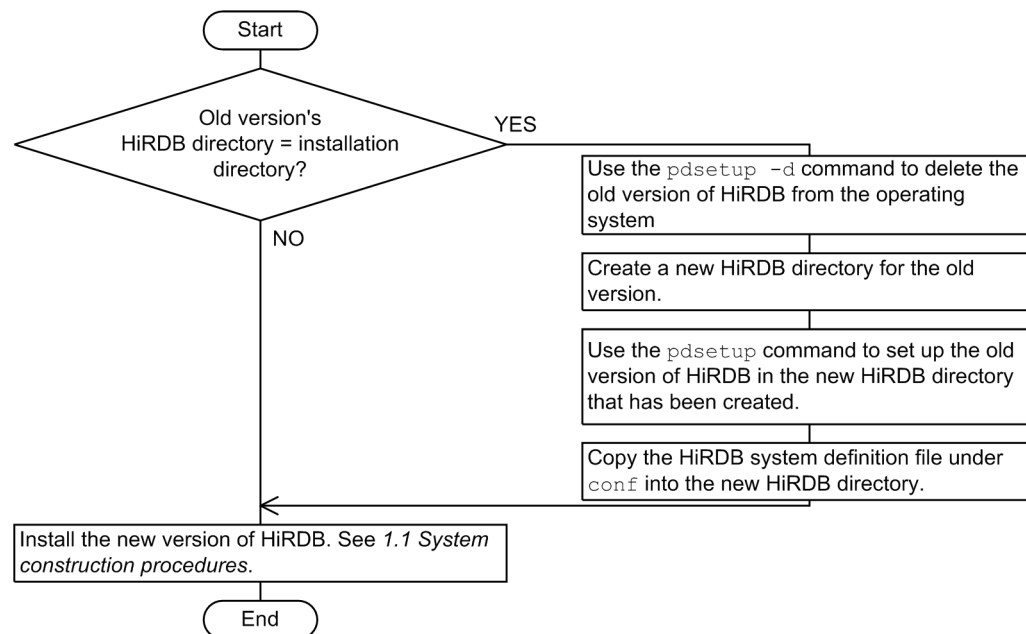
#2

For the execution result of the `pdvrup` command, check the `KFPX24404-I` message in the message log file or syslog file.

1.4.3 Installing a new version while retaining the old version

The following figure illustrates the procedure for installing a more recent version of HiRDB while retaining the previous version (to create a multi-HiRDB configuration with both earlier and more recent versions).

Figure 1–3: Installing a new version while retaining the old version (to create multi-HiRDB)



1.4.4 Upgrading the HiRDB plug-ins

When upgrading HiRDB, you also need to upgrade your plug-ins. For the HiRDB version required for plug-ins and the plug-in upgrading procedure, see an applicable plug-in manual and 5.2 *Upgrading plug-ins*.

1.4.5 Using Java stored procedures and functions

With HP-UX and Solaris HiRDB, JRE (the Java execution environment), which is required to use Java stored procedures and functions, is bundled with version 07-02 and earlier but not with version 07-03 or later. When you update HiRDB version 07-02 or earlier to version 07-03 or later, the version of JRE that was installed with HiRDB version 07-02 or earlier is deleted in the following situations:

When	What is deleted
<ul style="list-style-type: none"> When uninstalling HiRDB version 07-02 or earlier When overwrite installing HiRDB version 07-03 or later 	JRE of the HiRDB installation directory
When executing the <code>pdsetup -d</code> command and sending <code>y</code> in response to the <code>KFPS00036-Q</code> message	JRE of the HiRDB directory

Therefore, if you perform the installation described in 1.4.3 *Installing a new version while retaining the old version*, you can use the JRE of the HiRDB directory of the old version (07-02 or earlier). However, if you install a new version (07-03 or later), the JRE of the HiRDB installation directory is deleted.

Note the following tips when upgrading to HiRDB version 07-03 or later:

- To use Java stored procedures and functions, you must obtain JRE beforehand (you can obtain it from the website of the platform vendor). For details about the specific JRE that is required to use Java stored procedures and functions, see the *HiRDB Version 9 System Operation Guide*.

- To use Java stored procedures and functions, you must specify in the `pd_java_runtimepath` operand the root directory of JRE to be used. Also, as required, specify in the `pd_java_libpath` operand the directory where the Java virtual machine is stored. For details about the `pd_java_runtimepath` and `pd_java_libpath` operands, see the manual *HiRDB Version 9 System Definition*.
- To use the JRE installed with the old version after upgrading to a new version, back up JRE to a directory other than either the installation directory or the HiRDB operation directory before deleting JRE. By specifying in the `pd_java_runtimepath` operand the directory in which JRE is backed up, you can use the backed up version of JRE to run Java stored procedures and functions.

1.4.6 In the event of an upgrading error

This section discusses the actions that you should take if any of the following events occurs:

- The `pdvrup` command results in a return code of 5 or greater.
- During HiRDB startup, the `KFPX24404-I` message is issued with a return code of 5 or greater.

In this case, check the message that is output along with this message and take an appropriate action.

(1) When you do not need to terminate HiRDB

Eliminate the cause of the error and reenter the `pdvrup` command.

(2) When you need to terminate HiRDB

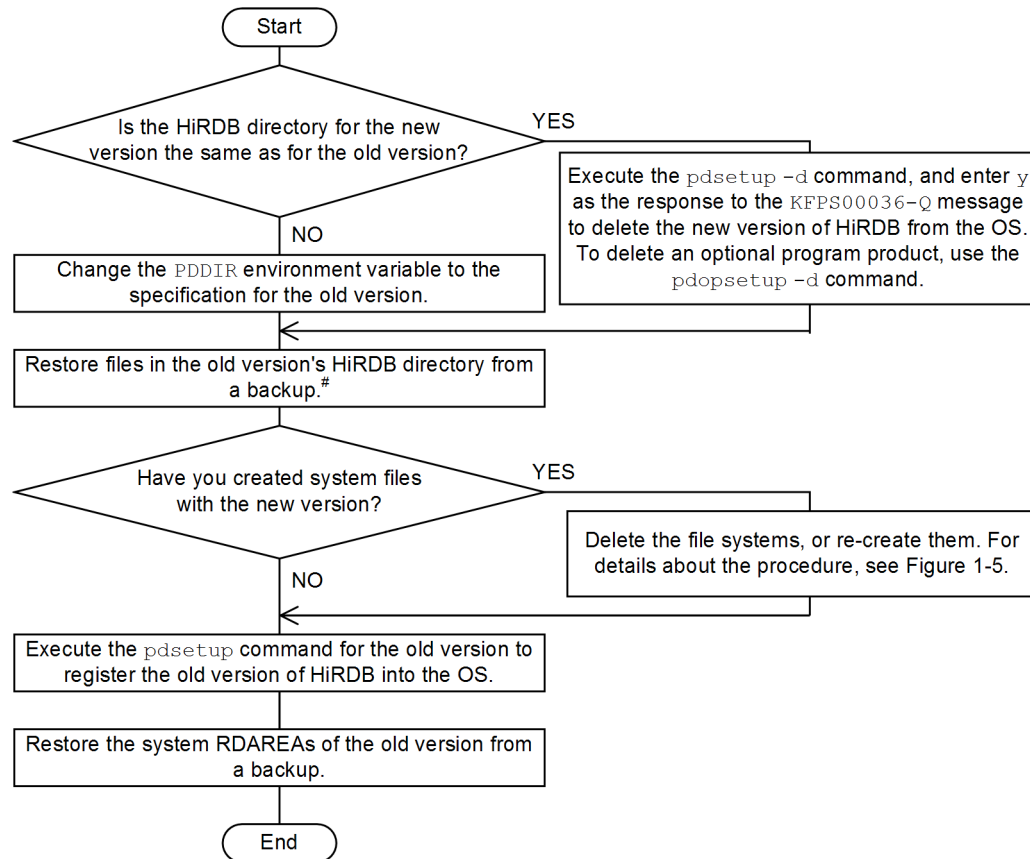
If you need to terminate HiRDB to correct the error, terminate HiRDB once. After eliminating the cause of the error, use the `pdstart` command to start HiRDB. When HiRDB starts, the `pdvrup` command entry request message (`KFPS05203-Q`) is issued, so that you can reenter the `pdvrup` command.

(3) When you need to restore the old version of HiRDB

To correct the error, you may need to restore an earlier version of HiRDB, depending on the error. For example, if insufficient data dictionary RDAREA space is the cause of the upgrading error, you need to restore the previous version of HiRDB and use the database structure modification utility (`pdmod`). In this case, restore the previous version of HiRDB, eliminate the cause of the error, and then upgrade HiRDB again.

The following figure illustrates the procedure for restoring the previous version of HiRDB.

Figure 1-4: Procedure to restore HiRDB to the previous version (when version upgrade fails)

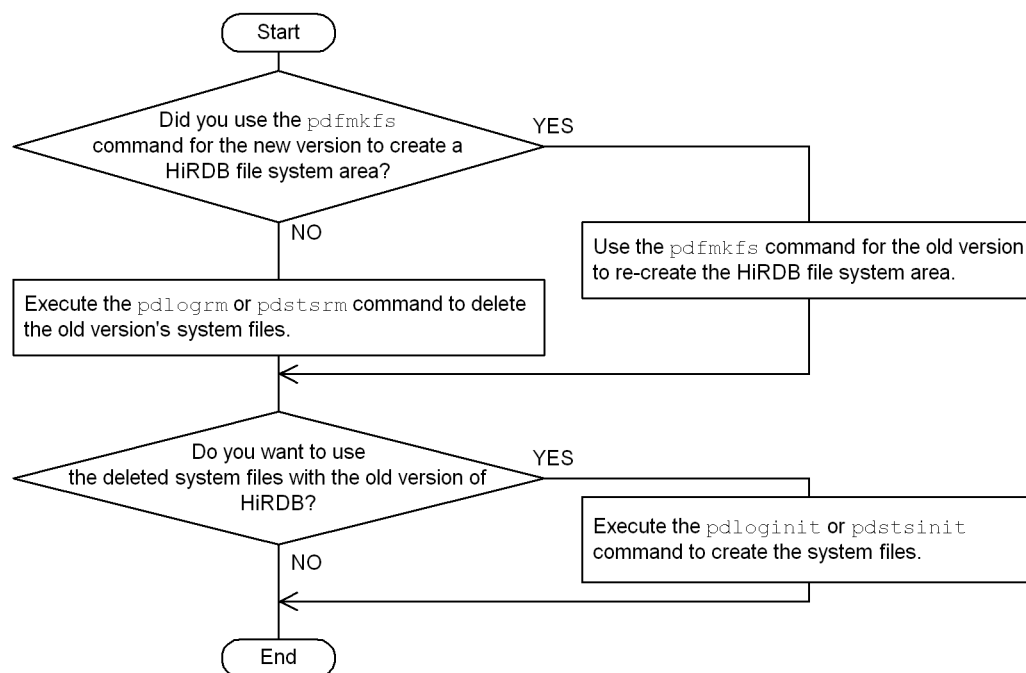


#

If there is no backup copy but the system definition files under `$PDDIR/conf` have been retained, you can use those files as they are. If there is no backup copy and the files under `$PDDIR/conf` have also been deleted, you must re-create the system definition files.

The following figure illustrates the procedure for deleting and re-creating the system files created by the later version.

Figure 1–5: Procedure for deleting and re-creating the system files created by the new version



1.4.7 Restoring an earlier version of HiRDB

To downgrade the version after a successful HiRDB upgrade (for example, to go back to the original operation after upgrading HiRDB as a trial), you must re-create HiRDB using the previous version.

(1) Prerequisites

To be able to re-create the previous version after an upgrade, you must have backups available from before you upgraded of all the RDAREAs and system definition files. Make these backups using the database copy utility (pdcopy).

(2) Procedure for re-creating HiRDB

The procedure for re-creating HiRDB is basically the same as for initially installing HiRDB and configuring the environment. The procedure for re-creating HiRDB using the previous version is described below.

1. Stop HiRDB (pdstop command).
2. Delete the more recent version of HiRDB from the OS (execute the pdsetup -d command, answering Y).
3. Uninstall the more recent version of HiRDB.^{#1}
4. Uninstall the previous version.
5. Register the previous version of HiRDB in the OS (execute the pdsetup command).
6. Set the previous-version environment.

Create the HiRDB file system area (execute the pdfmkfs command).^{#2}

In the HiRDB file system area, create the system files (system log file, sync point dump file, and status file) (execute the pdloginit and pdstsinit commands).

If the system definitions were changed in the more recent version of HiRDB, they must be changed to the previous-version system definitions file acquired prior to the upgrade (returning to the system definitions of the previous version of HiRDB).

7. Use the pdstart -r command to start HiRDB in order to start up the database recovery utility (pdrstr).

8. Using the database recovery utility (`pdrstr`), restore the database from the backup file made prior to the upgrade (restore all RDAREAs). Do not use the unload log file that includes the log that was updated by HiRDB after the upgrade.
9. Stop HiRDB (`pdstop` command).
10. Start HiRDB (`pdstart` command).

#1

If you are running multiple HiRDB systems in a multi-HiRDB configuration, uninstalling HiRDB will delete all HiRDB systems on the server. For this reason, do an overwrite install of the previous version of HiRDB over the more recent version of HiRDB without uninstalling the later version of HiRDB.

#2

If you are not executing the `pdfmkfs` command with the more recent version of HiRDB, there is no need to create a HiRDB file system area with the previous version.

(3) Notes

The following are notes concerning downgrading to a previous version.

1. If you use the system switchover function

Restore both the running system and the standby system to the previous version.
2. If an option program product has been set up

If after you restore the previous version, the option program product is no longer the required version, change to an option program product that is compatible with the previous version of HiRDB.
3. If you use realtime SAN replication

Restore the main site and remote sites to the previous version simultaneously using the *reflect all data* method. Also, if you use the *log-only synchronous* method, application of the system log must be executed after step 9 of the procedure for re-creating HiRDB.
4. If you use the security monitoring function

When you create a system file in step 5 of the procedure for re-creating HiRDB, also create an audit trail file.
5. If you use the HiRDB Datareplicator data link functionality
 - HiRDB Datareplicator must also be restored to the previous version in between steps 1 and 4 of the procedure for re-creating HiRDB.
 - HiRDB and the HiRDB Datareplicator link environment must be reinitialized after step 9 of the procedure for re-creating HiRDB.

1.5 Updating to HiRDB update version

A HiRDB update version has the same version number and revision number as the active HiRDB, such as 07-02, followed by a code, *-mn*; for example, 07-02-01, where the underlined part is the *-mn* code. For versions earlier than 07-02, *m* can be a /, an alphabetic character (excluding I, O, and P to T) or a number, and *n* is an alphabetic character from A to Z. For versions 07-02 and later, *m* and *n* are both numbers.

You can switch to the HiRDB update version while the existing version of HiRDB is running.

If you exit HiRDB to switch to the update version, apply the default behavior for omitted operands that is already in effect.

1.5.1 Updating HiRDB

There are the two ways to update HiRDB:

- Use the installer.
- Obtain and use the update patch from the Web.

The two methods are explained below.

(1) Using the installer to update HiRDB

You can use the installer to update HiRDB. There are two ways to use the installer to update HiRDB.

(a) Updating after terminating HiRDB

Switches performed after terminating HiRDB are performed according to the same method described in 1.4.2 *Replacing an existing version with the new version*. Check the following references before you update to the HiRDB update version:

- The description and notes of 1.4 *Upgrading HiRDB*
- The operating procedures of 1.4.2 *Replacing an existing version with the new version*[#]
#: The operations in the procedure that involve the `pdvrup` command are not needed.

Also perform the following before you update to the HiRDB update version.

1. Check whether HiRDB is in online status.
Use the `pdls` command to check whether all units are shown as `ACTIVE`. If they are, proceed to step 2. If HiRDB has already terminated, proceed to step 3.
2. Terminate HiRDB.
Stop HiRDB in any shutdown mode. For a HiRDB parallel server configuration, stop all HiRDB units.
3. Check the status of the HiRDB unit.
Execute the `pdls -d ust` command to check the status of the HiRDB unit. If the termination status is 4 (unit status is `STARTING` or `STOPPING`), HiRDB is in the process of starting or stopping. After processing has finished, execute the `pdls -d ust` command again.
For a HiRDB parallel server configuration, execute the `pdls -d ust` command on all HiRDB units and check the HiRDB unit status.
4. Release library sharing.
Perform (6) *Releasing library sharing* described in 1.4.1 *Before upgrading*.
5. Stop commands, utilities, applications, and HiRDB-linked programs.
Follow the instructions in (7) *Stopping commands, utilities, applications, and HiRDB-linked programs* in the subsection 1.4.1 *Before upgrading*.

(b) Updating with HiRDB operating

See 1.5.2 *Prerequisites* and following sections.

(2) Using the update patch to update HiRDB

When the version and revision numbers are the same as the active HiRDB, you can use the update patch to update HiRDB. The patch is available from the Web. There are two ways to use the patch to update HiRDB:

(a) Quit HiRDB and update

For details of how to use this method, read `RELEASE.TXT` or `RELEASE.EUC`, which is included with the update patch.

(b) Update while running HiRDB

See *1.5.2 Prerequisites* and following sections.

1.5.2 Prerequisites

The following prerequisites must be satisfied in order to update while HiRDB is operating:

- Version, HiRDB server type, addressing mode
The following items must be the same for the HiRDB update version and the operating HiRDB; these items can be checked with the `pdadmvr` command:
 - Version number and revision number
 - HiRDB server type (HiRDB single server configuration or HiRDB parallel server configuration)
 - Addressing mode (32-bit mode or 64-bit mode)
- Operating system parameters
Estimates of the operating system parameters that will be required by HiRDB after the switch must be within the bounds of the currently set kernel parameters.
- HiRDB directory
The installation directory must not be the same as the HiRDB directory.
- Library sharing
In the case of multi-HiRDBs, the libraries must not be shared.
- Free disk space
There must be sufficient free disk space in the HiRDB directory to store both the currently operating HiRDB and the HiRDB update version. See the release notes for the free space required for the HiRDB update version.
- HiRDB client
A HiRDB client that is working online must be running on a HiRDB server that is not being updated. Such a HiRDB client that is operating on a HiRDB server that is to be updated must be terminated and the online job must be terminated.
- Client library
The HiRDB/Developer's Kit and HiRDB/Run Time that are being used by a HiRDB client that is working online must be version 07-00 or later. If an earlier version is being used, the connection of the HiRDB client will be released during updating.
- Application of the automatic reconnect facility
A HiRDB client that is connected to HiRDB must use the automatic reconnect facility (`PDAUTORECONNECT=YES`). If the automatic reconnect facility is not being used, the connection of the HiRDB client will be released during updating. For details about the automatic reconnect facility, see the *HiRDB Version 9 UAP Development Guide*.

1.5.3 Update procedure

To update to the HiRDB update version:

1. Install the HiRDB update version.

The HiRDB update version requires overwrite installation. For a HiRDB parallel server configuration, overwrite installation of the HiRDB update version must be performed for each unit. For details about the installation procedure, see *Chapter 2. Installation*.

2. Copy the HiRDB update version to the update directory.

To copy the HiRDB update version stored in the installation directory to the update directory in the HiRDB directory (\$PDDIR/renew), execute the following directory command of the operating HiRDB: *HiRDB-update-version-installation-directory/bin/pdprgcopy*. For a HiRDB parallel server configuration, execute the *pdprgcopy* command at the unit where the system manager is installed.

3. Check that HiRDB is online.

Use the *pdls* command to check that all units display as *ACTIVE*.

4. Update to the HiRDB update version.

To update HiRDB, execute the *HiRDB-update-version-installation-directory/bin/pdprgrenew* directory command of the operating HiRDB. When this command is executed, the operating HiRDB is saved to the backup directory (\$PDDIR/renew_bak), at which point the operating HiRDB can be updated with the HiRDB update version copied in step 2 to the update directory. For a HiRDB parallel server configuration, execute the *pdprgrenew* command on the unit where the system manager is installed.

1.5.4 Update procedure when the system switchover facility is used

When the system switchover facility is used, an operating HiRDB can be updated in the following cases:

- Standby system switchover
This can be done only when the system that is running is operating as the primary system. When the system that is running is operating as the secondary system, execute the command after performing system switchover.
- Standby-less system switchover
This can be done only when all normal BESs are operating. Switchover cannot occur when alternating.

The following describes how to update to the HiRDB update version when the system switchover facility is used.

- When operating in server mode

Standby system switchover

1. If the secondary system is running, perform system switchover in such a manner that the primary system will be running after the switchover.
2. Terminate the standby system.
3. Use the running system to update to the HiRDB update version.
4. Use the *pdsetup -d* command to delete the HiRDB standby system from the OS (respond with *y* to the *KFPS00036-Q* message).
5. Perform overwrite installation of the HiRDB update version on the standby system.
6. Use the *pdsetup* command to register the HiRDB standby system with the OS.
7. Re-start the standby system terminated in step 1. On the standby HiRDB system, execute the *pdstart* command (for a HiRDB parallel server configuration, execute the *pdstart -q* command).

Standby-less system switchover (1:1)

1. If the alternate BES is running, switch back the system.
2. Remove the alternate portion from waiting status. For details about the procedure, see the *HiRDB Version 9 System Operation Guide*.
3. On the normal BES unit that is running, update to the HiRDB update version.
Because the alternate portion that was removed from waiting status in step 2 goes into waiting status automatically when the *pdprgrenew* command is executed, no other action is required.

Standby-less system switchover (effects distributed)

1. If the accepting unit's guest BES is running, switch back to the normal unit.
2. Release the acceptable status of all inactive guest BESs that belong to the HA group.

3. On the running normal unit, update to the HiRDB update version.

Because the guest BESs whose acceptable status was released in step 2 go into acceptable status automatically when the `pdprgnew` command is executed, no other action is required.

- Operating in monitor mode
 1. If the secondary system is running, perform switchover so that the primary system will be running.
 2. On the running system, update to the HiRDB update version.
 3. Use the `pdsetup -d` command to remove the HiRDB standby system from the OS (respond with `y` to the `KFPS00036-Q` message).
 4. On the standby system, perform overwrite installation of the HiRDB update version.
 5. Use the `pdsetup` command to register the HiRDB standby system with the OS.

1.5.5 Cautions

- When you cannot update to the HiRDB update version
Depending on the HiRDB operating status, it might not be possible to update to the HiRDB update version. For details, see the description of the `pdprgnew` command in the manual *HiRDB Version 9 Command Reference*.
- UAP response delay
While the `pdprgnew` command is executing, UAP response times will be delayed. Thus, we recommend that you execute this command when traffic in the system is relatively low.
- Definition reconfiguration
The memory requirements change when you update to the HiRDB update version, which necessitates reconfiguration of the system definition. Thus, the HiRDB system definition must be reconfigured in advance with the `pdchgconf` system reconfiguration command. For details about the system reconfiguration command, see the *HiRDB Version 9 System Operation Guide*.
- Execution of operation commands and utilities
You must not execute operation commands or utilities while the `pdprgnew` command is executing. Doing so may result in an error causing HiRDB to terminate or in a failure when HiRDB is updated.
- Use of the system switchover facility
You cannot use the system switchover facility while updating to the update version.
- Invalid holdable cursor
As the cursor cannot be maintained while updating to the update version, a UAP that uses the holdable cursor cannot be used just before or after updating. Doing so will result in an error at the UAP.
- Invalid `LOCK` statement of the `UNTIL DISCONNECT` setting
As the `UNTIL DISCONNECT` setting cannot maintain an exclusive lock while updating to the update version, the `LOCK` statement cannot use the `UNTIL DISCONNECT` setting just before or after updating. Doing so will result in an error at the UAP.

1.5.6 Operation considerations

This subsection provides important information about updating to the HiRDB update version.

- Global buffers allocated using the database structure modification utility (`pdmod`) become invalid. The global buffers must be reallocated after the update is completed.
- The count startup point of the `pd_spool_cleanup_interval` operand is reset at the time of the update.
- If the `pd_spool_cleanup` operand is set to `normal` or `force`, any troubleshooting information output from completion of the update is deleted.
- If the `pdstbegin` command or the `pdstend` command is to be used to set collection conditions for statistical information to values different from the settings of the `pd_statistics` operand or the `pdstbegin` operand, use the following method:

- If statistical information was collected with the `pdstbegin` command in an environment that was activated without setting the `pd_statistics` operand or the `pdstbegin` operand, statistical information will not be collected after the update. In such a case, the `pdstbegin` command will have to be re-executed.
- If statistical information collection was terminated by using the `pdstend` command in an environment that was activated where the `pd_statistics` operand was specified as A or Y or where the `pdstbegin` operand was specified, or if the `pdstbegin` command was executed to change the type of statistical information to be collected, the collection of statistical information after the update will be as specified in the system common definition. Therefore, the `pdstend` command and the `pdstbegin` command must be re-executed.
- Because the list used in any narrowed search will disappear, such a list must be re-created with the `ASSIGN LIST` statement after the update.
- Because the number of resident processes altered by the `pdchprc` command returns to the number of resident processes specified by the HiRDB system definition, the `pdchprc` command must be re-executed after the update.
- The system log file is replaced during updating to the HiRDB update version. Before updating, check that there are swappable system log files. If system log files are needed, perform the following:
 - If there are no swappable system log files
If there are files in unload wait status, unload them. If there are no files in unload wait status, use the `pdchgconf` system reconfiguration command to add swappable log files. For details about the system reconfiguration command, see the *HiRDB Version 9 System Operation Guide*.
If the update is performed when there are no swappable system log files, you must terminate HiRDB with the `Psjnf07` or `Psjn381` code once the `KFPS01256-E` message has been output. In such a case, prepare swappable files, and then use the `pdstart` command to start HiRDB.
 - If there is only one swappable system log file
It is possible to update to the HiRDB update version in such a case, although the `KFPS01224-I` message will be output during the update to indicate that there are no log files. After the update, prepare swappable system log files.
- The message log file is replaced during updating to the HiRDB update version. However, the `KFPS01910-I` message notifying you that the message log file has been replaced is not output. If you want to save the messages in the message log file, back up the file prior to updating.

1.5.7 Related product limitations and considerations

- Plug-ins
An operating HiRDB can be updated to the HiRDB update version even when plug-ins are used. However, the plug-ins cannot be updated.
- HiRDB Datareplicator linkage facility
Do not execute the `pdprgnew` command on HiRDB while the HiRDB Datareplicator is being used for data extraction. If you are updating to the HiRDB update version without having terminated online applications, the HiRDB Datareplicator that is engaged in extraction must be terminated. However, do not stop the HiRDB Datareplicator linkage (do not execute the `pdprplstop` command). If the HiRDB Datareplicator linkage is interrupted, the extracting database and the target database can lose integration.

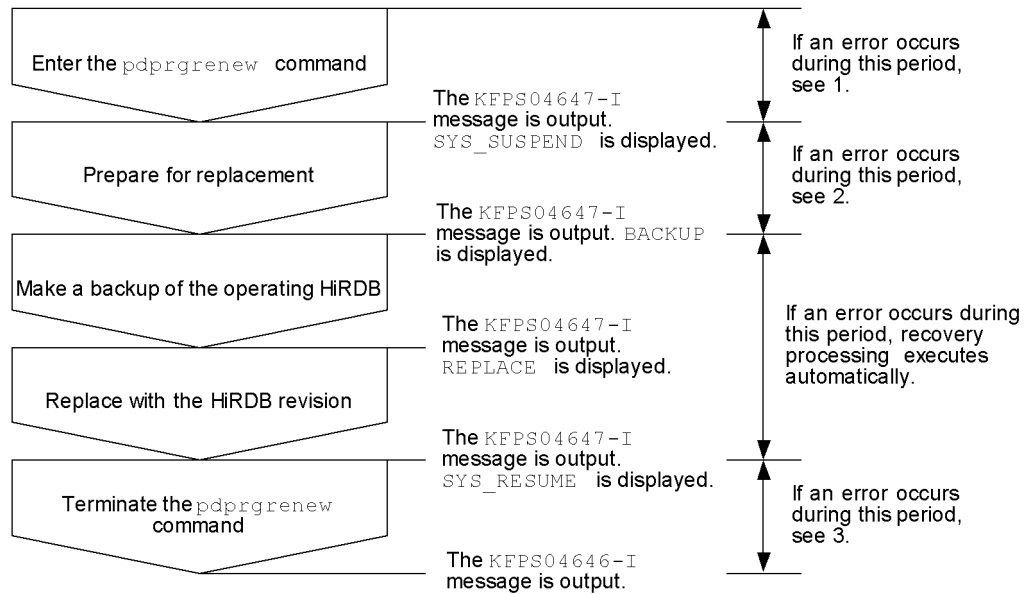
1.5.8 Operation when an error occurs during installation

(1) Error handling

If an error occurs during updating to the HiRDB update version, the `pdprgnew` command returns automatically to the pre-update HiRDB and starts HiRDB operation. If this command outputs the `KFPS04646-I` message with return code 12 and then terminates, it means that the operation to return to the pre-update HiRDB failed. In such a case, refer to the preparation error that is output or to the error messages and `KFPS04647-I` message output to the syslogfile for guidance on what to do next.

The following figure shows how to handle errors that might occur during updating to the HiRDB update version.

Figure 1–6: Handling errors during updating to the HiRDB update version



1. Correct whatever caused the error in the `pdprg Renew` command, and then re-execute the `pdprg Renew` command.
2. If there is a HiRDB server process, forcibly stop HiRDB with the `pdstop -f` command, and then execute the `pdprg Renew -b` command. If there is no HiRDB server process, execute the `pdprg Renew -b` command. When the `pdprg Renew -b` command is executed, recovery processing restarts the pre-update HiRDB. Error messages and abort codes relating to a HiRDB shutdown processing failure may also be displayed. Follow the message handling procedure, and check that HiRDB has returned to the pre-update environment.
3. If there is a HiRDB server process, forcibly stop HiRDB with the `pdstop -f` command, and then execute the `pdprg Renew -b` command. If there is no HiRDB server process, execute the `pdprg Renew -b` command. When the `pdprg Renew -b` command is executed, recovery processing returns to the directory for updating the HiRDB update version. Error messages and abort codes relating to a HiRDB startup processing failure may also be displayed. If after updating to the HiRDB update version there are problems with the operating environment, follow the procedure for the displayed message.

(2) Checking whether HiRDB returned to pre-update status in the event of a failure during updating

If updating to the HiRDB update version failed, you can check whether HiRDB returned to its pre-update status by checking the following items. If these conditions are satisfied, HiRDB returned to the pre-update status.

- The version displayed by the `pdadmvr -s` command matches the HiRDB version before the update.
- The HiRDB is in online status (the result of the `pdls` command is that all units are displayed as `ACTIVE`).
- There is no backup directory (`$PDDIR/renew_bak`).

1.6 Using JP1/Software Distribution to execute online distribution of HiRDB

If you are upgrading multiple HiRDBs in a network, using JP1/Software Distribution to execute online distribution of HiRDB (remote installation) can reduce the work involved in HiRDB installation and setup on each server machine. To execute online distribution of HiRDB, you need one of the following products:

JP1 Version 5

- JP1/Software Distribution
- JP1/Software Distribution Workstation

JP1 Version 6

- JP1/Software Distribution Manager
- JP1/Software Distribution SubManager or JP1/Software Distribution Client

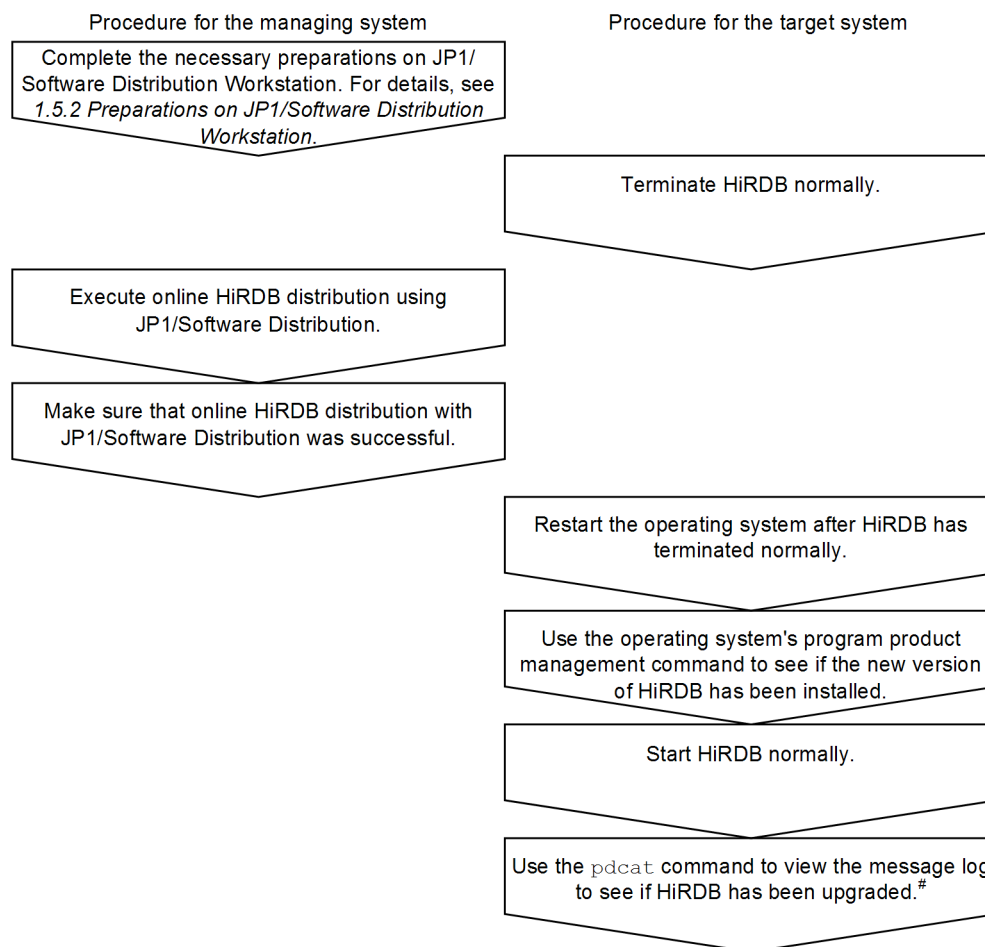
Notes about this section

- If you are using JP1 Version 6, replace JP1/Software Distribution with JP1/Software Distribution Manager and JP1/Software Distribution Workstation with JP1/Software Distribution SubManager or JP1/Software Distribution Client.
- This section assumes that you are familiar with the operation of JP1/Software Distribution and JP1/Software Distribution Workstation.
- Information provided here is for the following versions of HiRDB:
HP-UX
Solaris

1.6.1 Online distribution procedure

The following figure illustrates the HiRDB online distribution procedure that uses JP1/Software Distribution.

Figure 1–7: HiRDB online distribution procedure using JP1/Software Distribution



1.6.2 Preparations on JP1/Software Distribution Workstation

Before executing online HiRDB distribution, you need to set up your JP1/Software Distribution Workstation on the managing server. This section describes only those JP1/Software Distribution Workstation settings that require special attention.

(1) Specifying the registration file for JP1/Software Distribution Workstation

This section discusses the `InstallTiming` parameter that is specified in the registration file when registering HiRDB as a resource in JP1/Software Distribution Workstation.

OS type	Value of <code>InstallTiming</code> parameter
HP-UX or Solaris	Specify <code>BOOT</code> in the <code>InstallTiming</code> parameter. Example: <code>InstallTiming BOOT</code>

To automate the HiRDB setup procedure, specify a HiRDB setup automation shell during the program product installation postprocessing for the registration file (`APafterGeneration`). For details about the HiRDB setup automation shell, see subsection (2).

(2) Creating a HiRDB setup automation shell

To automate the HiRDB setup procedure, create a shell as the program product installation postprocessing. Specify the `pdsetup` command in this shell to set up HiRDB. This automates HiRDB setup after online distribution.

The following shows coding examples of the HiRDB setup automation shell:

(a) HP-UX HiRDB single server configuration

Example 1

This sample shell uses the installation directory as the HiRDB directory:

<code>/bin/chown <u>USR1</u> /opt/HiRDB_S</code>	1
<code>/bin/chgrp <u>GRP1</u> /opt/HiRDB_S</code>	1
<code>/bin/echo 'y' /opt/HiRDB_S/bin/pdsetup -d /opt/HiRDB_S</code>	
<code>/opt/HiRDB_S/bin/pdsetup -c <u>sjis</u> /opt/HiRDB_S</code>	2

Explanation:

1. If you use the installation directory as the HiRDB directory, redefine the directory's owner and group as the HiRDB administrator and HiRDB group. Replace the underlined parts with the actual HiRDB administrator's user ID and group ID.
2. Replace the underlined part with the actual character encoding you want to use.

Example 2

This sample shell does not use the installation directory as the HiRDB directory:

<code>DIR=<u>/usr/HiRDB1</u></code>	1
<code>/bin/echo 'y' /opt/HiRDB_S/bin/pdsetup -d \$DIR</code>	
<code>/opt/HiRDB_S/bin/pdsetup -c <u>sjis</u> \$DIR</code>	2
<code>DIR=<u>/usr/HiRDB2</u></code>	3
<code>/bin/echo 'y' /opt/HiRDB_S/bin/pdsetup -d \$DIR</code>	3
<code>/opt/HiRDB_S/bin/pdsetup -c <u>sjis</u> \$DIR</code>	3

Explanation:

1. Specify the name of the HiRDB directory.
2. Replace the underlined part with the actual character encoding you want to use.
3. Repeat as many times as there are HiRDBs being distributed online.

(b) HP-UX HiRDB parallel server configuration

Example 1

This sample shell uses the installation directory as the HiRDB directory:

<code>/bin/chown <u>USR1</u> /opt/HiRDB_P</code>	1
<code>/bin/chgrp <u>GRP1</u> /opt/HiRDB_P</code>	1
<code>/bin/echo 'y' /opt/HiRDB_P/bin/pdsetup -d /opt/HiRDB_P</code>	
<code>/opt/HiRDB_P/bin/pdsetup -c <u>sjis</u> /opt/HiRDB_P</code>	2

Explanation:

1. If you use the installation directory as the HiRDB directory, redefine the directory's owner and group as the HiRDB administrator and HiRDB group. Replace the underlined parts with the actual HiRDB administrator's user ID and group ID.
2. Replace the underlined part with the actual character encoding you want to use.

Example 2

This sample shell does not use the installation directory as the HiRDB directory:

<code>DIR=<u>/usr/HiRDB1</u></code>	1
<code>/bin/echo 'y' /opt/HiRDB_P/bin/pdsetup -d \$DIR</code>	
<code>/opt/HiRDB_P/bin/pdsetup -c <u>sjis</u> \$DIR</code>	2
<code>DIR=<u>/usr/HiRDB2</u></code>	3
<code>/bin/echo 'y' /opt/HiRDB_P/bin/pdsetup -d \$DIR</code>	3
<code>/opt/HiRDB_P/bin/pdsetup -c <u>sjis</u> \$DIR</code>	3

Explanation:

1. Specify the name of the HiRDB directory.

2. Replace the underlined part with the actual character encoding you want to use.
3. Repeat as many times as there are HiRDBs being distributed online.

(c) Solaris HiRDB single server configuration

Example 1

This sample shell uses the installation directory as the HiRDB directory:

```
/bin/chown USR1 /opt/HiRDB_S 1
/bin/chgrp GRP1 /opt/HiRDB_S 1
/bin/echo 'y' | /opt/HiRDB_S/bin/pdsetup -d /opt/HiRDB_S
/opt/HiRDB_S/bin/pdsetup -c ujis /opt/HiRDB_S 2
```

Explanation:

1. If you use the installation directory as the HiRDB directory, redefine the directory's owner and group as the HiRDB administrator and HiRDB group. Replace the underlined parts with the actual HiRDB administrator's user ID and group ID.
2. Replace the underlined part with the actual character encoding you want to use.

Example 2

This sample shell does not use the installation directory as the HiRDB directory:

```
DIR=/usr/HiRDB1 1
/bin/echo 'y' | /opt/HiRDB_S/bin/pdsetup -d $DIR
/opt/HiRDB_S/bin/pdsetup -c ujis $DIR 2
DIR=/usr/HiRDB2 3
/bin/echo 'y' | /opt/HiRDB_S/bin/pdsetup -d $DIR 3
/opt/HiRDB_S/bin/pdsetup -c ujis $DIR 3
```

Explanation:

1. Specify the name of the HiRDB directory.
2. Replace the underlined part with the actual character encoding you want to use.
3. Repeat as many times as there are HiRDBs being distributed online.

(d) Solaris HiRDB parallel server configuration

Example 1

This sample shell uses the installation directory as the HiRDB directory:

```
/bin/chown USR1 /opt/HiRDB_P 1
/bin/chgrp GRP1 /opt/HiRDB_P 1
/bin/echo 'y' | /opt/HiRDB_P/bin/pdsetup -d /opt/HiRDB_P
/opt/HiRDB_P/bin/pdsetup -c ujis /opt/HiRDB_P 2
```

Explanation:

1. If you use the installation directory as the HiRDB directory, redefine the directory's owner and group as the HiRDB administrator and HiRDB group. Replace the underlined parts with the actual HiRDB administrator's user ID and group ID.
2. Replace the underlined part with the actual character encoding you want to use.

Example 2

This sample shell does not use the installation directory as the HiRDB directory:

```
DIR=/usr/HiRDB1 1
/bin/echo 'y' | /opt/HiRDB_P/bin/pdsetup -d $DIR
/opt/HiRDB_P/bin/pdsetup -c ujis $DIR 2
DIR=/usr/HiRDB2 3
/bin/echo 'y' | /opt/HiRDB_P/bin/pdsetup -d $DIR 3
/opt/HiRDB_P/bin/pdsetup -c ujis $DIR 3
```

Explanation:

1. Specify the name of the HiRDB directory.
2. Replace the underlined part with the actual character encoding you want to use.
3. Repeat as many times as there are HiRDBs being distributed online.

1.6.3 Notes about online distribution

- When upgrading your HiRDB, be sure to terminate the HiRDB at the target system using the `pdstop` command. For details about how to upgrade HiRDB, see *1.4 Upgrading HiRDB*.
- No response is available during the program product installation postprocessing that is specified in the `APafterGeneration` parameter for the JP1/Software Distribution Workstation registration file. Therefore, if you are using the `pdsetup -d HiRDB` command that requests the user's response, suppress a response wait by passing an appropriate response with the UNIX `echo` command, as shown in the coding sample. Note that you cannot use any HiRDB command other than `pdsetup`.
- The `pdvrrup` command starts automatically. There is no need to enter this command while upgrading your HiRDB.

1.7 Migrating to 64-bit mode HiRDB

This section describes how to migrate from 32-bit mode HiRDB to 64-bit mode HiRDB on the same machine.

1.7.1 Considerations when migrating to 64-bit mode

(1) Incompatible files

Files that were used by 32-bit mode HiRDB can generally be used by 64-bit mode HiRDB. However, the following files are not compatible with 64-bit mode HiRDB and cannot be used.

- Backup files
- HiRDB files that consist of a master directory RDAREA and a data directory RDAREA

(2) Operands with altered default values

The default values of some HiRDB system definition operands change when HiRDB is migrated from the 32-bit mode to the 64-bit mode. Check the changes in *Operands with different default values in 32-bit and 64-bit mode* in the manual *HiRDB Version 9 System Definition*.

(3) Differences in memory requirements

When HiRDB is migrated from 32-bit mode to 64-bit mode, more memory is required. For details about calculating memory requirements, see *Chapter 15. Storage Requirements for HiRDB*.

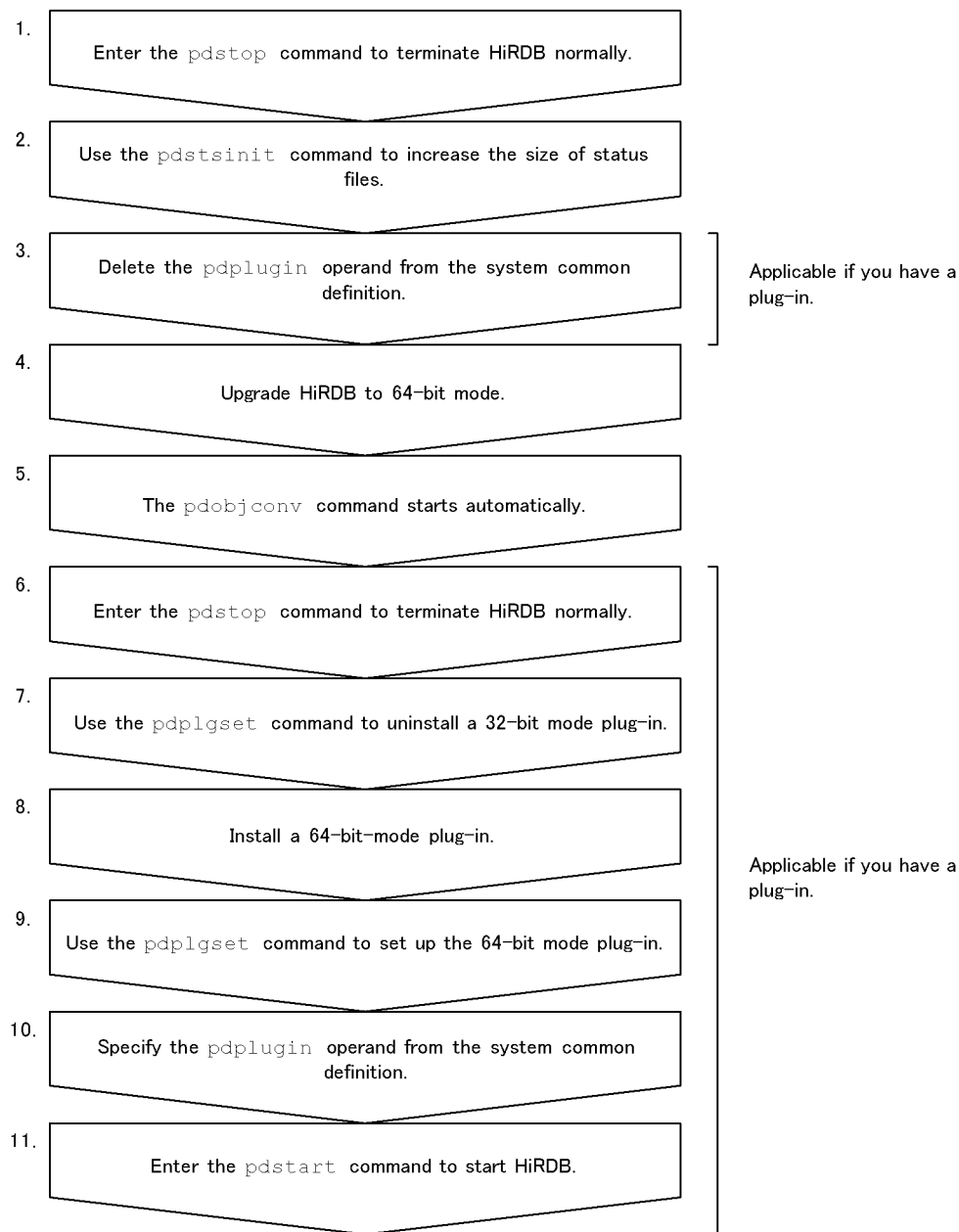
(4) Differences in the UOC interface

When HiRDB is migrated to 64-bit mode, the UOC interface changes for the database creation utility (`pdload`) and the database reorganization utility (`pdroorg`). This means that the UOC must be re-created. For details about the UOC interface, see the manual *HiRDB Version 9 Command Reference*.

1.7.2 How to migrate to 64-bit mode

The following figure illustrates how to migrate to 64-bit mode.

Figure 1–8: How to migrate to 64-bit mode



1. Enter the `pdstop` command to terminate HiRDB normally.

2. Use the `pdstsinit` command to increase the size of status files.

See *17.3 Determining the size of status files* to re-estimate the size of status files. If necessary, use the `pdstsinit` command to increase the size of status files.

3. Delete the `pdplugin` operand from the system common definition.

Delete the `pdplugin` operand from the system common definition. Otherwise, HiRDB does not start normally after being upgraded to 64-bit mode.

4. Upgrade HiRDB to 64-bit mode.

Upgrade your HiRDB to 64-bit mode. For details about how to upgrade HiRDB, see *1.4 Upgrading HiRDB*.

Before upgrading HiRDB to 64-bit mode, check the data dictionary RDAREA for available space. In this case, you need the space discussed in *1.4.1 Before upgrading*, plus the space for the following tables:

SQL_TABLES table: $1 + \uparrow 5 \div S \uparrow$

SQL_VIEW_DEF table: $2 + \uparrow 200 \div S \uparrow$

5. The `pdobjconv` command starts automatically.

The `pdvtrup` command is executed during the upgrading procedure. When this `pdvtrup` command terminates normally, the `pdobjconv` command^{#1} is executed automatically. If this command returns code^{#2} 0 or 4, migration to 64-bit mode was successful. If the return code is neither 0 nor 4, continue the 64-bit-mode migration procedure with the procedure explained in *1.7.3 In the event of an SQL object migration error*.

#1: This command makes the view tables and SQL objects for procedures and functions created in 32-bit mode also available in 64-bit mode.

#2: The `KFPX21002-I` message displays the return code. This message is output to the system log file and `syslogfile`. If the return code is 8 or 12, the message is also output to the standard error output. The following explains the return code:

0:

The `pdobjconv` command terminated normally.

4:

There was a warning message, but the `pdobjconv` command terminated normally.

8:

A migration error occurred on some of the SQL objects. Check the cause of the error and correct it based on the message and `pdobjconv` command processing result (SQL object migration information). Another possibility is that a utility-execution error occurred.

12:

The `pdobjconv` command terminated abnormally. Check the cause of the error and correct it based on the message and `pdobjconv` command processing result (SQL object migration information). If you use the `pdcancel` command to cancel the `pdobjconv` command, or an error occurs in the `pdobjconv` command process, the command returns code 12.

6. Enter the `pdstop` command to terminate HiRDB normally.

7. Use the `pdplgset` command to uninstall the 32-bit-mode plug-ins.

Execute the command in the format `pdplgset -d plug-in-name`.

With some plug-ins, you need to make a backup copy before uninstalling them. See each plug-in's documentation to determine whether it needs to be backed up.

8. Install the 64-bit-mode plug-ins.

Install the applicable 64-bit-mode plug-ins. For the installation procedure, see each plug-in's documentation.

9. Use the `pdplgset` command to set up the 64-bit-mode plug-ins.

Execute the command in the format `pdplgset plug-in-name plug-in-installation-directory-name`.

10. Specify the `pdplugin` operand in the system common definition.

Specify the name of the 64-bit-mode plug-in in the `pdplugin` operand in the system common definition.

11. Enter the `pdstart` command to start HiRDB.

1.7.3 In the event of an SQL object migration error

This section discusses actions that should be taken if the `pdobjconv` command returns code 8 or 12.

(1) Return code 8

A migration error occurred in some of the SQL objects. See the SQL object migration information to check for the SQL objects that resulted in a migration error. For details about the SQL object migration information, see the `pdobjconv` command in the manual *HiRDB Version 9 Command Reference*.

To migrate an SQL object that resulted in a migration error, check the output message and eliminate the cause of the error, then execute the `pdobjconv` command. If you have terminated HiRDB, start HiRDB again and then execute the `pdobjconv` command.

(2) Return code 12

The `pdobjconv` command terminated abnormally. Check the output message and eliminate the cause of the abnormal termination, then execute the `pdobjconv` command. If you have terminated HiRDB, start HiRDB again and then execute the `pdobjconv` command.

1.7.4 In the event of a 64-bit-mode migration error (restoring the old version)

For details about how to restore the old version of HiRDB due to a 64-bit-mode migration error, see *1.4.6 In the event of an upgrading error*.

After restoring the old version of HiRDB according to the instructions provided in *1.4.6 In the event of an upgrading error*, use the `pdstsinit` command to initialize all status files. If you do not initialize all status files, you cannot start HiRDB normally.

2

Installation

This chapter describes tasks that are required before and after installation, and explains procedures for installing and uninstalling HiRDB. It also provides notes concerning installation of option program products.

2.1 Pre-installation procedure

This section describes the procedures that must be completed before HiRDB can be installed.

If you are using a HiRDB parallel server configuration, use the same version platform on all server machines.

2.1.1 Checking and modifying OS parameters

Executor: Superuser

The message queue and semaphore requirements used by HiRDB must be estimated, and the OS parameters (kernel parameters) must be changed to reflect these requirements. For the details about specifying the OS parameters, see Chapter 20. *Specifying OS Parameters*.

For AIX, the following also apply:

- You must first edit the `/etc/security/limits` file to change the `root` user's or HiRDB administrator's limitation values for OS system resources, and then restart the OS.
For example, in AIX, the default limitation value for a regular file is 1 gigabyte. If the size of work files output during utility execution exceeds 1 gigabyte, the HiRDB utility terminates abnormally. In such cases, you must change limitation values for OS system resources; you must make these changes in advance.
- The syslogfile must be specified for output.

If `No-space` (insufficient space to write in the file) is output as an error message during installation, the following may be the cause:

- If this error occurs even when disk space is sufficient, you might have not defined the HiRDB file system area for a large file, or the case falls under OS kernel parameter restrictions. For details about OS kernel parameter restrictions, see Chapter 20. *Specifying OS Parameters*.

2.1.2 Registering the HiRDB administrator

Executor: Superuser

In the OS of all server machines, register the user who will administer HiRDB. The following table lists the information to be registered.

Table 2–1: Information to be registered in OS on each server machine

Information to be registered	Format
Log-in name	1-8 alphanumeric characters beginning with an alphabetic character [#]
User ID	Any characters
Group ID	
Home directory	
Login shell	

[#]: None of the following words can be used: `ALL`, `HiRDB`, `MASTER`, or `PUBLIC`.

If the system switchover facility is to be used with a HiRDB parallel server configuration or HiRDB single server configuration, specify the same user ID as the HiRDB administrator registered in the OS for all server machines that operate HiRDB. Note that it is not necessary to use the same password.

Note

A password should be registered after the user ID has been registered.

The user who uses the user ID registered here to log onto the system is called the HiRDB administrator. The following privileges are granted to the HiRDB administrator:

1. Access privilege as the owner of various HiRDB system files and directories. The holder of this privilege can prohibit write operations by other users.
2. Privilege to execute the HiRDB operation commands and utilities.

Multi-HiRDB

A HiRDB administrator must be registered for each HiRDB.

2.1.3 Setting up a HiRDB group

Executor: Superuser

A superuser can set up a group of HiRDB users in the OS of each server machine. The name of the group must be eight or fewer alphanumeric characters starting with a letter.

By setting a HiRDB group, users other than the group can be blocked from creating files under the HRDB file system area and HiRDB directory. This enhances security.

Multi-HiRDB

If a separate group is set up for each HiRDB, each HiRDB's users can be monitored.

2.1.4 Creating the installation directory

Executor: Superuser

To protect the root partition from excessive workload, the superuser must create an installation directory before installing HiRDB. An installation directory should be created in a dedicated disk partition to avoid an excessive workload on the file system.

HiRDB is installed under this installation directory.

For details about disk partitions, see the applicable OS documentation.

To install a new HiRDB, create the installation directory on each server machine. The following table lists the HiRDB installation directories.

Table 2–2: HiRDB installation directories

HiRDB type	HP-UX, Solaris, AIX, or Linux
HiRDB single server configuration	/opt/HiRDB_S
HiRDB parallel server configuration	/opt/HiRDB_P

2.1.5 Registering host names

Register the host names that HiRDB will use (host names specified by the system definitions and client environment definitions) in the `hosts` file or DNS, and resolve the names.

Specify host names in the system definitions and client environment definitions in host name, IP address or FQDN format.

When the HiRDB system is configured with only a HiRDB single server configuration[#], a loopback address can be specified in the system definitions or client environment definitions. If a loopback address is specified, the host name need not be registered.

#

A HiRDB system made up of HiRDB single server configurations only is one that meets the following conditions:

- The HiRDB clients are on the same machines as the HiRDB servers (there are no HiRDB clients on separate machines).
- There is no utility special unit.

Reference note

A loopback address is an IP address in the range 127.0.0.0 to 127.255.255.255 (for example, 127.0.0.1). The OS specifications dictate which IP addresses can be specified as loopback addresses.

Note that since HiRDB treats `localhost` as an ordinary host name, when `localhost` is specified as a host name in the system definitions, it must be registered as a host name and name resolution must be performed.

2.2 HiRDB installation procedure

This section explains the HiRDB installation procedure.

2.2.1 Installing HiRDB

Executor: Superuser

Use Hitachi Program Product Installer to install HiRDB on each server machine.

Upgrading HiRDB

Before installing HiRDB, check for any active HiRDB under the installation directory (such as by using the OS's `ps` command). If there is an active HiRDB, after it is terminated normally using the `pdstop` command, execute the `pdsetup -d` command, respond with `y` to message KFPS00036-Q, and then delete HiRDB from the OS. After this, perform the installation.

HiRDB parallel server configuration

Install the same version of HiRDB on all server machines that constitute the HiRDB parallel server configuration.

Multi-HiRDB

When you install multiple HiRDB single server configurations or multiple HiRDB parallel server configurations, they all have the same installation directory. If you install multiple HiRDBs one after another, only the last HiRDB installed is effective because they all share the same installation directory.

Therefore, once you have installed the first HiRDB, create its HiRDB directory and register the HiRDB in the operating system (by following the procedures described in 2.3.1 *Creating the HiRDB directory* and 2.3.3 *Registering HiRDB and option program products in the operating system*). You can then install the next HiRDB.

2.2.2 Installing option program product

Executor: Superuser

To use HiRDB option program products, you must first install them. The following table lists the option program product facilities and, for a HiRDB parallel server configuration, servers where option program products are installed.

Table 2-3: Option program products facility and target installation servers

Product name	Facilities available with introduction of the option program product	Target installation server machine
HiRDB Staticizer Option	<ul style="list-style-type: none"> Inner replica facility Updatable online reorganization 	All server machines
HiRDB Advanced High Availability	<ul style="list-style-type: none"> Standby-less system switchover (1:1) facility Standby-less system switchover (effects distributed) facility 	
	Dynamic update of global buffer (<code>pdbufmod</code> command)	
	System reconfiguration command (<code>pdchgconf</code> command)	
	Table matrix partitioning	
	Changing the partition storage conditions (<code>ALTER TABLE</code>)	
HiRDB Non Recover FES	Recovery-unnecessary front-end server	
HiRDB Accelerator	<ul style="list-style-type: none"> Memory database Rapid batch facility using in-memory data processing 	

Product name	Facilities available with introduction of the option program product	Target installation server machine
HiRDB Disaster Recovery Light Edition	Real Time SAN Replication of log-only synchronous method	All server machines configuring a HiRDB system at a business site and all server machines configuring a HiRDB system at a log applicable site.

2.2.3 Installing plug-ins

The superuser must install a plug-in by using the OS's installer. For the installation procedure, see the documentation provided with the corresponding software.

2.3 Post-installation procedures

This section describes the procedures that must be completed after HiRDB has been installed.

2.3.1 Creating the HiRDB directory

Executor: HiRDB administrator

The superuser must create a HiRDB directory in each server machine. This directory is used to execute HiRDB. The HiRDB directory stores various directories and files.

You must not use the installation directory as the HiRDB directory. If the installation directory is used for the HiRDB directory, disk space problems could result or installation may fail. For each installation, the owner of the installation directory must change to the HiRDB administrator at the root, and the changes shown in (1) below must be made to the group and mode.

(1) Information to be specified in the HiRDB directory

You can assign any name to the HiRDB directory, but be sure to specify all the following information. The pathname of the HiRDB directory must be no longer than 128 characters (bytes). For Linux, specify 1-27 characters (bytes) as the pathname.

- Directory name: *Any name*
- Owner: HiRDB administrator
- Group: HiRDB group
- Mode: 0755

Also note the following:

- Specify a character string that begins with / (forward slash) and that consists of only the following characters:
 - Alphanumeric characters
 - _ (underscore)
 - . (period)
 - / (forward slash separating paths)
- The forward slash (/) alone cannot be specified.
- A path name must not exceed 128 characters (bytes). For the Linux version, a path name must not exceed 118 characters (bytes).

(2) Points to be noted when creating the HiRDB directory

1. To create the HiRDB directory, you need at least 2 gigabytes of disk space. This is just a guideline; the actual size that is required depends on the size of the system.
2. You should not create the HiRDB directory in the root directory. Files containing troubleshooting information are created in the HiRDB directory (files under `$PDDIR/spool`). If you do not delete these files periodically using the `pdcsPOOL` command, a space shortage may occur on the disk, resulting in adverse effects on operating system operation.
If you do create the HiRDB directory in the root directory, you should use a different partition than that of the root directory.
3. Create the HiRDB directory on a local disk. Do not specify `nosuid` as the mount type of the file system in `/etc/checklist` or `etc/fstab`.
4. Do not use the HiRDB directory as the target of a symbolic link.
5. For multi-HiRDB, create a HiRDB directory with a different name for each HiRDB.
6. To employ a 2-to-1 or mutual system switchover organization for a HiRDB parallel server configuration, you cannot share the same HiRDB directory among all server machines. For details about how to define the HiRDB

directory, see *Example of mutual switchover configuration* in *System configuration examples of a HiRDB parallel server configuration* in the *HiRDB Version 9 System Operation Guide*.

(3) Deleting files from the HiRDB directory

HiRDB outputs troubleshooting information to `$PDDIR/spool` in situations such as when a server process or client is forcibly terminated. When a command or utility is aborted by pressing **Ctrl + C** (or some other method) and no output destination has been explicitly specified for work files, temporary work files output under `$PDDIR/tmp` by the command or utility are retained rather than deleted. These files are often the cause of a shortage of disk space available to the HiRDB directory. Because insufficient disk space for the HiRDB directory can cause abnormal termination, HiRDB deletes the following files periodically:

- Troubleshooting information files (files in `$PDDIR/spool`)
- Temporary work files (files in `$PDDIR/tmp`)
- Files in the directory specified in the `pd_tmp_directory` operand

For details about these consistently increasing files, see *1.3.2 Files that consistently increase in size*.

Normally, these files are deleted every 24 hours. The interval between deletions can be changed with the `pd_spool_cleanup_interval` operand. You can also specify deletion of only those files output prior to the date specified by the `pd_spool_cleanup_interval_level` operand.

It is also possible to delete all at once all troubleshooting information (all files in `$PDDIR/spool`).

- Use the `pdcspool` command to delete the troubleshooting information files. Temporary work files (in `$PDDIR/tmp`) can also be deleted.
- Automatically delete the troubleshooting information files during the HiRDB startup. In this case, use the `pd_spool_cleanup` operand to specify whether to delete the troubleshooting information files. If you omit this operand, the troubleshooting information files are deleted automatically.

You can also specify deletion of only those troubleshooting information files output prior to the date specified by the `pd_spool_cleanup_level` operand.

Note

You can select the troubleshooting information to be deleted using the `pdcspool` command's option, `pd_spool_cleanup_level` operand or `pd_spool_cleanup_interval` operand.

(4) Making a backup copy of the HiRDB directory

To prepare for possible errors on the disk that contains the HiRDB directory, make a backup copy of the files in the HiRDB directory (files under `$PDDIR/conf`). You need the backup copy of the files under `$PDDIR/conf` to restore the HiRDB directory. There are HiRDB system definition files under `$PDDIR/conf`. If you have modified the HiRDB system definition, back up the files contained under `$PDDIR/conf`.

If you have created user files under the HiRDB directory, also back up those files. You need this backup copy to restore the HiRDB directory.

For details about how to recover the HiRDB directory, see the *HiRDB Version 9 System Operation Guide*.

- Backing up the disk that contains the HiRDB directory
If you need to back up the disk that contains the HiRDB directory, use the following procedure:

1. Terminate HiRDB with the `pdstop` command.
2. Execute the `pdsetup -d` command. Enter `n` as a response.#
3. Use the OS function (command) to back up the disk that contains the HiRDB directory.
4. Execute the `pdsetup` command.
5. Enter the `pdstart` command to start HiRDB.

#: You need to execute the `pdsetup -d` command to remove all the files that may cause problems after recovery. Additionally, you need to enter `n` as a response to retain all the files that are required after recovery.

2.3.2 Creating a work file output directory

Executor: HiRDB administrator

You can create a directory to serve as the output destination of work files output by HiRDB. You can then specify that directory as the output destination for the various work files that are generated when commands or utilities are executed. Since there will consequently be only one output destination for each unit, the difficult job of administering work files becomes easier. HiRDB can be operated without creating a work file output directory, but work files are then output to a variety of destinations and it is impossible to delete work files using the `pdcsPOOL` command. For this reason, we recommend that you create a work file output directory.

(1) Calculating the size of the work file output directory

Set the free space in the work file output directory to be at least the value shown below. When HiRDB or a command terminates abnormally while a work file is being output, the work file is not deleted. For this reason, before executing the `pdcsPOOL` command, set a value that ensures sufficient space in the work file output directory so that disk space does not run short.

Work file output directory size (kilobytes) =
 $178,224 + a + b + c + d + e + f + g + h$

Variable	Description	Formula reference site
<i>a</i>	Space for the process results file when <code>pdconstck</code> is executed	<i>19.1.10 File sizes required for execution of the integrity check utility (pdconstck)</i>
<i>b</i>	Space for the following files when <code>pddbstd</code> is executed: <ul style="list-style-type: none"> • Work files • Work files for sorting 	<i>19.1.4 File sizes required for the execution of the database condition analysis utility (pddbstd)</i>
<i>c</i>	Space for the following files when <code>pdload</code> is executed: <ul style="list-style-type: none"> • Index information file • Error information file • Temporary file for creating error information file • LOB middle file • Work file for sorting 	<i>19.1.1 File sizes required for the execution of the database load utility (pdload)</i>
<i>d</i>	When <code>pdorend</code> is executed	$2,400 \times \text{mapping key column count} \times \text{skipped SQL statement count} + 100$
<i>e</i>	Space for the following files when <code>pdrbal</code> is executed: <ul style="list-style-type: none"> • Index information file • Work file for sorting 	<i>19.1.9 File sizes required for execution of the rebalancing utility (pdrbal)</i>
<i>f</i>	Space for the following files when <code>pdorg</code> is executed: <ul style="list-style-type: none"> • Index information file • Work file for sorting 	<i>19.1.2 File sizes required for the execution of the database reorganization utility (pdorg)</i>
<i>g</i>	Space for the work directory for sorting when <code>pdrstr -w</code> is executed	The manual <i>HiRDB Version 9 Command Reference</i>
<i>h</i>	Space for the following files when <code>pdstedit</code> is executed: <ul style="list-style-type: none"> • Work files • Work file for sorting 	<i>19.1.3 File sizes required for the execution of the statistics analysis utility (pdstedit)</i>

Variable	Description	Formula reference site
	<ul style="list-style-type: none"> DAT-format files 	

(2) Specifying the work file output directory

To create a single output destination for work files, specify the created directory in the `pd_tmp_directory` operand.

If the `pd_tmp_directory` operand is not specified, HiRDB outputs work files to the directories determined by individual commands or utilities. The work file output destination is the following:

1. The output destination specified in the command option or utility control statement.
2. If not specified as in location 1, the output destination specified by the `pd_tmp_directory` operand.
3. If not specified as in location 2, the output destination specified by environment variable `TMPDIR`[#].
4. If not specified as in location 3, the `/tmp` directory.

#

If the command is running on the server side, this will be the environment variable `TMPDIR` set in the process server process (`pdprcd`).

(3) Deleting work files

HiRDB normally deletes work files every 24 hours. This deletion interval can be changed with the `pd_spool_cleanup_interval` operand. You can also specify deletion of only those files output prior to the date specified by the `pd_spool_cleanup_interval_level` operand. In this case, files in the work file output directory specified in the `pd_tmp_directory` operand are deleted.

Those work files output by commands and utilities that are not deleted by HiRDB must be deleted periodically using the `pdcspool` command. In this case as well, files in the work file output directory specified in the `pd_tmp_directory` operand are deleted.

For details about deleting work files, see 2.3.1(3) *Deleting files from the HiRDB directory*.

2.3.3 Registering HiRDB and option program products in the operating system

Executor: Superuser

(1) OS registration method

(a) Executing the `pdsetup` command

The superuser executes the `pdsetup` command to register HiRDB into the OS. The `pdsetup` command copies the directories and files from the installation directory to the HiRDB directory. For a HiRDB parallel server configuration, execute the `pdsetup` command for each server machine.

To perform registration into the OS:

1. Execute the `pdsetup` command.
2. Check the termination status.

If the termination status of the `pdsetup` command is not 0, an error may have occurred. See the message output to the syslog file, eliminate the cause of the error, and then re-register HiRDB into the OS.

For details about the `pdsetup` command, see the manual *HiRDB Version 9 Command Reference*.

Notes

- The `pdsetup` command is located in the `bin` directory under the HiRDB installation directory.

- If you execute the `pdsetup` command, the `inittab` directory is automatically created in the `conf` directory under the HiRDB installation directory. This directory is used to back up the `/etc/inittab` file. If the `/etc/inittab` file is damaged during or after the execution of the `pdsetup` command, use this backup file to start the operating system.
- After you execute the `pdsetup` command, do not delete files or directories created under the HiRDB directory, and do not change owners or access permissions. Doing so can prevent HiRDB from starting.
- The `pdsetup` command copies or overwrites the installed HiRDB load modules to the specified HiRDB directory. To prevent loss of other program or user data, specify either of the following in the HiRDB directory:
 - A directory created exclusively for HiRDB
 - A HiRDB installation directory (for a HiRDB single server configuration, `/opt/HiRDB_S`; for a HiRDB parallel server configuration, `/opt/HiRDB_P`)

(b) Default operand behavior selected by the `pdsetup` command

The `-v` option of the `pdsetup` command can be used to choose the default behavior when operands are omitted. After you have executed the `pdsetup` command to register HiRDB in the OS, if you want to change the operand default behavior, execute the `pdsetenv` command. To check the mode being used to apply the default operand behavior, execute the command `pdadmvr`, or check message `KFPS01826-I`, which is output when HiRDB starts.

Rules

- In a system switchover configuration, the default behavior when operands are omitted must be the same in the primary and secondary systems.
- In a HiRDB parallel server configuration, the default behavior when operands are omitted must be the same for all units.

(c) Specifying the character encoding by using the `pdsetup` command

■ Specifying at the server

Use the `-c` option of the `pdsetup` command to specify the character encoding to be used in with HiRDB. The following table shows the character encodings that can be used with HiRDB.

Table 2-4: Character encoding supported by HiRDB

Character encoding	OS			
	HP-UX	Solaris	AIX	Linux
Shift JIS Kanji Code	Default	Usable	Default	Usable
EUC Chinese Kanji Code	Usable	Usable	Usable	Usable
Chinese Character Code (GB18030)	Usable	Usable	Usable	Usable
Single-byte character encoding	Usable	Usable	Usable	Usable
EUC Japanese Kanji Code	Usable	Default	Usable	Default
Unicode (UTF-8)	Usable	Usable	Usable	Usable
Unicode (IVS-supported UTF-8)	Usable	Not usable	Usable	Usable

Default: Usable character encoding (the default character encoding when the `pdsetup` command `-c` option is omitted)

Usable: Usable character encoding

Not usable: Not a usable character encoding

Use the `pdadmvr -c` command to check which character encoding is in use.

■ Specifying at the client

On the client side, specify the character encoding with the `LANG` or `PDCLTLANG` operand in the client environment definition as appropriate to the server's character encoding. The client environment definition is

referenced during UAP creation or execution. For the client environment variables supported based on the server's available character encoding, see the *HiRDB Version 9 UAP Development Guide*.

(d) Specifying the `pdsetup` command when using the POSIX library version

If you are using the POSIX library version, specify the `-l` option in the `pdsetup` command. You need the POSIX library version to use the following facilities:

- Java stored procedures and Java stored functions[#]

For details about Java stored procedures and Java stored functions, see the *HiRDB Version 9 UAP Development Guide*.

The Linux version supports Java stored procedures and Java stored functions without requiring any special setup.

[#]: The following versions of HiRDB support Java stored procedures and Java stored functions:

- HP-UX version (32-bit mode POSIX library version) and HP-UX (IPF version)
- Solaris version (32-bit mode POSIX library version)
- AIX version (32-bit mode POSIX library version)
- Linux version (32-bit mode) and Linux version (EM64T)

(2) Registering option program products

To install option program products, use the `pdopsetup` command to register them in the OS.

■ When using a HiRDB parallel server configuration

Execute the `pdopsetup` command on each machine to make sure the option program product is registered in all units.

■ When using the system switchover facility

- When you register the option program product on the primary system, register the same product on the secondary system (execute the `pdopsetup` command on the secondary system also).
- If two or more HiRDB systems are registered to the OS on a single server machine, such as in a system switchover configuration, execute the `pdopsetup` command on each HiRDB system (or each unit in the case of a HiRDB parallel server configuration). For example, in a mutual system switchover configuration, if two HiRDB systems (primary and secondary) exist on one server machine, the `pdopsetup` command must be executed twice on the server machine (once on the primary HiRDB system and once on the secondary HiRDB system).

■ When using Real Time SAN Replication

If you register an option program product on the main site, register the same product on the remote site (execute the `pdopsetup` command on the remote site also).

2.3.4 Setting environment variables

(1) Environment variables that need to be set by the HiRDB administrator

Set the environment variables shown in the table below in the HiRDB administrator environment.

Place the environment variables in one of the following files according to the server machine's login shell:

- Bourne shell: `$HOME/.profile`
- C shell: `$HOME/.cshrc`

Table 2–5: Information to be set in HiRDB administrator defined environment variables

Environment variable	Information to be set
PDDIR ^{#1}	Specifies the absolute path name of the HiRDB directory.

Environment variable	Information to be set
PDCONFPATH ^{#2}	Specifies the absolute path name of the directory that is used to store the HiRDB system definition file. The unit control information definition always uses the file under \$PDDIR/conf, regardless of this setting. If the PDCONFPATH operand is specified in the unit control information definition, the same setting must be specified here as in that PDCONFPATH operand.
PATH	Adds \$PDDIR/bin.
SHLIB_PATH ^{#3}	Adds \$PDDIR/lib.

#1: The PDDIR absolute path name can contain the following byte lengths:

- HP-UX, Solaris, and AIX: 128 bytes
- Linux: 118 bytes

#2: The PDCONFPATH absolute path name can contain a maximum of 213 bytes.

#3: For Solaris and Linux, use LD_LIBRARY_PATH; for AIX, use LIBPATH.

For details about the LANG and PDLANG environment variable settings, see the *HiRDB Version 9 UAP Development Guide*.

Note

If the pd_tmp_directory operand is specified, there is no need to specify the environment variable TMPDIR.

If the environment variable TMPDIR is omitted when the operand pd_tmp_directory is not specified, HiRDB creates temporary files in the same directory it would have done if the operand pd_tmp_directory had been omitted. In addition, when the environment variable TMPDIR is specified, the directory path must not exceed 512 bytes in length. When TMPDIR is set and HiRDB commands or utilities are interrupted, HiRDB might create files beginning with pdcmd or plcmd in the directory specified by TMPDIR. If some files that begin with pdcmd or plcmd remain in that directory after completion of the HiRDB commands or utilities, you must delete them, such as by using the operating system's rm command.

(2) Environment variables that need to be set by users who execute UAPs

If a UAP is to be executed, the HiRDB administrator must set a client environment definition in the environment variables for each applicable user. The format of the client environment definition depends on the client machine's OS or shell being used. For details about the information to be set in the client environment definition, see the *HiRDB Version 9 UAP Development Guide*.

(3) Environment variables that need to be set by users who define tables and indexes

If tables and indexes are to be defined, you need to set the following environment variables for each executing user. Shown here are the environment variables to be set in the Bourne shell environment.

- PDHOST=*HiRDB-server's-host-name* [, *secondary-HiRDB-server's-host-name*]
This environment variable specifies the host name of the HiRDB server to be connected. For a HiRDB parallel server configuration, specify the host name of the server machine at which the system manager is defined. If PDFESHOST is specified, you can specify the host name of PDFESHOST. When the host name of PDFESHOST is specified, the HiRDB server can be connected even if an error occurs on the system manager unit.
- \$PDUSER=*authorization-identifier/password*
This environment variable specifies the authorization identifier and password. You need to assign the schema definition privilege to the user specified here. For the authorization identifier specified for defining a schema (CREATE SCHEMA), assign the authorization identifier specified by this environment variable. To use lower-case alphabetic characters for the authorization identifier and password, use the "*authorization-identifier*" / "*password*" format.
- PDNAMEPORT=*HiRDB-server's-port-number*
This environment variable specifies the port number of the HiRDB server. Specify the port number of the HiRDB system that accesses the host specified in PDHOST.

2.3.5 Setting a remote shell execution environment

Executor: HiRDB administrator

Set a remote shell execution environment in the following cases:

- For a HiRDB parallel server configuration.
- When you are using the system switchover facility without inheriting IP addresses
- When a utility special unit is used.

(1) For a HiRDB parallel server configuration

Some HiRDB commands are executed at each server machine using the remote shell function. For this purpose, you must set appropriate information in `/etc/hosts.equiv` or `$HOME/.rhosts` to enable mutual log-in between the server machines that constitute the HiRDB parallel server configuration.

When you create the `$HOME/.cshrc` file without a registration terminal, make sure that data is not output to the standard output file or the standard error file.

(2) When you are using the system switchover facility without inheriting IP addresses

When you are using the system switchover facility without inheriting IP addresses, use the remote shell facility to transfer running system information to the standby system. For this purpose, you must set appropriate information in `/etc/hosts.equiv` or `$HOME/.rhosts` to enable mutual log-in between the server machine constituting the primary system and the server machine constituting the secondary system.

When you create the `$HOME/.cshrc` file without a registration terminal, make sure that data is not output to the standard output file or the standard error file.

(3) When a utility special unit is used

Some HiRDB commands entered from a HiRDB single server configuration can be executed on a utility special unit by using the remote shell facility. For this purpose, the superuser must set appropriate information in `/etc/hosts.equiv` or `$HOME/.rhosts` to enable mutual log-in between the HiRDB single server configuration and the utility special unit.

When the `$HOME/.cshrc` file is created without a registration terminal, the superuser should ensure that data is not output to the standard output file or the standard error file.

2.3.6 Notes about background execution of HiRDB operation commands

To execute HiRDB operation commands in the background mode, care must be taken to ensure that background job terminal output is not suppressed by a terminal port option.

If a command is executed while background job terminal output is suppressed, the command will not terminate and an invalid process will remain in the HiRDB system.

The `tostop` option of the `stty` command is used to set the terminal port option. For the specification of the `stty` command's `tostop` option, see the applicable OS documentation.

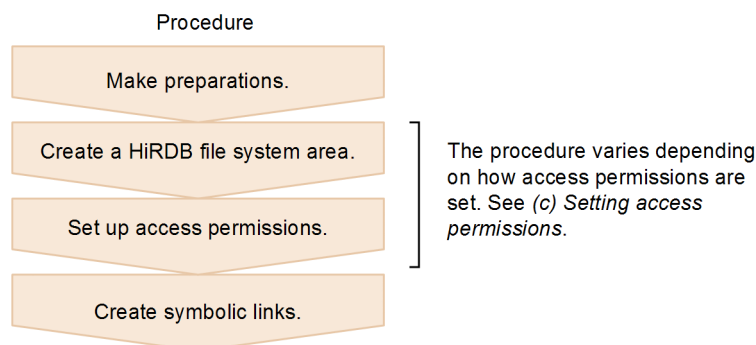
2.3.7 Preparing to create the HiRDB file system area

This section describes tasks you must do before you create a HiRDB file system area, procedures for creating a HiRDB file system area, and Hitachi's approach to access permissions for HiRDB file system areas.

(1) Creating a HiRDB file system area in an ordinary file

The following figure illustrates the procedure for creating a HiRDB file system area in an ordinary file.

Figure 2–1: Creating a HiRDB file system area in an ordinary file

**(a) Preparations****Executor: Superuser**

Do the following:

- Initialize a hard disk.
- Set up partitions on the initialized hard disk.
- Initialize the setup partitions as a UNIX file system.

See the OS documentation for instructions on performing these tasks.

(b) Creating a HiRDB file system area**Executor: HiRDB administrator**

Execute the `pdfmkfs` command to create a HiRDB file system area in the UNIX file system area. The file mode initial value is 660 (octal).

(c) Setting access permissions**Executor: HiRDB administrator**

To prevent unauthorized access by users lacking permission, change the file mode of the created HiRDB file system area.

Change the file mode using the `umask` or `chmod` command. Execute the `umask` command prior to creating the HiRDB file system area, and the `chmod` command after creating the HiRDB file system area.

See the OS documentation for details about these commands.

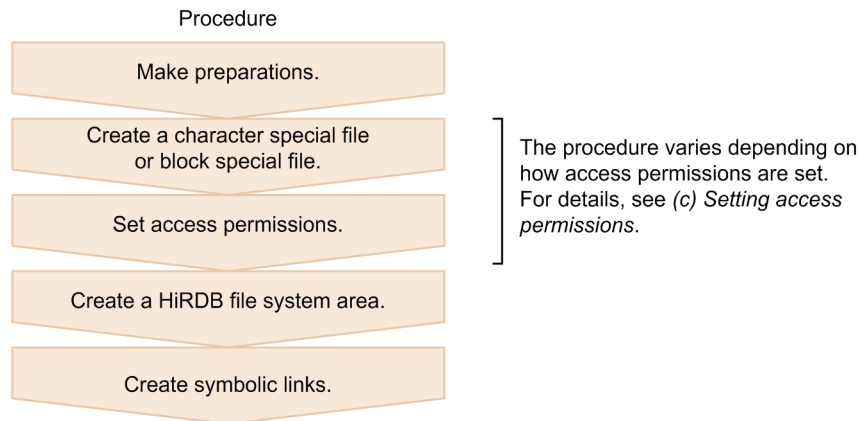
For details about access restrictions, see (3) *HiRDB file system area access permissions*.**(d) Setting up symbolic links****Executor: HiRDB administrator**

We recommend using a name symbolically linked to the actual name using the OS's `ln` command rather than using the unaltered actual name of an ordinary file as the name of the HiRDB file system area.

See the OS documentation for details about the `ln` command.**(2) Creating a HiRDB file system area in a character special file or block special file**

The procedure to create a HiRDB file system area in a character special file or block special file is shown in the following figure.

Figure 2–2: Procedure to create a HiRDB file system area in a character special file or block special file



(a) Preparations

Executor: Superuser

Do the following:

- Initialize a hard disk.
- Set up partitions on the initialized hard disk.

See the OS documentation for instructions on performing these tasks.

(b) Creating a character special file or block special file

Executor: Superuser

Create a character special file or block special file and set its mode.

Creating a file

■ Character special file

Execute the `mknod` command to create a character special file for a disk partition.

For Linux 5 and later, create the file using a method other than the `mknod` command.

The following table shows how to create character special files in each OS. For details about the commands and functions, see the OS documentation.

Table 2–6: Creating character special files

OS	Disk partition	Creation method	Notes
Linux 5 or later	LV	Execute the <code>raw</code> command. To enable the disk partition, execute the <code>raw</code> command after LV is recognized.	To enable the disk partition, execute the <code>raw</code> command again when the OS restarts. To specify automatically execution of the <code>raw</code> command when the OS restarts, state the <code>raw</code> command in <code>/etc/rc.local</code> .
	Non-LV	Use the <code>udev</code> function.	Create a rule file for the <code>udev</code> function that defined the character special file, and place it in a suitable location.
Other operating systems	All types	Execute the <code>mknod</code> command.	None

■ Block special file

Use a disk partition as a block special file using OS commands (`fdisk`, `parted`, or `mknod`).

For details about the commands and functions, see the OS documentation.

Setting the mode

Set the mode of the created character special file or block special file as follows:

Owner, access permissions		Information to be set
Owner	User ID	HiRDB administrator
	Group ID	Group ID of the HiRDB administrator
Access permissions	Owner	<code>r</code> <code>w</code> (can read/write)
	Group	<code>r</code> <code>w</code> (can read/write)
	Other	<code>--</code> (cannot access)

(c) Setting access permissions**Executor: Superuser**

To prevent unauthorized access from unauthorized users, change the file mode of the created character special file or block special file.

Change the file mode using the `umask` or `chmod` command. Execute the `umask` command prior to creating the character special file or block special file and the `chmod` command after creating the file.

See the OS documentation for details about these commands.

In addition, in the case of Linux 6 or later, separate configuration of `udev` is required, because in the normal configuration it is not possible to secure access privileges to the block special file to the HiRDB administrator. For details about configuring `udev`, see the OS documentation.

For details about access restrictions, see (3) *HiRDB file system area access permissions*.

(d) Creating a HiRDB file system area**Executor: HiRDB administrator**

Execute the `pdfmkfs` command to create a HiRDB file system area in a character special file or block special file.

(e) Setting up symbolic links**Executor: HiRDB administrator**

As the name of the HiRDB file system area, we recommend using a name symbolically linked to the actual name using the OS's `ln` command, rather than using the unaltered actual name of a character special file or block special file.

See the OS documentation for details about the `ln` command.

(3) HiRDB file system area access permissions

This subsection describes Hitachi's approach to setting HiRDB file system area access permissions.

(a) Approach to set values

For greater security, we recommend that HiRDB establish a user group that has the same group ID on the OS as the HiRDB administrator, and that access to it by users outside that group be restricted. Hitachi also recommends only granting read/write access permissions for the HiRDB file system area to the area owner and the group.

For example, in a multi-HiRDB configuration, you can divide up access by HiRDB by setting up groups for each HiRDB. This can prevent unintended access to other HiRDBs.

For details about HiRDB groups, see 2.1.3 *Setting up a HiRDB group*.

(b) Changing access permissions

Notes on changing HiRDB file system area file modes are given below for each access permissions setting.

Access permissions setting (octal notation)	Description and notes
660	Recommended value.
640	Eliminates update permissions for users that have the same group ID as the HiRDB administrator. Operating commands, utilities and the like that allow execution by users other than the HiRDB administrator might not always be executable with user permissions other than those of the HiRDB administrator.
600	Only the HiRDB administrator has access permissions. Operating commands, utilities and the like that allow execution by users other than the HiRDB administrator might not always be executable with user permissions other than those of the HiRDB administrator.
Other	No change.

(c) umask settings

When changing the file mode of the HiRDB file system area, character special file, or block special file, set `umask`, by using the explanation in section (b) as a guide.

The initial value for file mode of the HiRDB file system area created by the `pdfmkfs` command grants the owner and group read/write permissions (660 in octal notation). For details about the initial file mode values of character special files or block special files created by the OS's `mknod` command, see the OS documentation.

(d) Access permissions for HiRDB files

The access permissions of the OS described above are enabled for the HiRDB file system area, but OS access permissions for files are disabled. HiRDB file access permissions cannot be controlled by HiRDB. Accordingly, to restrict access to individual HiRDB files, divide the HiRDB file system area and change the access permissions for each individual HiRDB file system area.

2.4 Uninstallation of HiRDB

Executor: Superuser

Uninstall HiRDB only if you will no longer use HiRDB on the applicable server machine. Otherwise, you should not uninstall HiRDB.

Notes about uninstalling HiRDB

Before you uninstall HiRDB, make sure that all commands, utilities, applications, HiRDB Datareplicator, and HiRDB Dataextractor have stopped. If any of these programs is running, deletion of executable files and shared libraries may fail.

When a HiRDB single server configuration is uninstalled, files and directories required for HiRDB execution that are located in the following directories on the server machine are deleted:

- /opt/HiRDB_S
- All HiRDB directories that the user has registered in the OS using the `pdsetup` command[#]

When a HiRDB parallel server configuration is uninstalled, files and directories required for HiRDB execution that are located in the following directories on that server machine are deleted.

- /opt/HiRDB_P
- All HiRDB directories that the user has registered in the OS using the `pdsetup` command[#]

[#]

When the HiRDB system is deleted from the OS using the `pdsetup -d` command, HiRDB directories for which you responded with `n` to the `KFPS00036-Q` message are also deleted.

The procedure for uninstalling HiRDB is described below. In the case of a HiRDB parallel server configuration, you must uninstall HiRDB from all server machines that constitute the HiRDB parallel server configuration.

To uninstall HiRDB:

1. Stop HiRDB.

Execute the `pdls -d ust` command, and then, if HiRDB is running, use the `pdstop` command to stop HiRDB. If an uninstall is performed while a HiRDB is running, HiRDB processes and OS resources that were being used by HiRDB remain resident. Should that happen, reboot the OS to stop the HiRDB processes and release the OS resources.

2. Delete HiRDB from the OS.

Execute the `pdsetup -d` command, respond with `y` to the `KFPS00036-Q` message, and then delete HiRDB from the OS.

If the termination status of the `pdsetup -d` command is not 0, an error may have occurred. See the message output to the syslog file, eliminate the cause of the error, and then re-execute the removal of HiRDB registration from OS again.

3. Uninstall HiRDB.

Use the Hitachi Program Product Installer to uninstall HiRDB from each server machine.

At this time, check the results in the Hitachi Program Product Installer window. If the uninstallation was not successful, check the message output to the syslog file, eliminate the cause of the error, and then re-execute the uninstallation.

3

Setting Up an Environment Using the Simple Setup Tool

This chapter describes the procedure for using the simple setup tool to set up a HiRDB environment.

3.1 Overview of the simple setup tool

The simple setup tool is not supported.

4

Setting Up an Environment Using Commands

This chapter describes the procedure for setting up the HiRDB environment using commands.

4.1 Overview of environment setup using commands

(1) Items to be defined before environment setup

Before starting the HiRDB environment setup, design the system configuration for the following items:

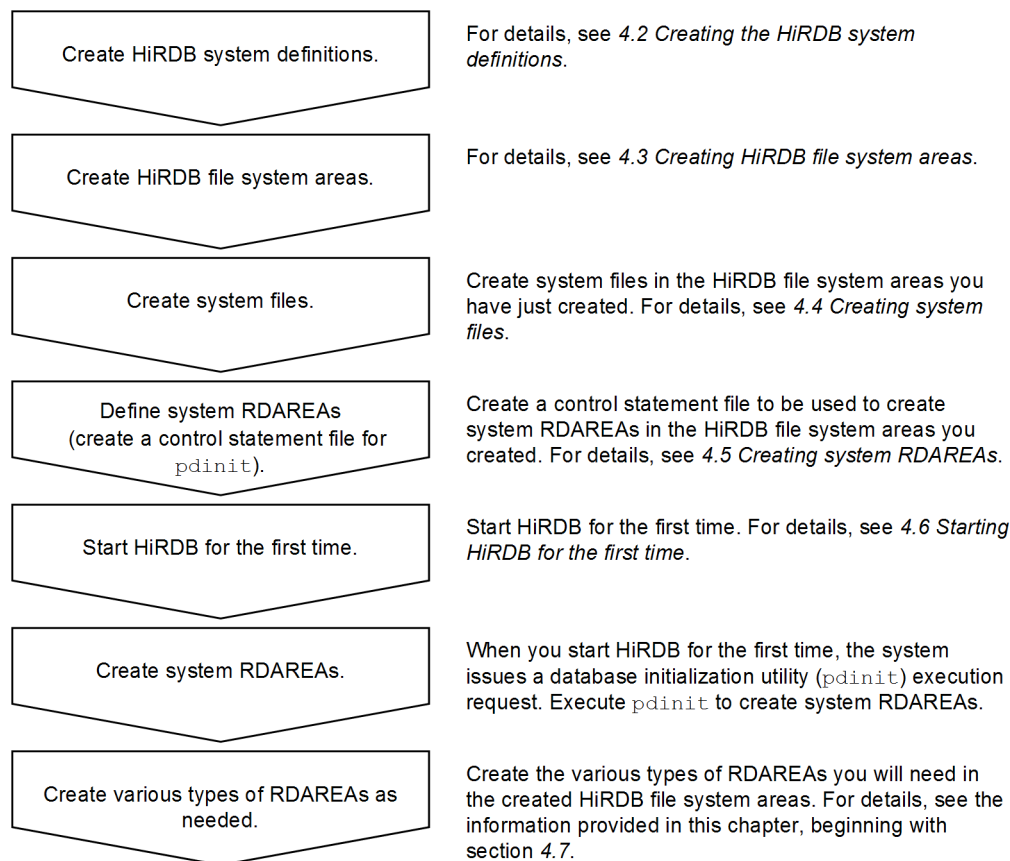
- Units and servers
- HiRDB file system areas
- System files
- Work table files
- RDAREAs

Determine the configuration of these items by referring to *Chapter 8. Designing a HiRDB single server configuration*, or *Chapter 9. Designing a HiRDB parallel server configuration*. After that, set up the HiRDB environment according to the information provided in this chapter, beginning at 4.2 *Creating the HiRDB system definitions*.

(2) Environment setup procedure

The following figure illustrates the procedure for using commands to set up the HiRDB environment.

Figure 4–1: Procedure for using commands to set up the HiRDB environment



You will be defining the following information in this chapter:

- First, use the database-initialization utility (`pdinit`) to create system RDAREAs (master directory RDAREA, data directory RDAREA, and data dictionary RDAREA), so that you can start HiRDB.
- Next, use the database structure modification utility (`pdmod`) to add any required RDAREAs (user RDAREAs, data dictionary RDAREAs, user LOB RDAREAs, and list RDAREAs).

For the user RDAREAs, data dictionary LOB RDAREAs, user LOB RDAREAs, and RDAREAs for lists, you can use the database initialization utility (`pdinit`) to create them together with the system RDAREAs.

4.2 Creating the HiRDB system definitions

Executor: HiRDB administrator

Create HiRDB system definitions according to the designed system configuration and operating environment. This section discusses the following topics:

- Creating HiRDB system definitions
- Sharing HiRDB system definition files (HiRDB parallel server configuration only)
- Modifying HiRDB system definitions
- Modifying UAP environment definitions

For details about the HiRDB system definition operands, see the manual *HiRDB Version 9 System Definition*.

Notes

- After creating HiRDB system definitions, use the `pdconfchk` command to check their conformity. This command checks the definitions required for starting HiRDB for any inconsistencies. For details about the operands supported by the `pdconfchk` command, see the manual *HiRDB Version 9 System Definition*.
- Set and maintain permission to access the HiRDB system definition file so that only the owner of the file (HiRDB administrator) is granted read and write privileges.
- After modifying the HiRDB system definitions, be sure to back up the files under `$PDDIR/conf`. To protect against possible errors on the disk that contains the HiRDB directory, you need to back up the files in the HiRDB directory (files under `$PDDIR/conf`). To restore the HiRDB directory, you need a backup copy of the files under `$PDDIR/conf`.

4.2.1 Creating HiRDB system definitions (HiRDB single server configuration)

(1) Creating system common definitions (HiRDB single server configuration)

For system common definitions, define the HiRDB configuration and common information. Store the system common definitions created in the following file:

- `$PDDIR/conf/pdsys` file

The system common definitions include the definitions of unit configuration, server configuration, and global buffer.

Note that HiRDB commands and utilities operate in conformance with the definitions in this definitions file.

Consequently, grant the read privilege (`r`) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(2) Creating unit control information definitions (HiRDB single server configuration)

For unit control information definitions, define the unit's execution environment. Store the unit control information definitions created in the following file:

- `$PDDIR/conf/pdutsys` file

The unit control information definitions include the definitions of status files for units.

Note that HiRDB commands and utilities operate in conformance with the definitions in this definitions file.

Consequently, grant the read privilege (`r`) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(3) Creating single server definitions

For single server definitions, define the single server's execution environment. Store the created single server definitions in the following file:

- `$PDDIR/conf/server-name#` file

HiRDB commands and utilities operate in conformance with the definitions in this definitions file. Consequently, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

The following shows an example of items that can be specified in the single server definitions:

- System log files
- Synchronization point dump files
- Status files for server
- Work table files

A single server definition is not required for utility special units.

#: Use the server name that is specified in the `-s` option of the `pdstart` operand in the system common definitions. For example, if your specification is `pdstart -s sds1`, then store the single server definitions in the following file:

- `$PDDIR/conf/sds1` file

(4) UAP environment definition creation (optional)

Define UAP execution environments. Create UAP environment definitions as needed, and store them in the following file:

- `$PDDIR/conf/pduapenv/any-name#`

The HiRDB administrator must grant to the users who will use a UAP environment definition the read privilege (r) and the execute privilege (x) for the `$PDDIR/conf/pduapenv` directory. Read (r) privilege must also be granted for the UAP environment definition file.

Also, since HiRDB commands and utilities operate in conformance with the definitions in this definitions file, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

The following are examples of the items that can be specified in a UAP environment definition:

- The action to be taken by the UAP if local buffers are used to access an RDAREA or index, but the RDAREA or index is being used by another user.
- The local buffers to be used by the UAP.

#: The file name must begin with an alphabetic character and must not exceed eight characters in length.

(5) Creating an SQL reserved word definition (optional)

To use the SQL reserved word deletion facility, you must define the reserved words to be deleted for each UAP. You create an SQL reserved word definition as needed and store it in the following file:

- `$PDDIR/conf/pdrsvwd/any-name#`

The HiRDB administrator must grant to a user who uses the SQL reserved word definition the read (r) and execution (x) privileges for the `$PDDIR/conf/pdrsvwd` directory and the read privilege (r) for the SQL reserved word deletion file.

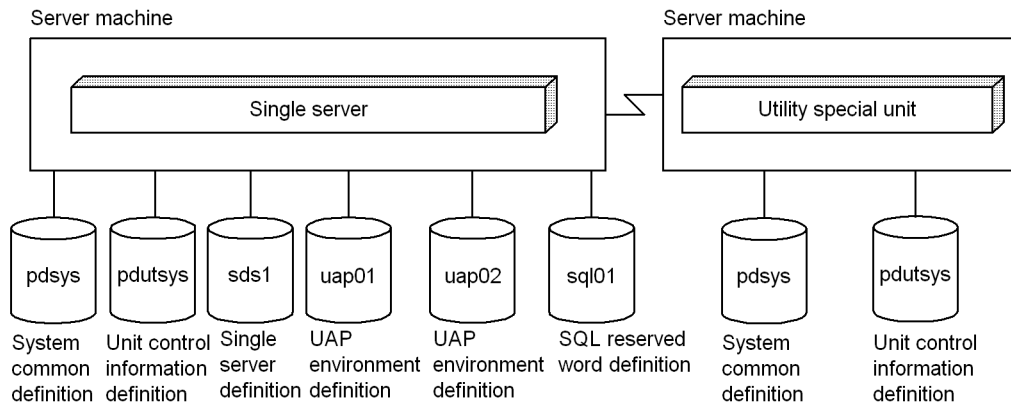
Also, since HiRDB commands and utilities operate in conformance with the definitions in this definitions file, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

#: The file name must begin with an alphabetic character and must not exceed eight alphanumeric characters in length.

(6) Configuring the HiRDB system definition files

The following figure shows an example of a configuration of HiRDB system definition files.

Figure 4–2: Configuration of HiRDB system definition files: HiRDB single server configuration



Note: Single server definition, UAP environment definition, and SQL reserved word definition are not needed for utility special units.

4.2.2 Creating HiRDB system definitions (HiRDB parallel server configuration)

(1) Creating system common definitions (HiRDB parallel server configuration)

For system common definitions, define the HiRDB configuration and common information. After you create the system common definitions, store them in the following file:

- `$PDDIR/conf/pdsys` file

Create the same system common definitions for each server machine.

The system common definitions include the definitions of unit configuration, server configuration, and global buffer.

Note that HiRDB commands and utilities operate in conformance with the definitions in this definitions file.

Consequently, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(2) Creating unit control information definitions (HiRDB single server configuration)

For unit control information definitions, define the unit's execution environment. Store the unit control information definitions created in the following file:

- `$PDDIR/conf/pdutsys` file

Create the unit control information definitions for each server machine.

The unit control information definitions include the definitions of status files for units.

Note that HiRDB commands and utilities operate in conformance with the definitions in this definitions file.

Consequently, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(3) Creating single server definitions

For single server definitions, define the default values of the server-definition operands, which are explained in (4)-(6), below. Create the server common definitions for each server as required and store them in the following file:

- `$PDDIR/conf/pdsvrc` file

The server common definitions are useful in the following cases:

- There are many servers to be defined per server machine.
- There are many definitions that are common to multiple servers.

The information specified in the server common definitions takes effect on all the servers defined in the corresponding server machine. If there are many definitions that are common to multiple servers, you should specify the common information in the server common definitions and the information unique to individual servers in the corresponding server definitions.

If you are sharing HiRDB system definition files, you should create the server common definitions.

Also, since HiRDB commands and utilities operate in conformance with the definitions in this definitions file, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(4) Creating front-end server definitions

For front-end server definitions, define the front-end server's execution environment. Save the front-end server definitions created in the following file:

- `$PDDIR/conf/server-name#` file

Create the front-end server definitions in the server machine where the front-end server is defined. The following shows an example of items that can be specified in the front-end server definitions:

- System log files for the front-end server
- Synchronization point dump files for the front-end server
- Status files for the front-end server

#: Use the server name that is specified in the `-s` option of the `pdstart` operand in the system common definitions. For example, if your specification is `pdstart -s f001`, then store the front-end server definitions in the following file:

- `$PDDIR/conf/f001` file

Note that HiRDB commands and utilities operate in conformance with the definitions in this definitions file. Consequently, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(5) Creating dictionary server definitions

For dictionary server definitions, define the dictionary server's execution environment. Save the dictionary server definitions created in the following file:

- `$PDDIR/conf/server-name#` file

Create the dictionary server definitions in the dictionary server machine where the dictionary server is defined. The following shows an example of items that can be specified in the dictionary server definitions:

- System log files for the dictionary server
- Synchronization point dump files for the dictionary server
- Status files for the dictionary server
- Work table files

#: Use the server name that is specified in the `-s` option of the `pdstart` operand in the system common definitions. For example, if your specification is `pdstart -s dic`, then store the dictionary server definitions in the following file:

- `$PDDIR/conf/dic` file

Note that HiRDB commands and utilities operate in conformance with the definitions in this definitions file. Consequently, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(6) Creating back-end server definitions

For back-end server definitions, define the back-end server's execution environment. Save the back-end server definitions created in the following file:

- `$PDDIR/conf/server-name#` file

Create the back-end server definitions in the server machine where the back-end server is defined. The following shows an example of items that can be specified in the back-end server definitions:

- System log files for the back-end server
- Synchronization point dump files for the back-end server
- Status files for the back-end server
- Work table files

#: Use the server name that is specified in the `-s` option of the `pdstart` operand in the system common definitions. For example, if your specification is `pdstart -s b001`, then store the back-end server definitions in the following file:

- `$PDDIR/conf/b001` file

Note that HiRDB commands and utilities operate in conformance with the definitions in this definitions file. Consequently, grant the read privilege (`r`) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

(7) UAP environment definition creation (optional)

Define UAP execution environments. Create UAP environment definitions as needed, and store them in the following file:

- `$PDDIR/conf/pduapenv/any-name#`

A UAP environment definition is created at the unit with the front-end server. If there are multiple front-end servers, the UAP environment definition can be defined at any of them, as appropriate.

The HiRDB administrator must grant to the users who will use a UAP environment definition the read privilege (`r`) and the execute privilege (`x`) for the `$PDDIR/conf/pduapenv` directory. Read (`r`) privilege must also be granted for the UAP environment definition file.

Also, since HiRDB commands and utilities operate in conformance with the definitions in this definitions file, grant the read privilege (`r`) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

The following are examples of the items that can be specified in a UAP environment definition:

- The action to be taken by the UAP if local buffers are used to access an RDAREA or index, but the RDAREA or index is being used by another user.
- The local buffers to be used by the UAP.

#: The file name must begin with an alphabetic character and must not exceed eight characters in length.

(8) Creating an SQL reserved word definition (optional)

To use the SQL reserved word deletion facility, you must define the reserved words to be deleted for each UAP. You create an SQL reserved word definition as needed and store it in the following file:

- `$PDDIR/conf/pdrsvwd/any-name#`

Create the SQL reserved word definition on the unit where the front-end server is located. In the event of multiple front-end servers, define the SQL reserved word definition on the front-end server to which the UAP environment definition is to be applied.

The HiRDB administrator must grant to a user who uses the SQL reserved word definition the read (`r`) and execution (`x`) privileges for the `$PDDIR/conf/pdrsvwd` directory and the read privilege (`r`) for the SQL reserved word deletion file.

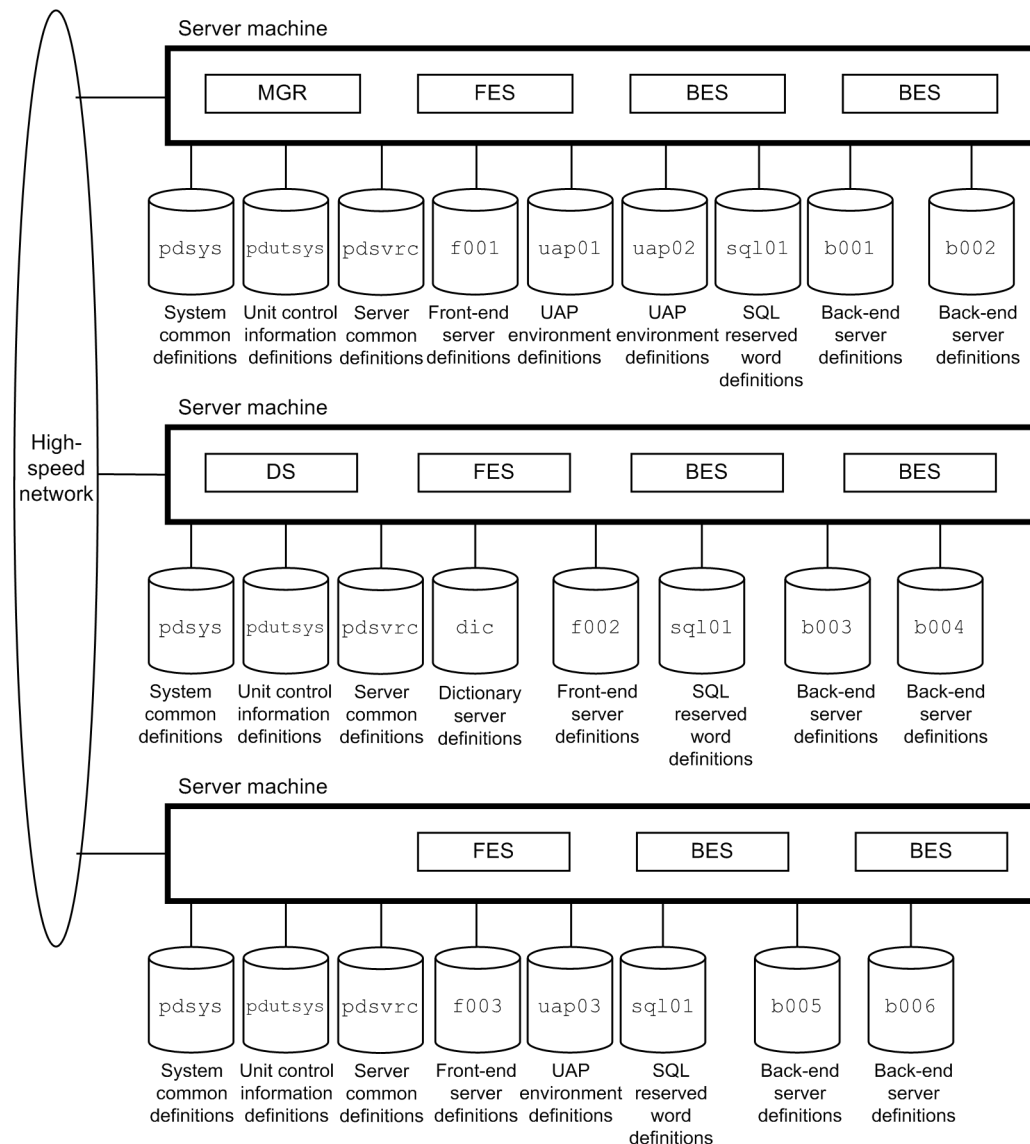
Also, since HiRDB commands and utilities operate in conformance with the definitions in this definitions file, grant the read privilege (r) for this definitions file to users who execute HiRDB commands or utilities (users on the OS).

#: The file name must begin with an alphabetic character and must not exceed eight alphanumeric characters in length.

(9) Configuring the HiRDB system definition files

The following figure shows an example of a configuration of HiRDB system definition files.

Figure 4-3: Configuration of HiRDB system definition files: HiRDB parallel server configuration



Note 1: The system common definitions must be identical on all server machines.

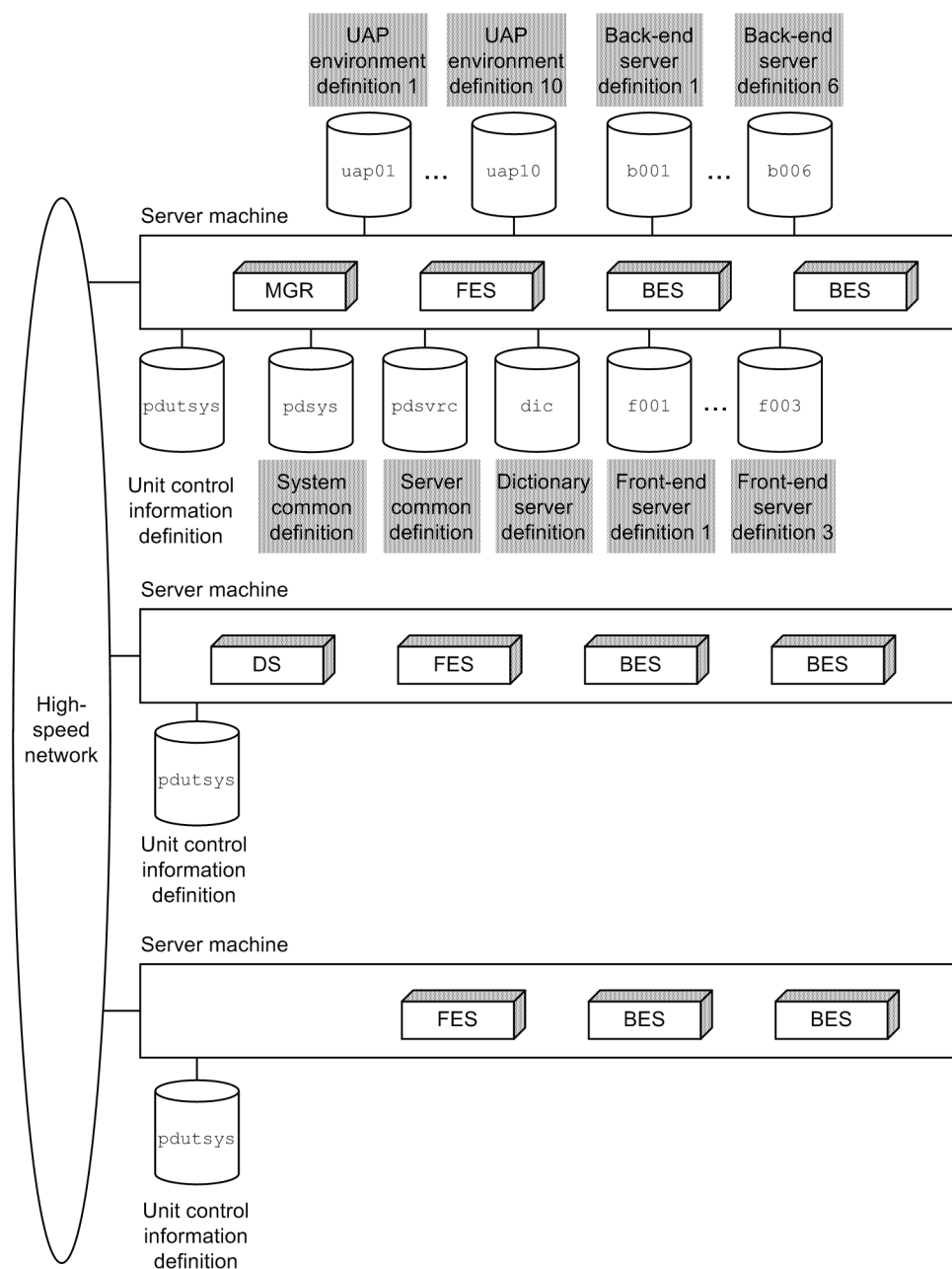
Note 2: The SQL reserved word definitions must be identical on all server machines.

4.2.3 Sharing HiRDB system definition files (HiRDB parallel server configuration)

For a HiRDB parallel server configuration, the HiRDB administrator must create HiRDB system definition files and manage them for each server machine. However, if the file sharing facility (NFS) is used, all the HiRDB system definition files, except for unit control information definition files, can be managed by a single server machine. This is

called sharing HiRDB system definition files. The following figure illustrates sharing of HiRDB system definition files.

Figure 4–4: Sharing HiRDB system definition files



Note

Shading indicates the shared files.

(1) Differences depending on whether definitions are shared

When not shared	When shared
HiRDB system definitions must be managed for each server machine.	HiRDB system definitions (except for unit control information definitions) can be managed by one server machine.

When not shared	When shared
<p>When the system common definition is modified, the same modifications must be made as many times as there are server machines.</p> <p>For example, if there are four server machines, there are four system common definitions. Because the contents of these four system common definitions must be identical, all four definitions must be modified.</p>	<ul style="list-style-type: none"> • Because there is only one system common definition, necessary modifications need to be made only once. • Because there are fewer modifications to be made, the chances of making errors are reduced significantly. • If HiRDB can no longer reference the shared directory, HiRDB may terminate abnormally.

(2) How to share HiRDB system definition files

To share HiRDB system definition files, do the following:

1. Create a directory to store the HiRDB system definition files that are to be shared. Create this directory at the server machine that is to manage the HiRDB system definition files that are to be shared. This directory is called the shared directory. These definition files can be managed by any server machine, but it is recommended that they be managed by the server machine where the system manager is defined.
2. Create the HiRDB system definition files (except for unit control information definition files) under the shared directory.
3. Create the unit control information definition files under `$PDDIR/conf/` at each server machine. In this case, specify the name of the shared directory in the `PDCONFPATH` operand.

4.2.4 Modifying HiRDB system definitions (excluding UAP environment definitions)

This section describes how to modify HiRDB system definitions.

Note

- After modifying the HiRDB system definitions, be sure to back up the files under `$PDDIR/conf`. To protect against possible errors on the disk that contains the HiRDB directory, you need to back up the files in the HiRDB directory (files under `$PDDIR/conf`). To restore the HiRDB directory, you need a backup copy of the files under `$PDDIR/conf`. If `$PDCONFPATH` is under the HiRDB directory, back it up in the same manner.
- For a HiRDB parallel server configuration, create subdirectories for each unit under `$PDDIR/conf` and `$PDCONFPATH`, and check the contents of the HiRDB system definition.

(1) How to modify HiRDB system definitions

This section describes how to modify HiRDB system definitions. In this explanation, the directory that stores the unit control information definition file is referred to as `$PDDIR/conf`, and the directory that stores any other HiRDB system definition files is referred to as `$PDCONFPATH`.

Procedure

1. Create subdirectories under `$PDDIR/conf` and `$PDCONFPATH`. In this example, the subdirectories will be named `work`.
2. Copy the unit control information definition file under `$PDDIR/conf/work`. Copy the other HiRDB system definition files under `$PDCONFPATH/work`.
3. Modify the HiRDB system definitions copied into `$PDDIR/conf/work` and `$PDCONFPATH/work`.
4. Use the `pdconfchk -d work` command to check the contents of the HiRDB system definitions in `$PDDIR/conf/work` and `$PDCONFPATH/work`. If an error is detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
5. Use the `pdstop` command to terminate HiRDB normally.
6. Use the `pdlogunld` command to unload system log files in unload wait status.

7. Replace the HiRDB system definition files by copying the HiRDB system definition files modified in step 3 under `$PDDIR/conf` and `$PDCONFPATH`.
8. If the values specified for the following operands have been modified, use the `pdloginit` command to initialize the system log files:
 - `pd_log_dual`
 - `pdstart`
9. Use the `pdstart` command to perform a HiRDB normal startup.

(2) How to modify HiRDB system definitions with the system reconfiguration command

When the system reconfiguration command (`pdchgconf` command) is used, the HiRDB system definitions can be modified while HiRDB is operating, which means that HiRDB need not be terminated. However, in order to use this command, HiRDB Advanced High Availability must be installed. The following section shows how to modify a HiRDB system definition with the system reconfiguration command.

Procedure

1. Create the `$PDDIR/conf/chgconf` directory.
2. Copy the HiRDB system definition files currently being used to the directory created in step 1.
3. Modify the HiRDB system definitions in `$PDDIR/conf/chgconf`.
4. Use the `pdconfchk` command to perform a check of the HiRDB system definitions in `$PDDIR/conf/chgconf`. If an error is detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
5. Use the `pdchgconf` command to replace the HiRDB system definitions with the modified HiRDB system definitions.

When the `pdchgconf` command is executed, the HiRDB system definition files currently being used (before modification) will be saved in `$PDDIR/conf/backconf`. Therefore, the modified HiRDB system definition files in `$PDDIR/conf/chgconf` will be copied to `$PDDIR/conf`.

Notes

- If a transaction or utility is still operating 15 minutes after the `pdchgconf` command was entered, the `pdchgconf` command terminates abnormally.
- There are restrictions on the use of the system reconfiguration command to modify HiRDB system definitions; for details about the restrictions, see the *HiRDB Version 9 System Operation Guide*.

(3) Notes

- When a system common definition is modified, the same modification must be made to the system common definitions for all the server machines (applicable to a HiRDB parallel server configuration).
- HiRDB system definitions that are being used by an active HiRDB must not be modified or deleted. If they are modified or deleted, operation of the HiRDB cannot be guaranteed.
- In the event of HiRDB planned termination, forced termination, or abnormal termination, some items can be modified using the HiRDB system definition operands, but some items cannot be modified in this manner. For details, see the manual *HiRDB Version 9 System Definition*.

4.2.5 Modifying a UAP environment definition

This section describes how to modify a UAP environment definition.

Procedure

1. Check that the UAP that uses the UAP environment definition is not executing. If the UAP is executing when the UAP environment definition is modified, the executing UAP will usually use the UAP environment definition before modification; however, depending on the timing, the modified UAP environment definition may be used.
2. Modify the UAP environment definition.

3. Execute the UAP using the modified UAP environment definition.

4.3 Creating HiRDB file system areas

Executor: HiRDB administrator

Use the `pdfmkfs` command to create an area where you can create HiRDB files (*HiRDB file system area*). A regular file or character special file is used to create a HiRDB file system area.

Files in a file system that accesses files over a network such as NFS cannot be used as HiRDB file system areas.

4.3.1 Types of HiRDB file system areas

Create a different HiRDB file system area for each purpose, as shown in the following table. Use the `pdfmkfs` command's `-k` option to specify the purpose.

Table 4–1: Types of HiRDB file system areas

NO.	Type of HiRDB file system area	-k option value
1	RDAREAs	DB
2	Shared RDAREA	SDB
3	System files	SYS
4	Work table files	WORK
5	Utilities	UTL
6	RDAREAs for lists	WORK

Operating HiRDB requires HiRDB file system areas 1, 3 and 4.

For details about how to design a HiRDB file system area, see *8.2 Designing HiRDB file system areas* for a HiRDB single server configuration, and *9.2 Designing HiRDB file system areas* for a HiRDB parallel server configuration.

Note

The size of a HiRDB file system area that is to be initialized must not exceed the partition size. If it exceeds the partition size, the next partition physically following that partition may be damaged.

4.3.2 Using character special files

(1) Range of applicability of character special files

The utility files that can be created in character special files when UTL is specified in the `pdfmkfs` command's `-k` option (the purpose of the HiRDB file system area) are as follows.

- Backup files
- Unload log files
- Unload data files
- Differential backup management files
- Index information files

(2) Initial setting

If character special files are to be used, you must use the `pdfmkfs` command to make an initial setting that the HiRDB file system area will use character special files. If the character special files are symbolic links, specify the names of the links.

Before you specify the initial setting, modify the owner and access privileges of the file system area. For details about these modifications, see *2.3.7(2) Creating a HiRDB file system area in a character special file or block special file*.

The following is an example of specifying the `pdfmkfs` command:

```
pdfmkfs -n 200 -l 20 -k DB /dev/raw/raw1
```

4.3.3 Creating a large file

The maximum size of a HiRDB file system area is 2,047 MB (about 2 GB). To create a HiRDB file system area that is larger than this size, you need to create the HiRDB file system area as a large file. If it is created as a large file, the maximum size of the HiRDB file system area is 1,048,575 MB.

- Method for creating a large file

You can create a large file in the same manner as with a HiRDB file system area. Use the `pdfmkfs` command's `-n` option to specify the size of the HiRDB file system area that is to be a large file (2 gigabytes or greater).

- Files to be used with utilities

The following utilities may not support large files:

- Database load utility (`pdload`)
- Database reorganization utility (`pdrorg`)

See the manual *HiRDB Version 9 Command Reference* to determine whether these utilities support large files.

4.3.4 Example 1 (creating a HiRDB file system area for RDAREAs)

The following shows an example of creating a HiRDB file system area for RDAREAs:

Example

Create a HiRDB file system area for RDAREAs:

```
pdfmkfs -n 50 -l 10 -k DB -i /dbarea01
```

Explanation:

- n: Specifies the size of the HiRDB file system area in MB.
- l: Specifies the maximum number of files that can be created in this HiRDB file system area.
- k: Specifies the purpose of this HiRDB file system area.

This example specifies `DB` because this HiRDB file system area is for RDAREAs.

- i: Specifies that the entire HiRDB file system area is to be initialized.

When the `-i` option is specified, the system allocates the entire area. When the `-i` option is omitted, the system creates only the management information for the HiRDB file system area.

`/dbarea01`: Specifies the name of the HiRDB file system area to be created.

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

4.3.5 Example 2 (creating a HiRDB file system area for system files)

The following shows an example of creating a HiRDB file system area for system files:

Example

Create a HiRDB file system area for system files:

```
pdfmkfs -n 50 -l 20 -k SYS -i /sysarea01
```

Explanation:

- n: Specifies the size of the HiRDB file system area in MB.
- l: Specifies the maximum number of files that can be created in this HiRDB file system area.

-k: Specifies the purpose of this HiRDB file system area.

This example specifies `SYS` because this HiRDB file system area is for system files.

-i: Specifies that the entire HiRDB file system area is to be initialized.

When the `-i` option is specified, the system allocates the entire area. When the `-i` option is omitted, the system creates only the management information for the HiRDB file system area.

`/sysarea01:` Specifies the name of the HiRDB file system area to be created.

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

4.3.6 Example 3 (creating a HiRDB file system area for work table files)

The following shows an example of creating a HiRDB file system area for work table files:

Example

Create a HiRDB file system area for work table files:

```
pdfmkfs -n 50 -l 20 -k WORK -e 3300 -i -a /workarea01
```

Explanation:

-n: Specifies the size of the HiRDB file system area in MB.

For details about how to estimate the area size, see *Chapter 18. Determining Work Table File Size*.

-l: Specifies the maximum number of files that can be created in this HiRDB file system area.

-k: Specifies the purpose of this HiRDB file system area.

This example specifies `WORK` because this HiRDB file system area is for work table files.

-e: Specifies the number of HiRDB file extensions permitted for this HiRDB file system area.

-i: Specifies that the entire HiRDB file system area is to be initialized.

When the `-i` option is specified, the system allocates the entire area. When the `-i` option is omitted, the system creates only the management information for the HiRDB file system area.

-a: Specifies that the HiRDB file system area is to be extended automatically.

Specify the `-a` option to automatically extend the HiRDB file system area as much as necessary even if it exceeds the size specified with the `-n` option when, for example, an SQL statement that uses automatic extension of an RDAREA or a work table is executed.

`/workarea01:` Specifies the name of the HiRDB file system area to be created.

Enter the name that was specified in the `pdwork` operand in the HiRDB system definitions.

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

4.3.7 Example 4 (creating a HiRDB file system area for utilities)

This section shows an example of creating a HiRDB file system area for utilities. The following files are created in the HiRDB file system area for utilities:

- Backup files
- Unload data files
- Unload log files
- Differential backup management files
- Index information files

Example

Create a HiRDB file system area for utilities:

```
pdfmkfs -n 50 -l 10 -k UTL -i /utlarea01
```

Explanation:

- n: Specifies the size of the HiRDB file system area in MB.
- l: Specifies the maximum number of files that can be created in this HiRDB file system area.
- k: Specifies the purpose of this HiRDB file system area.

This example specifies UTL because this HiRDB file system area is for utilities.

- i: Specifies that the entire HiRDB file system area is to be initialized.

When the -i option is specified, the system allocates the entire area. When the -i option is omitted, the system creates only the management information for the HiRDB file system area.

/utlarea01: Specifies the name of the HiRDB file system area to be created.

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

4.3.8 Example 5 (creating a HiRDB file system area for list RDAREAs)

The following shows an example of creating a HiRDB file system area for list RDAREAs:

Example

Create a HiRDB file system area for list RDAREAs:

```
pdfmkfs -n 50 -l 10 -k WORK -i /listarea01
```

Explanation:

- n: Specifies the size of the HiRDB file system area in MB.
- l: Specifies the maximum number of files that can be created in this HiRDB file system area.
- k: Specifies the purpose of this HiRDB file system area.

This example specifies WORK because this HiRDB file system area is for RDAREAs for lists.

- i: Specifies that the entire HiRDB file system area is to be initialized.

When the -i option is specified, the system allocates the entire area. When the -i option is omitted, the system creates only the management information for the HiRDB file system area.

/listarea01: Specifies the name of the HiRDB file system area to be created.

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

4.4 Creating system files

The HiRDB administrator creates system files in HiRDB file system areas as explained in 4.3 *Creating HiRDB file system areas*. There are three types of system files:

- System log files
- Synchronization point dump files
- Status files

For details about how to design system files, see 8.3 *Designing system files* for a HiRDB single server configuration and 9.3 *Designing system files* for a HiRDB parallel server configuration.

4.4.1 Creating system log files

The HiRDB administrator executes the `pdloginit` command to create system log files in a HiRDB file system area.

Example

Create system log files (`log01`) in a HiRDB file system area (`/sysarea01`):

```
pdloginit -d sys -s b001 -f /sysarea01/log01 -n 1024
```

Explanation:

- d `sys`: Specifies that this is a system log file.
- s: Specifies the name of the server corresponding to this system log file.
This specification is not necessary for a HiRDB single server configuration.
- f: Specifies a name for this system log file.
Enter the name that was specified with the `pdlogadpf -d sys` server definition operand in the HiRDB system definitions.
- n: Specifies the number of records for this system log file.
The size of one system log file equals the record length times the number of records (bytes). A system log file normally has a record length of 1,024 bytes, but if the `pd_log_rec_leng` operand is specified, the value of the `pd_log_rec_leng` operand is used.

After the command has executed, we recommend that you check the execution results for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

Relationship with HiRDB system definitions

This `pdloginit` command is associated with the following server definition operands in the HiRDB system definitions:

- `pdlogadfg -d sys`
- `pdlogadpf -d sys`

You need to define the system log file created using these operands.

4.4.2 Creating synchronization point dump files

The HiRDB administrator executes the `pdloginit` command to create synchronization point dump files in a HiRDB file system area.

Example

Create a synchronization point dump file (`b1sync01`) in a HiRDB file system area (`/sysarea01`):

```
pdloginit -d spd -s b001 -f /sysarea01/sync01 -n 64
```

Explanation:

- d `spd`: Specifies that this is a synchronization point dump file.

- s: Specifies the name of the server corresponding to this synchronization point dump file.
This specification is not necessary for a HiRDB single server configuration.
- f: Specifies a name for this synchronization point dump file.
Enter the name that was specified with the `pdlogadpf -d spd` server definition operand in the HiRDB system definitions.
- n: Specifies the number of records for this synchronization point dump file.
The size of one synchronization point dump file equals 4,096 times the record length (bytes).

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

Relationship with HiRDB system definitions

This `pdloginit` command is associated with the following server definition operands in the HiRDB system definitions:

- `pdlogadfg -d spd`
- `pdlogadpf -d spd`

You need to define the synchronization point dump file created using these operands.

4.4.3 Creating status files

The HiRDB administrator uses the `pdstsinit` command to create status files in a HiRDB file system area. The HiRDB administrator must create status files for both the unit and server.

Example

Create a server status file (`b1sts01a`) in a HiRDB file system area (`/sysarea01`):

```
pdstsinit -s b001 -f /sysarea01/sts01 -l 4096 -c 256
```

Explanation:

- s: Specifies the name of the server corresponding to this server status file.
- f: Specifies a name for this server status file.
Enter the name that was specified with the `pd_sts_file_name` server definition operand in the HiRDB system definitions.
- l: Specifies the record length for this status file.
- c: Specifies the number of records for this status file.
The size of one status file is equal to the record length times the number of records (bytes).

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

Relationship with HiRDB system definitions

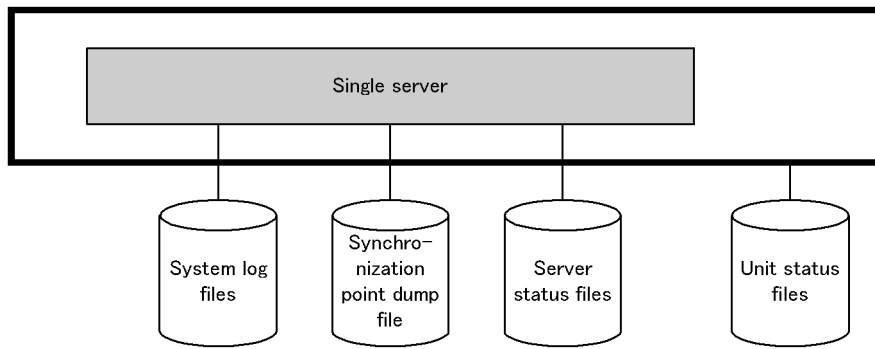
This `pdstsinit` command is associated with the following operands in the HiRDB system definitions:

- `pd_syssts_file_name` (unit status file)
- `pd_sts_file_name` (server status file)

You need to define the status file created using these operands.

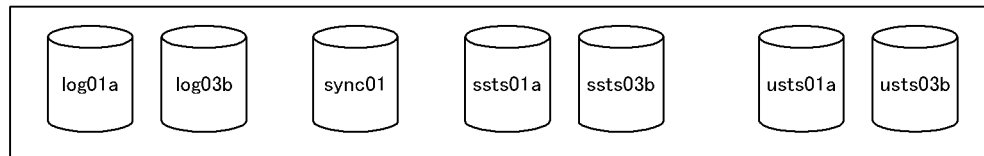
4.4.4 Example of system file creation (HiRDB single server configuration)

This section explains the system file creation procedure by way of example, based on the following system configuration:

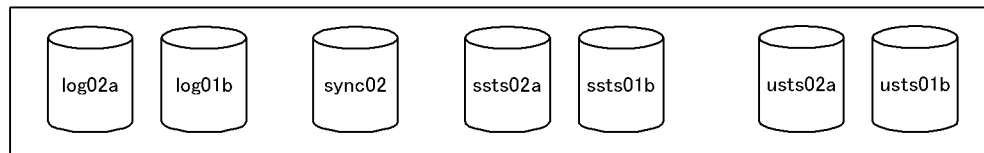


Configuration of the HiRDB file system area

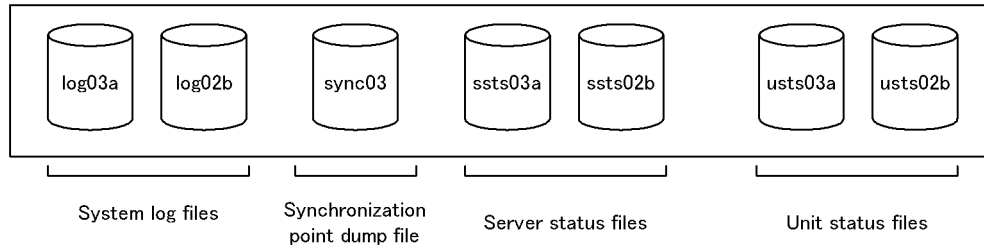
HiRDB file system area (/sysarea01)



HiRDB file system area (/sysarea02)



HiRDB file system area (/sysarea03)



(1) Defining system files (specifying HiRDB system definitions)

Define the system files in the HiRDB system definitions.

(a) Unit control information definitions (for unit status files)

Define the unit status files in the unit control information definitions.

Definition example

```
set pd_syssts_file_name_1="usts1", "/sysarea01/usts01a" \
                        , "/sysarea02/usts01b" \
set pd_syssts_file_name_2="usts2", "/sysarea02/usts02a" \
                        , "/sysarea03/usts02b" \
set pd_syssts_file_name_3="usts3", "/sysarea03/usts03a" \
                        , "/sysarea01/usts03b"
```

(b) Single server definitions

Define system log files, the synchronization point dump file, and server status files in the single server definitions.

Definition example of system log files

```
pdlogadfg -d sys -g log1 ONL
pdlogadfg -d sys -g log2 ONL
pdlogadfg -d sys -g log3 ONL
pdlogadpf -d sys -g log1 -a "/sysarea01/log01a"\
-b "/sysarea02/log01b"
pdlogadpf -d sys -g log2 -a "/sysarea02/log02a"\
-b "/sysarea03/log02b"
pdlogadpf -d sys -g log3 -a "/sysarea03/log03a"\
-b "/sysarea01/log03b"
```

Definition example of synchronization point dump files

```
pdlogadfg -d spd -g sync1 ONL
pdlogadfg -d spd -g sync2 ONL
pdlogadfg -d spd -g sync3 ONL
pdlogadpf -d spd -g sync1 -a "/sysarea01/sync01"
pdlogadpf -d spd -g sync2 -a "/sysarea02/sync02"
pdlogadpf -d spd -g sync3 -a "/sysarea03/sync03"
```

Definition example of server status files

```
set pd_sts_file_name_1="sst1", "/sysarea01/ssts01a"\
, "/sysarea02/ssts01b"
set pd_sts_file_name_2="sst2", "/sysarea02/ssts02a"\
, "/sysarea03/ssts02b"
set pd_sts_file_name_3="sst3", "/sysarea03/ssts03a"\
, "/sysarea01/ssts03b"
```

(2) Creating the HiRDB file system areas

Use the `pdfmkfs` command to create the HiRDB file system areas.

Example of command entry

```
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea01
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea02
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea03
```

(3) Creating the system files**(a) Creating the system log files**

Use the `pdloginit` command to create the system log files.

Example of command entry

```
pdloginit -d sys -f /sysarea01/log01a -n 1024
pdloginit -d sys -f /sysarea01/log03b -n 1024
pdloginit -d sys -f /sysarea02/log02a -n 1024
pdloginit -d sys -f /sysarea02/log01b -n 1024
pdloginit -d sys -f /sysarea03/log03a -n 1024
pdloginit -d sys -f /sysarea03/log02b -n 1024
```

(b) Creating the synchronization point dump file

Use the `pdloginit` command to create the synchronization point dump file.

Example of command entry

```
pdloginit -d spd -f /sysarea01/sync01 -n 64
pdloginit -d spd -f /sysarea02/sync02 -n 64
pdloginit -d spd -f /sysarea03/sync03 -n 64
```

(c) Creating the server status files

Use the `pdstsinit` command to create the server status files.

Example of command entry

```
pdstsinit -s sds1 -f /sysarea01/ssts01a -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea01/ssts03b -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea02/ssts02a -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea02/ssts01b -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea03/ssts03a -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea03/ssts02b -l 4096 -c 256
```

(d) Creating the unit status files

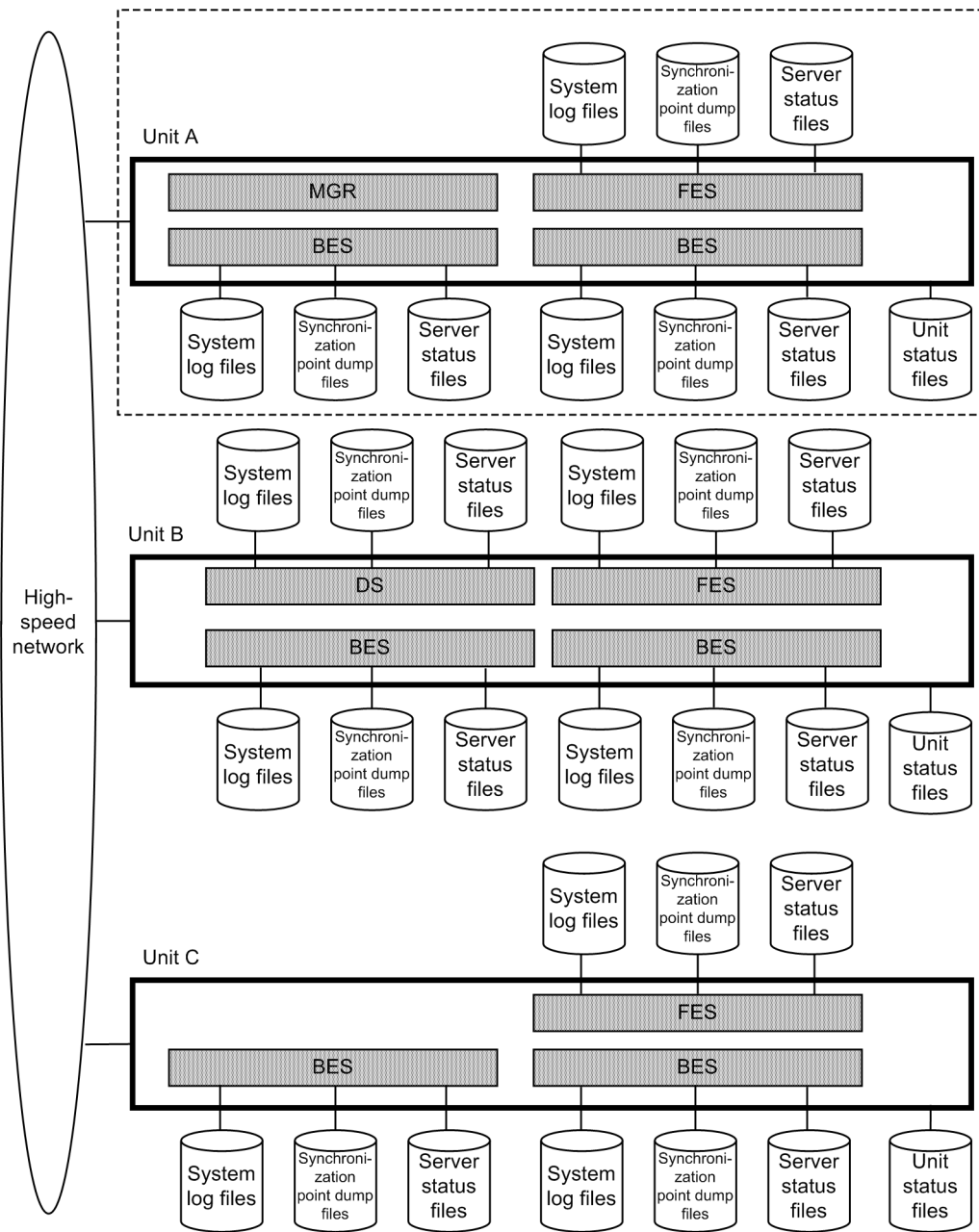
Use the `pdstsinit` command to create the unit status files.

Example of command entry

```
pdstsinit -u unt1 -f /sysarea01/usts01a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea01/usts03b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/usts02a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/usts01b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/usts03a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/usts02b -l 4096 -c 256
```

4.4.5 Example of system file creation (HiRDB parallel server configuration)

This section explains the system file creation procedure by way of example, based on the following system configuration:



MGR: System manager

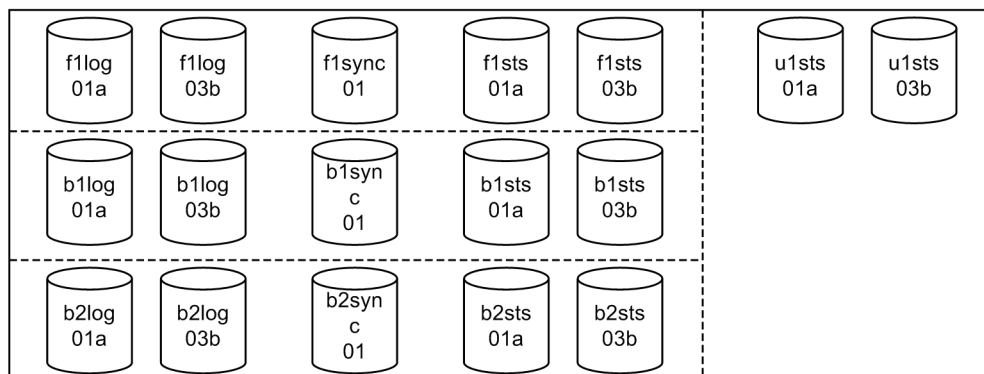
FES: Front-end server

DS: Dictionary server

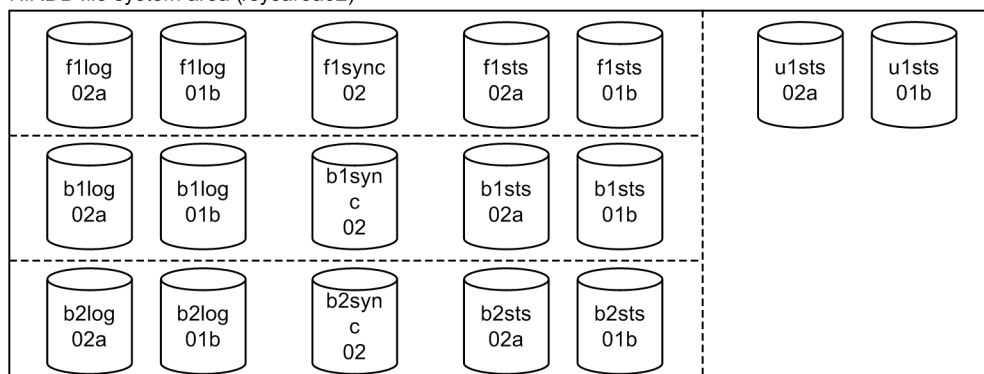
BES: Back-end server

Configuration of the HiRDB file system area

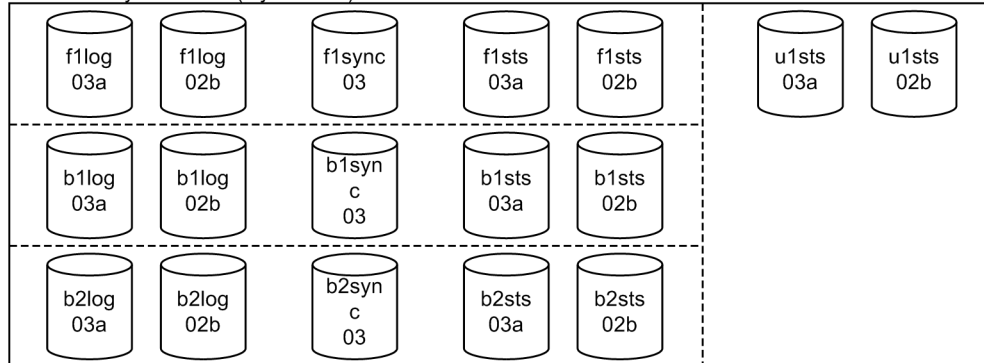
HiRDB file system area (/sysarea01)



HiRDB file system area (/sysarea02)



HiRDB file system area (/sysarea03)



System log files

Synchronization
point dump files

Server status files

Unit status files

Explanation

This is a sample configuration of HiRDB file system areas for unit A. The following examples are all based on the system file creation for unit A.

(1) Defining system files (specifying HiRDB system definitions)

Define the system files in the HiRDB system definitions.

(a) Unit control information definitions (for unit status files)

Define the unit status files in the unit control information definitions.

Definition example

```
set pd_syssts_file_name_1="ulsts1","/sysarea01/ulsts01a"\
, "/sysarea02/ulsts01b"
set pd_syssts_file_name_2="ulsts2","/sysarea02/ulsts02a"\
, "/sysarea03/ulsts02b"
set pd_syssts_file_name_3="ulsts3","/sysarea03/ulsts03a"\
, "/sysarea01/ulsts03b"
```

(b) FES1 front-end server definitions

Define system log files, the synchronization point dump files, and server status files in the FES1 front-end server definitions.

Definition example of system log files

```
pdlogadfg -d sys -g fllog1 ONL
pdlogadfg -d sys -g fllog2 ONL
pdlogadfg -d sys -g fllog3 ONL
pdlogadpf -d sys -g fllog1 -a "/sysarea01/fllog01a"\
-b "/sysarea02/fllog01b"
pdlogadpf -d sys -g fllog2 -a "/sysarea02/fllog02a"\
-b "/sysarea03/fllog02b"
pdlogadpf -d sys -g fllog3 -a "/sysarea03/fllog03a"\
-b "/sysarea01/fllog03b"
```

Definition example of synchronization point dump files

```
pdlogadfg -d spd -g flsync1 ONL
pdlogadfg -d spd -g flsync2 ONL
pdlogadfg -d spd -g flsync3 ONL
pdlogadpf -d spd -g flsync1 -a "/sysarea01/flsync01"
pdlogadpf -d spd -g flsync2 -a "/sysarea02/flsync02"
pdlogadpf -d spd -g flsync3 -a "/sysarea03/flsync03"
```

Definition example of server status files

```
set pd_sts_file_name_1="flsts1","/sysarea01/flsts01a"\
, "/sysarea02/flsts01b"
set pd_sts_file_name_2="flsts2","/sysarea02/flsts02a"\
, "/sysarea03/flsts02b"
set pd_sts_file_name_3="flsts3","/sysarea03/flsts03a"\
, "/sysarea01/flsts03b"
```

(c) BES1 back-end server definitions

Define system log files, synchronization point dump files, and server status files in the BES1 back-end server definitions.

Definition example of system log files

```
pdlogadfg -d sys -g bllog1 ONL
pdlogadfg -d sys -g bllog2 ONL
pdlogadfg -d sys -g bllog3 ONL
pdlogadpf -d sys -g bllog1 -a "/sysarea01/bllog01a"\
-b "/sysarea02/bllog01b"
pdlogadpf -d sys -g bllog2 -a "/sysarea02/bllog02a"\
-b "/sysarea03/bllog02b"
pdlogadpf -d sys -g bllog3 -a "/sysarea03/bllog03a"\
-b "/sysarea01/bllog03b"
```

Definition example of synchronization point dump files

```
pdlogadfg -d spd -g blsync1 ONL
pdlogadfg -d spd -g blsync2 ONL
pdlogadfg -d spd -g blsync3 ONL
pdlogadpf -d spd -g blsync1 -a "/sysarea01/blsync01"
pdlogadpf -d spd -g blsync2 -a "/sysarea02/blsync02"
pdlogadpf -d spd -g blsync3 -a "/sysarea03/blsync03"
```

Definition example of server status files

```
set pd_sts_file_name_1="blsts1","/sysarea01/blsts01a"\
, "/sysarea02/blsts01b"
set pd_sts_file_name_2="blsts2","/sysarea02/blsts02a"
```

4. Setting Up an Environment Using Commands

```
set pd_sts_file_name_3="b1sts3", "/sysarea03/b1sts02b"
, "/sysarea03/b1sts03a" \
, "/sysarea01/b1sts03b"
```

(d) BES2 back-end server definitions

Define system log files, synchronization point dump files, and server status files in the BES2 back-end server definitions.

Definition example of system log files

```
pdlogadfg -d sys -g b2log1 ONL
pdlogadfg -d sys -g b2log2 ONL
pdlogadfg -d sys -g b2log3 ONL
pdlogadpf -d sys -g b2log1 -a "/sysarea01/b2log01a" \
-b "/sysarea02/b2log01b"
pdlogadpf -d sys -g b2log2 -a "/sysarea02/b2log02a" \
-b "/sysarea03/b2log02b"
pdlogadpf -d sys -g b2log3 -a "/sysarea03/b2log03a" \
-b "/sysarea01/b2log03b"
```

Definition example of synchronization point dump files

```
pdlogadfg -d spd -g b2sync1 ONL
pdlogadfg -d spd -g b2sync2 ONL
pdlogadfg -d spd -g b2sync3 ONL
pdlogadpf -d spd -g b2sync1 -a "/sysarea01/b2sync01"
pdlogadpf -d spd -g b2sync2 -a "/sysarea02/b2sync02"
pdlogadpf -d spd -g b2sync3 -a "/sysarea03/b2sync03"
```

Definition example of server status files

```
set pd_sts_file_name_1="b2sts1", "/sysarea01/b2sts01a" \
, "/sysarea02/b2sts01b"
set pd_sts_file_name_2="b2sts2", "/sysarea02/b2sts02a" \
, "/sysarea03/b2sts02b"
set pd_sts_file_name_3="b2sts3", "/sysarea03/b2sts03a" \
, "/sysarea01/b2sts03b"
```

(2) Creating the HiRDB file system areas

Use the `pdfmkfs` command to create the HiRDB file system areas.

Example of command entry

```
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea01
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea02
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea03
```

(3) Creating the system files

(a) Creating the system log files

Use the `pdloginit` command to create the system log files.

Example of command entry (FES1)

```
pdloginit -d sys -s f001 -f /sysarea01/f1log01a -n 1024
pdloginit -d sys -s f001 -f /sysarea01/f1log03b -n 1024
pdloginit -d sys -s f001 -f /sysarea02/f1log02a -n 1024
pdloginit -d sys -s f001 -f /sysarea02/f1log01b -n 1024
pdloginit -d sys -s f001 -f /sysarea03/f1log03a -n 1024
pdloginit -d sys -s f001 -f /sysarea03/f1log02b -n 1024
```

Example of command entry (BES1)

```
pdloginit -d sys -s b001 -f /sysarea01/b1log01a -n 1024
pdloginit -d sys -s b001 -f /sysarea01/b1log03b -n 1024
pdloginit -d sys -s b001 -f /sysarea02/b1log02a -n 1024
pdloginit -d sys -s b001 -f /sysarea02/b1log01b -n 1024
```

```
pdloginit -d sys -s b001 -f /sysarea03/b1log03a -n 1024
pdloginit -d sys -s b001 -f /sysarea03/b1log02b -n 1024
```

Example of command entry (BES2)

```
pdloginit -d sys -s b002 -f /sysarea01/b2log01a -n 1024
pdloginit -d sys -s b002 -f /sysarea01/b2log03b -n 1024
pdloginit -d sys -s b002 -f /sysarea02/b2log02a -n 1024
pdloginit -d sys -s b002 -f /sysarea02/b2log01b -n 1024
pdloginit -d sys -s b002 -f /sysarea03/b2log03a -n 1024
pdloginit -d sys -s b002 -f /sysarea03/b2log02b -n 1024
```

(b) Creating the synchronization point dump file

Use the `pdloginit` command to create the synchronization point dump file.

Example of command entry (FES1)

```
pdloginit -d spd -s f001 -f /sysarea01/f1sync01 -n 64
pdloginit -d spd -s f001 -f /sysarea02/f1sync02 -n 64
pdloginit -d spd -s f001 -f /sysarea03/f1sync03 -n 64
```

Example of command entry (BES1)

```
pdloginit -d spd -s b001 -f /sysarea01/b1sync01 -n 64
pdloginit -d spd -s b001 -f /sysarea02/b1sync02 -n 64
pdloginit -d spd -s b001 -f /sysarea03/b1sync03 -n 64
```

Example of command entry (BES2)

```
pdloginit -d spd -s b002 -f /sysarea01/b2sync01 -n 64
pdloginit -d spd -s b002 -f /sysarea02/b2sync02 -n 64
pdloginit -d spd -s b002 -f /sysarea03/b2sync03 -n 64
```

(c) Creating the server status files

Use the `pdstsinit` command to create the server status files.

Example of command entry (FES1)

```
pdstsinit -s f001 -f /sysarea01/f1sts01a -l 4096 -c 256
pdstsinit -s f001 -f /sysarea01/f1sts03b -l 4096 -c 256
pdstsinit -s f001 -f /sysarea02/f1sts02a -l 4096 -c 256
pdstsinit -s f001 -f /sysarea02/f1sts01b -l 4096 -c 256
pdstsinit -s f001 -f /sysarea03/f1sts03a -l 4096 -c 256
pdstsinit -s f001 -f /sysarea03/f1sts02b -l 4096 -c 256
```

Example of command entry (BES1)

```
pdstsinit -s b001 -f /sysarea01/b1sts01a -l 4096 -c 256
pdstsinit -s b001 -f /sysarea01/b1sts03b -l 4096 -c 256
pdstsinit -s b001 -f /sysarea02/b1sts02a -l 4096 -c 256
pdstsinit -s b001 -f /sysarea02/b1sts01b -l 4096 -c 256
pdstsinit -s b001 -f /sysarea03/b1sts03a -l 4096 -c 256
pdstsinit -s b001 -f /sysarea03/b1sts02b -l 4096 -c 256
```

Example of command entry (BES2)

```
pdstsinit -s b002 -f /sysarea01/b2sts01a -l 4096 -c 256
pdstsinit -s b002 -f /sysarea01/b2sts03b -l 4096 -c 256
pdstsinit -s b002 -f /sysarea02/b2sts02a -l 4096 -c 256
pdstsinit -s b002 -f /sysarea02/b2sts01b -l 4096 -c 256
pdstsinit -s b002 -f /sysarea03/b2sts03a -l 4096 -c 256
pdstsinit -s b002 -f /sysarea03/b2sts02b -l 4096 -c 256
```

(d) Creating the unit status files

Use the `pdstsinit` command to create the unit status files.

4. Setting Up an Environment Using Commands

Example of command entry

```
pdstsinit -u unt1 -f /sysarea01/ulsts01a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea01/ulsts03b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/ulsts02a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/ulsts01b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/ulsts03a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/ulsts02b -l 4096 -c 256
```

4.5 Creating system RDAREAs

Executor: HiRDB administrator

When starting HiRDB for the first time, the HiRDB administrator must create system RDAREAs using the database initialization utility (`pdinit`).

Execute the database initialization utility (`pdinit`) when you are starting HiRDB for the first time (when executing the first `pdstart` command after installation) in response to a command input request. You cannot execute the database initialization utility (`pdinit`) at any other time.

This section describes the contents of a *control statement file*, which is specified as an argument of the database initialization utility (`pdinit`) and provides an example of database initialization utility (`pdinit`) execution. To create system RDAREAs, use the `create rdarea` statement.

The system RDAREAs include:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREA

4.5.1 Basics

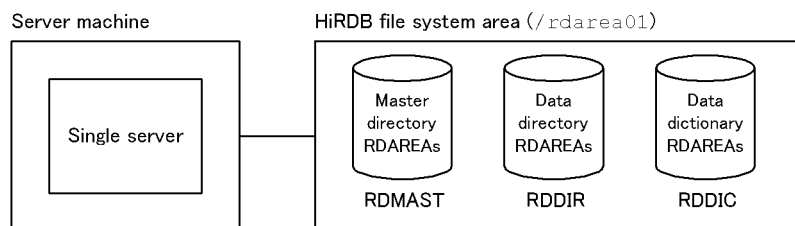
To create system RDAREAs, use the following general procedures:

1. Create a system RDAREA in the HiRDB file system area for RDAREAs that you created in 4.3 *Creating HiRDB file system areas*.
2. For a HiRDB parallel server configuration, create a system RDAREA in a HiRDB file system area for the server machine on which the dictionary server is defined.
3. For details about how to design the system RDAREA, see 8.4 *Placing RDAREAs* for a HiRDB single server configuration, and 9.4 *Placing RDAREAs* for a HiRDB parallel server configuration.
4. For a HiRDB parallel server configuration, execute the database initialization utility (`pdinit`) on the server machine where the system manager is defined.
5. This section only describes the creation of system RDAREAs with the `create rdarea` statement of the database initialization utility (`pdinit`). This is because the system RDAREAs are required for HiRDB operation. You can also use the database initialization utility's (`pdinit`) `create rdarea` statement to define the following RDAREAs:
 - User RDAREAs
 - User LOB RDAREAs
 - Data dictionary LOB RDAREAs
 - RDAREAs for lists

4.5.2 Example 1 (HiRDB single server configuration)

This example creates system RDAREAs in the following HiRDB file system area for RDAREAs:

- `/rdarea01`



(1) Creating the control statement file

Create the control statement file that is to be specified in the database initialization utility's (`pdinit`) argument. You can create the control statement file at any location. This example creates a file under the following filename:

- `/usr/hirdb/pdinit01`

Contents of the control statement file

<code>create rdarea RDMAST for masterdirectory</code>	1
<code>file name "/rdarea01/rdmast01"</code>	
<code>initial 10 segments;</code>	
<code>create rdarea RDDIR for datadirectory</code>	2
<code>file name "/rdarea01/rddir01"</code>	
<code>initial 5 segments;</code>	
<code>create rdarea RDDIC for datadictionary</code>	3
<code>extension use 50 segments</code>	
<code>file name "/rdarea01/rddic01"</code>	
<code>initial 20 segments;</code>	

Explanation:

1. Definition of the master directory RDAREA

This example creates a HiRDB file named `rdmast01` in the HiRDB file system area. This HiRDB file has 10 segments.

2. Definition of data directory RDAREA

This example creates a HiRDB file named `rddir01` in the HiRDB file system area. This HiRDB file has five segments.

3. Definition of the data dictionary RDAREA

This example creates a HiRDB file named `rddic01` in the HiRDB file system area. This HiRDB file has 20 segments.

The example uses the automatic RDAREA extension feature. The extension size is 50 segments.

(2) Executing the database initialization utility (`pdinit`)

Example of command entry

```
pdinit -d /usr/hirdb/pdinit01
```

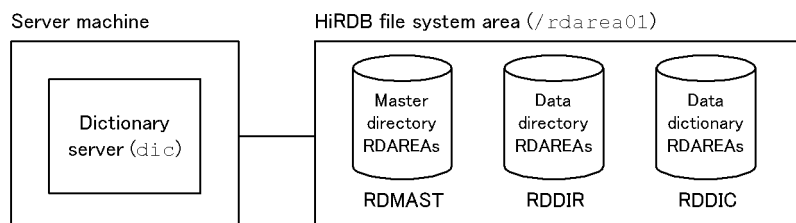
Explanation:

`-d`: Specifies the name of the control statement file that was previously created in (1).

4.5.3 Example 2 (HiRDB parallel server configuration)

This example creates system RDAREAs in the following HiRDB file system area for RDAREAs on the server machine where the dictionary server is defined:

- `/rdarea01`



(1) Creating the control statement file

Create the control statement file that is to be specified in the database initialization utility's (`pdinit`) argument. You can create the control statement file at any location. This example creates a file under the following filename:

- /usr/hirdb/pdinit01

Contents of the control statement file

create rdarea RDMAST for masterdirectory	1
server name dic	
file name "/rdarea01/rdmast01"	
initial 10 segments;	
create rdarea RDDIR for datadirectory	2
server name dic	
file name "/rdarea01/rddir01"	
initial 5 segments;	
create rdarea RDDIC for datadictionary	3
server name dic	
extension use 50 segments	
file name "/rdarea01/rddic01"	
initial 20 segments;	

Explanation:

1. Definition of the master directory RDAREA

This example specifies the name of the dictionary server (`dic`) that manages the master directory RDAREA. It creates a HiRDB file named `rdmast01` in the HiRDB file system area. This HiRDB file has 10 segments.

2. Definition of data directory RDAREA

This example specifies the name of the dictionary server (`dic`) that manages the data directory RDAREA. It creates a HiRDB file named `rddir01` in the HiRDB file system area. This HiRDB file has five segments.

3. Definition of the data dictionary RDAREA

This example specifies the name of the dictionary server (`dic`) that manages the data dictionary RDAREAs. It creates a HiRDB file named `rddic01` in the HiRDB file system area. This HiRDB file has 20 segments.

This example uses the automatic RDAREA extension feature. The extension size is 50 segments.

(2) Executing the database initialization utility (pdinit)

Example of command entry

```
pdinit -d /usr/hirdb/pdinit01
```

Explanation:

- d: Specifies the name of the control statement file that was previously created in (1).

4.6 Starting HiRDB for the first time

Executor: HiRDB administrator

You can execute the database initialization utility (`pdinit`), which was described in *4.5 Creating system RDAREAs*, only after and during the execution of the HiRDB initial startup command (`pdstart`) command.

(1) HiRDB initial startup method

To start HiRDB for the first time (initial startup) after creating HiRDB file system areas, execute the `pdstart` command. When you execute the `pdstart` command, a message is displayed that requests the execution of the database initialization utility (`pdinit`).

- To start a HiRDB single server configuration, execute the `pdstart` command from the server machine where the single server is defined.
- To start a HiRDB parallel server configuration, execute the `pdstart` command from the server machine where the system manager is defined.

(2) Prerequisites for RDAREA creation

To create any of the following RDAREAs, HiRDB must be active. Be sure to start HiRDB beforehand.

- User RDAREAs
- User LOB RDAREAs
- Data dictionary LOB RDAREAs
- RDAREAs for lists

4.7 Creating user RDAREAs

Executor: HiRDB administrator

The HiRDB administrator creates user RDAREAs for storing tables and indexes. To create a user RDAREA, use the database structure modification utility's (`pdmod`) `create rdarea` statement.

4.7.1 Basics

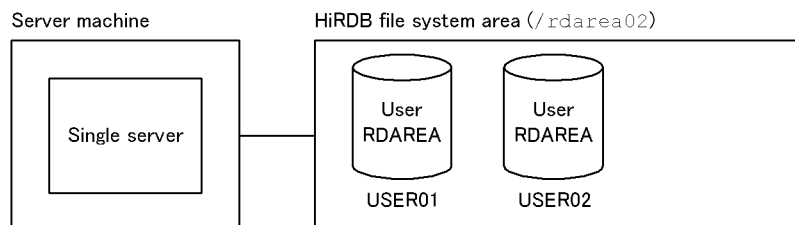
To create user RDAREAs, use the following general procedures:

1. Create a user RDAREA in the HiRDB file system area for RDAREAs that you created in *4.3 Creating HiRDB file system areas*.
2. For a HiRDB parallel server configuration, create a user RDAREA in a HiRDB file system area for the server machine on which the back-end server is defined.
3. For details about how to design the user RDAREA, see *8.4 Placing RDAREAs* for a HiRDB single server configuration, and *9.4 Placing RDAREAs* for a HiRDB parallel server configuration.
4. For a HiRDB parallel server configuration, execute the database structure modification utility (`pdmod`) on the server machine where the system manager is defined.
5. Before creating a user RDAREA, use the `pdls` command to make sure that HiRDB is running. For a HiRDB parallel server configuration, enter the `pdls` command from the server machine where the system manager is defined.
6. If HiRDB is not running, use the `pdstart` command to start it. To start a HiRDB parallel server configuration, enter the `pdstart` command from the server machine where the system manager is defined.

4.7.2 Example 1 (HiRDB single server configuration)

This example creates user RDAREAs in the following HiRDB file system area for RDAREAs:

- `/rdarea02`



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (`pdmod`) argument. You can create the control statement file at any location. This example creates a file under the following filename:

- `/usr/hirdb/pdmod01`

Contents of the control statement file

```

create rdarea USER01 for user used by PUBLIC          1
  extension use 50 segments
  file name "/rdarea02/user01"
  initial 500 segments;
create rdarea USER02 for user used by PUBLIC          2
  extension use 50 segments
  file name "/rdarea02/user02"
  initial 500 segments;
  
```

Explanation:

1. Definition of the user RDAREA (USER01)

USER01 is a public RDAREA (PUBLIC). This example creates a HiRDB file named `user01` in the HiRDB file system area. This HiRDB file has 500 segments. The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

2. Definition of the user RDAREA (USER02)

USER02 is a public RDAREA (PUBLIC). This example creates a HiRDB file named `user02` in the HiRDB file system area. This HiRDB file has 500 segments. The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod01
```

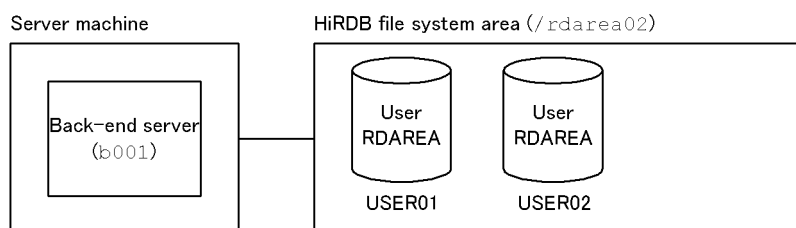
Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

4.7.3 Example 2 (HiRDB parallel server configuration)

This example creates user RDAREAs in the following HiRDB file system area for RDAREAs on the server machine where the back-end server is defined:

- /rdarea02



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (pdmod) argument. You can create the control statement file at any location. This example creates the file under the following filename:

- /usr/hirdb/pdmod01

Contents of the control statement file

```
create rdarea USER01 for user used by PUBLIC          1
  server name b001
  extension use 50 segments
  file name "/rdarea02/user01"
  initial 500 segments;
create rdarea USER02 for user used by PUBLIC          2
  server name b001
  extension use 50 segments
  file name "/rdarea02/user02"
  initial 500 segments;
```

Explanation:

1. Definition of the user RDAREA (USER01)

USER01 is a public RDAREA (PUBLIC). This example specifies the name of the back-end server (b001) that manages USER01. It creates a HiRDB file named `user01` in the HiRDB file system area. This HiRDB

file has 500 segments. The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

2. Definition of the user RDAREA (USER02)

USER02 is a public RDAREA (PUBLIC). This example specifies the name of the back-end server (b001) that manages USER02. It creates a HiRDB file named `user02` in the HiRDB file system area. This HiRDB file has 500 segments. The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod01
```

Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

4.8 Creating user LOB RDAREAs

Executor: HiRDB administrator

To create data with the LOB attribute, you need a user LOB RDAREA to store the data. To create a user LOB RDAREA, use the database structure modification utility's (`pdmod`) `create rdarea` statement.

4.8.1 Basics

To create user LOB RDAREAs, use the following general procedures:

1. Create a user LOB RDAREA in the HiRDB file system area for RDAREAs, which you created in *4.3 Creating HiRDB file system areas*.
2. For a HiRDB parallel server configuration, create a user LOB RDAREA in a HiRDB file system area for the server machine on which the back-end server is defined.
3. For details about how to design the user LOB RDAREA, see *8.4 Placing RDAREAs* for a HiRDB single server configuration, and *9.4 Placing RDAREAs* for a HiRDB parallel server configuration.
4. For a HiRDB parallel server configuration, execute the database structure modification utility (`pdmod`) on the server machine where the system manager is defined.
5. Before creating a user LOB RDAREA, use the `pdls` command to make sure that HiRDB is running. For a HiRDB parallel server configuration, enter the `pdls` command from the server machine where the system manager is defined.
6. If HiRDB is not running, use the `pdstart` command to start it. To start a HiRDB parallel server configuration, enter the `pdstart` command from the server machine where the system manager is defined.

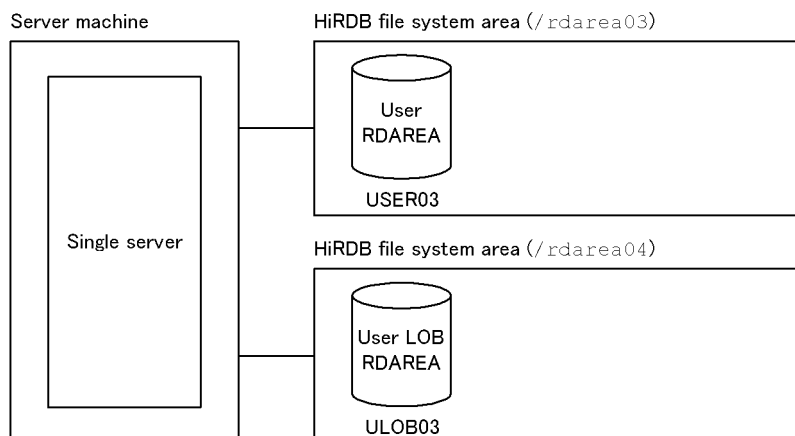
4.8.2 Example 1 (HiRDB single server configuration)

This example creates a user RDAREA to store a LOB column structure base table in the following HiRDB file system area for RDAREA:

- `/rdarea03`

Additionally, the example creates a user LOB RDAREA to store data with the LOB attribute in the following HiRDB file system area for LOB RDAREA:

- `/rdarea04`



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (pdmod) argument. You can create the control statement file at any location. This example creates a file under the following filename:

- /usr/hirdb/pdmod02

Contents of the control statement file

create rdarea USER03 for user used by PUBLIC	1
extension use 50 segments	
file name "/rdarea03/user03"	
initial 500 segments;	
create rdarea ULOB03 for LOB used by PUBLIC	2
extension use 50 segments	
file name "/rdarea04/ulob03"	
initial 20000 segments;	

Explanation:

1. Definition of the user RDAREA (USER03)

USER03 is a public RDAREA (PUBLIC). This example creates a HiRDB file named `user03` in the HiRDB file system area. This HiRDB file has 500 segments. The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

2. Definition of the user LOB RDAREA (ULOB03)

ULOB03 is a public RDAREA (PUBLIC). This example creates a HiRDB file named `ulob03` in the HiRDB file system area. This HiRDB file has 20,000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod02
```

Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

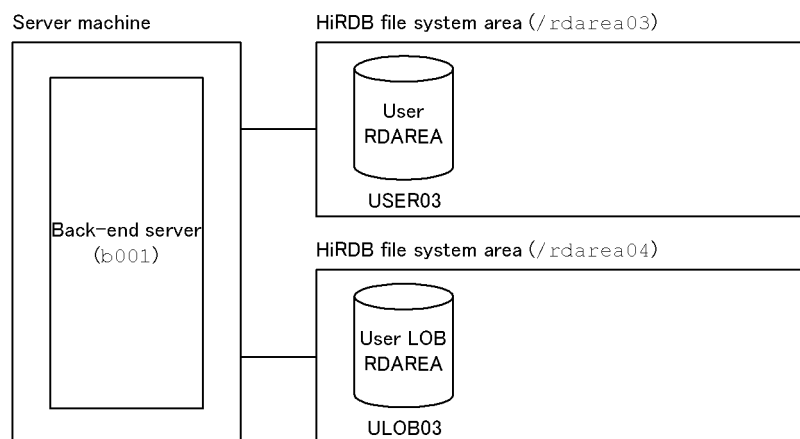
4.8.3 Example 2 (HiRDB parallel server configuration)

This example creates a user RDAREA to store a LOB column structure base table in the following HiRDB file system area for RDAREA:

- /rdarea03

Additionally, the example creates a user LOB RDAREA to store data with the LOB attribute in the following HiRDB file system area for RDAREA:

- /rdarea04



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (pdmod) argument. You can create the control statement file at any location. This example creates the file under the following filename:

- /usr/hirdb/pdmod02

Contents of the control statement file

```

create rdarea USER03 for user used by PUBLIC          1
  server name b001
  extension use 50 segments
  file name "/rdarea03/user03"
  initial 500 segments;
create rdarea ULOB03 for LOB used by PUBLIC            2
  server name b001
  extension use 50 segments
  file name "/rdarea04/ulob03"
  initial 20000 segments;
  
```

Explanation:

1. Definition of the user RDAREA (USER03)

USER03 is a public RDAREA (PUBLIC). This example specifies the name of the back-end server (b001) that manages USER03. It creates a HiRDB file named `user03` in the HiRDB file system area. This HiRDB file has 500 segments. The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

2. Definition of the user LOB RDAREA (ULOB03)

ULOB03 is a public RDAREA (PUBLIC). This example specifies the name of the back-end server (b001) that manages ULOB03. It creates a HiRDB file named `ulob03` in the HiRDB file system area. This HiRDB file has 20,000 segments. The example uses the RDAREA automatic extension feature. The extension size is 50 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod02
```

Explanation:

- a: Specifies the name of the control statement file that was previously created in (1).

4.9 Creating data dictionary LOB RDAREAs

Executor: HiRDB administrator

To use stored procedures or stored functions, you need a data dictionary LOB RDAREA. You can create a data dictionary LOB RDAREA using the database structure modification utility's (`pdmod`) `create rdarea` statement.

You need to provide separate data dictionary LOB RDAREAs according to these purposes:

- Data dictionary LOB RDAREA for storing stored procedures' or stored functions' definition source
- Data dictionary LOB RDAREA for storing stored procedures' or stored functions' SQL objects

4.9.1 Basics

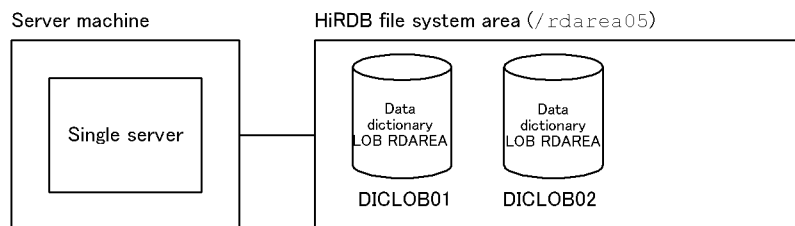
To create data dictionary LOB RDAREAs, use the following general procedures:

1. Create a data dictionary LOB RDAREA in the HiRDB file system area for RDAREAs, which you created in *4.3 Creating HiRDB file system areas*.
2. For a HiRDB parallel server configuration, create a data dictionary LOB RDAREA in a HiRDB file system area for the server machine on which the dictionary server is defined.
3. For details about how to design the data dictionary LOB RDAREA, see *8.4 Placing RDAREAs* for a HiRDB single server configuration, and *9.4 Placing RDAREAs* for a HiRDB parallel server configuration.
4. For a HiRDB parallel server configuration, execute the database structure modification utility (`pdmod`) on the server machine where the system manager is defined.
5. Before creating a data dictionary LOB RDAREA, use the `pdls` command to make sure that HiRDB is running. For a HiRDB parallel server configuration, enter the `pdls` command from the server machine where the system manager is defined.
6. If HiRDB is not running, use the `pdstart` command to start it. To start a HiRDB parallel server configuration, enter the `pdstart` command from the server machine where the system manager is defined.

4.9.2 Example 1 (HiRDB single server configuration)

This example creates data dictionary LOB RDAREAs in the following HiRDB file system area for RDAREA:

- `/rdarea05`



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (`pdmod`) argument. You can create the control statement file at any location. This example creates a file under the following filename:

- `/usr/hirdb/pdmod03`

Contents of the control statement file

```
create rdarea DICLOB01 for LOB used by HiRDB(SQL_ROUTINES)      1
extension use 1000 segments
file name "/rdarea05/diclob01"
```

```

        initial 10000 segments;
create rdarea DICLOB02 for LOB used by HiRDB(SQL_ROUTINES)      2
extension use 1000 segments
file name "/rdarea05/diclob02"
        initial 10000 segments;

```

Explanation:

1. Definition of the data dictionary RDAREA (DICLOB01)

DICLOB01 is a data dictionary LOB RDAREA for storing a definition source. The system uses the first data dictionary LOB RDAREA defined for storing a definition source. This example creates a HiRDB file named `diclob01` in the HiRDB file system area. This HiRDB file has 10,000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 1,000 segments.

2. Definition of the data dictionary RDAREA (DICLOB02)

DICLOB02 is a data dictionary LOB RDAREA for storing SQL objects. The system uses the second data dictionary LOB RDAREA defined for storing SQL definitions. This example creates a HiRDB file named `diclob02` in the HiRDB file system area. This HiRDB file has 10,000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 1,000 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod03
```

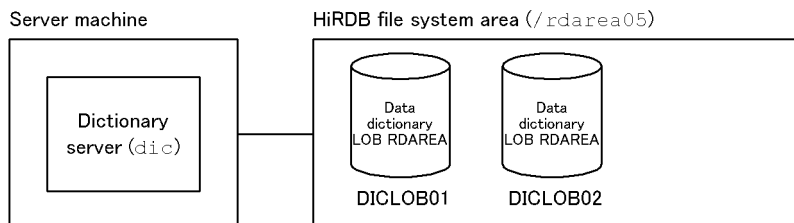
Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

4.9.3 Example 2 (HiRDB parallel server configuration)

This example creates data dictionary LOB RDAREAs in the following HiRDB file system area for RDAREA:

- /rdarea05



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (`pdmod`) argument. You can create the control statement file at any location. This example creates the file under the following filename:

- /usr/hirdb/pdmod03

Contents of the control statement file

```

create rdarea DICLOB01 for LOB used by HiRDB(SQL_ROUTINES)      1
server name dic
extension use 1000 segments
file name "/rdarea05/diclob01"
        initial 10000 segments;
create rdarea DICLOB02 for LOB used by HiRDB(SQL_ROUTINES)      2
server name dic
extension use 1000 segments
file name "/rdarea05/diclob02"
        initial 10000 segments;

```

Explanation:

1. Definition of the data dictionary RDAREA (DICLOB01)

DICLOB01 is a data dictionary LOB RDAREA for storing a definition source. The system uses the first data dictionary LOB RDAREA defined for storing a definition source. This example specifies the name of the dictionary server (dic) that manages this data dictionary LOB RDAREA. It creates a HiRDB file named diclob01 in the HiRDB file system area. This HiRDB file has 10,000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 1,000 segments.

2. Definition of the data dictionary RDAREA (DICLOB02)

DICLOB02 is a data dictionary LOB RDAREA for storing SQL objects. The system uses the second data dictionary LOB RDAREA defined for storing SQL definitions. This example specifies the name of the dictionary server (dic) that manages this data dictionary LOB RDAREA. It creates a HiRDB file named diclob02 in the HiRDB file system area. This HiRDB file has 10,000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 1,000 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod03
```

Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

4.10 Creating list RDAREAs

Executor: HiRDB administrator

To use a narrowed search, you need a list RDAREA. You can create list RDAREAs using the database structure modification utility's (`pdmod`) `create rdarea` statement.

4.10.1 Basics

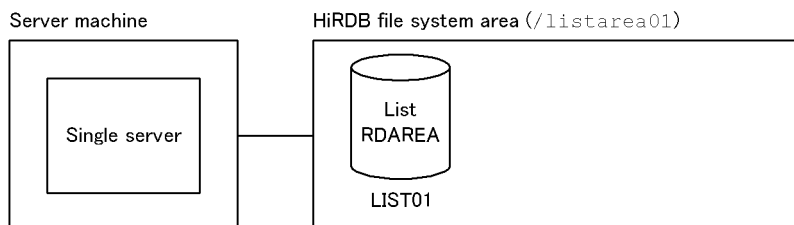
To create RDAREAs for lists, use the following general procedures:

1. Create a list RDAREA in the HiRDB file system area for list RDAREAs, which you created in *4.3 Creating HiRDB file system areas*.
2. For a HiRDB parallel server configuration, create a list RDAREA in a HiRDB file system area for the server machine on which the back-end server (that contains the base table) is defined.
3. For details about how to design RDAREAs for lists, see *8.4 Placing RDAREAs* for a HiRDB single server configuration, and *9.4 Placing RDAREAs* for a HiRDB parallel server configuration.
4. For a HiRDB parallel server configuration, execute the database structure modification utility (`pdmod`) on the server machine where the system manager is defined.
5. Before creating a list RDAREA, use the `pdls` command to make sure that HiRDB is running. For a HiRDB parallel server configuration, enter the `pdls` command from the server machine where the system manager is defined.
6. If HiRDB is not running, use the `pdstart` command to start it. To start a HiRDB parallel server configuration, enter the `pdstart` command from the server machine where the system manager is defined.

4.10.2 Example 1 (HiRDB single server configuration)

This example creates an list RDAREA in the following HiRDB file system area for list RDAREAs:

- `/listarea01`



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (`pdmod`) argument. You can create the control statement file at any location. This example creates a file under the following filename:

- `/usr/hirdb/pdmod04`

Contents of the control statement file

```
create rdarea LIST01 for list                                1
page 4096 characters storage control segment 2 pages
file name "/listarea01/list01"
initial 1000 segments;
```

Explanation:

1. Definition of the list RDAREAs (`LIST01`)

This example specifies the page length and segment size of the RDAREA. It creates a HiRDB file named `list01` in the HiRDB file system area. This HiRDB file has 1,000 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod04
```

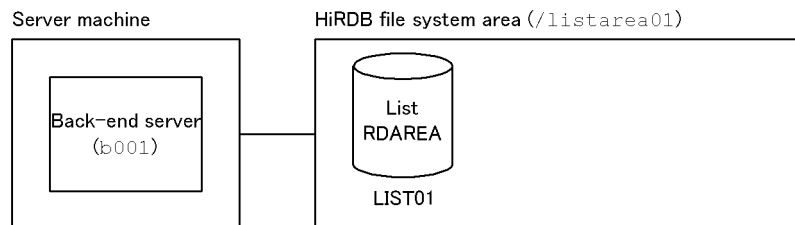
Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

4.10.3 Example 2 (HiRDB parallel server configuration)

This example creates a list RDAREA in the following HiRDB file system area for list RDAREAs:

- /listarea01



(1) Creating the control statement file

Create the control statement file that is to be specified in the database structure modification utility's (pdmod) argument. You can create the control statement file at any location. This example creates the file under the following filename:

- /usr/hirdb/pdmod04

Contents of the control statement file

```
create rdarea LIST01 for list                                1
server name b001
page 4096 characters storage control segment 2 pages
file name "/listarea01/list01"
initial 1000 segments;
```

Explanation:

1. Definition of the list RDAREA (LIST01)

This example specifies the name of the back-end server that manages LIST01 as well as the page length and segment size of the RDAREA. It creates a HiRDB file named `list01` in the HiRDB file system area. This HiRDB file has 1,000 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a /usr/hirdb/pdmod04
```

Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

5

Setting Up the Plug-in Environment

A plug-in environment is set up after the HiRDB environment setup. This chapter describes the procedures for setting up a plug-in environment, as well as for upgrading and deleting (uninstalling) plug-ins.

5.1 Overview of plug-in environment setup

This section describes the procedure for setting up HiRDB plug-ins.

5.1.1 Environment setup procedure

Executor: HiRDB administrator

This section describes the plug-in environment setup procedure using commands. This procedure assumes that the HiRDB environment setup has been completed (HiRDB is already running).

To set up the plug-in environment:

1. Estimate the resources needed to install plug-ins.
2. Terminate the active HiRDB.
3. Install plug-ins.
4. Set up plug-ins.
5. Start HiRDB.
6. Add data dictionary LOB RDAREAs, user RDAREAs, and user LOB RDAREAs.^{#1}
7. Register plug-ins.
8. Initialize the registry facility.^{#2}
9. Terminate HiRDB.
10. Add the `pdplugin` operand.
11. Start HiRDB.
12. Register registry information.

#1: A data dictionary LOB RDAREA is not necessary if stored functions, stored procedures, or plug-ins are already being used. A user RDAREA (user LOB RDAREA) is required if a table is created for a newly added plug-in.

#2: May not be needed, depending on the plug-in.

(1) Estimating resources

Before a plug-in can be installed into the HiRDB system, the sizes of the following resources must be estimated:

- Storage requirement for execution of the plug-in
- Disk space required in order to install the plug-in

For details about how to estimate the resources required for each plug-in, see the applicable plug-in documentation.

(2) Terminating HiRDB

Before setting up plug-ins, use the `pdstop` command to terminate the active HiRDB.

(3) Installing plug-ins

Install your plug-ins. For details about the installation procedure, see the applicable plug-in documentation.

(4) Setting up the plug-ins

The HiRDB administrator executes the setup procedure and specification.

The HiRDB administrator executes the `pdplgset` command to set up plugins in HiRDB. The HiRDB administrator must set up plugins in each server machine on which HiRDB is installed.

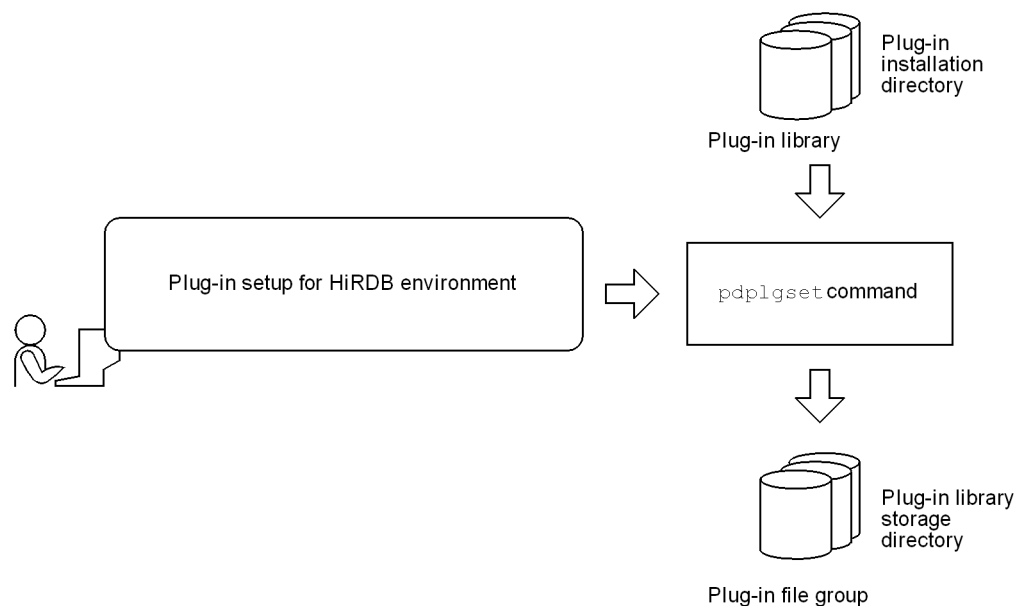
To set up plug-ins, use the following procedure:

1. Use the `pdls` command to check that HiRDB is not operating. If it is operating, use the `pdstop` command to terminate HiRDB normally.
2. Execute the `pdplgset` command.

When the `pdplgset` command executes, the plug-in library is copied from the directory where plugins are installed to `$PDDIR/plugin/`.

The following figure illustrates the plug-in setup procedure.

Figure 5–1: How to set up plug-ins



Explanation:

The `pdplgset` command automatically creates the plugin directory under the HiRDB directory (`$PDDIR`) and the directory with the plug-in name.

(5) Starting HiRDB

The HiRDB administrator uses the `pdstart` command to start HiRDB.

(6) Adding user RDAREAs

Before plug-ins are registered into HiRDB, the RDAREA administrator uses the `create rdarea` statement of the database structure modification utility (`pdmod`) to add RDAREAs. The following RDAREAs need to be added:

- User RDAREA^{#1}
- User LOB RDAREA^{#1}
- Data dictionary LOB RDAREA^{#2} (there is no need to add this RDAREA if stored procedures, stored functions, or plug-ins are already being used)

For details about how to add RDAREAs, see [4.7 Creating user RDAREAs](#), [4.8 Creating user LOB RDAREAs](#), or [4.9 Creating data dictionary LOB RDAREAs](#).

If a database environment has already been constructed, there is no need to add RDAREAs after installing plug-ins.

#1: You need to add this RDAREA if you want to create a separate table for plug-ins and store the table in a new RDAREA.

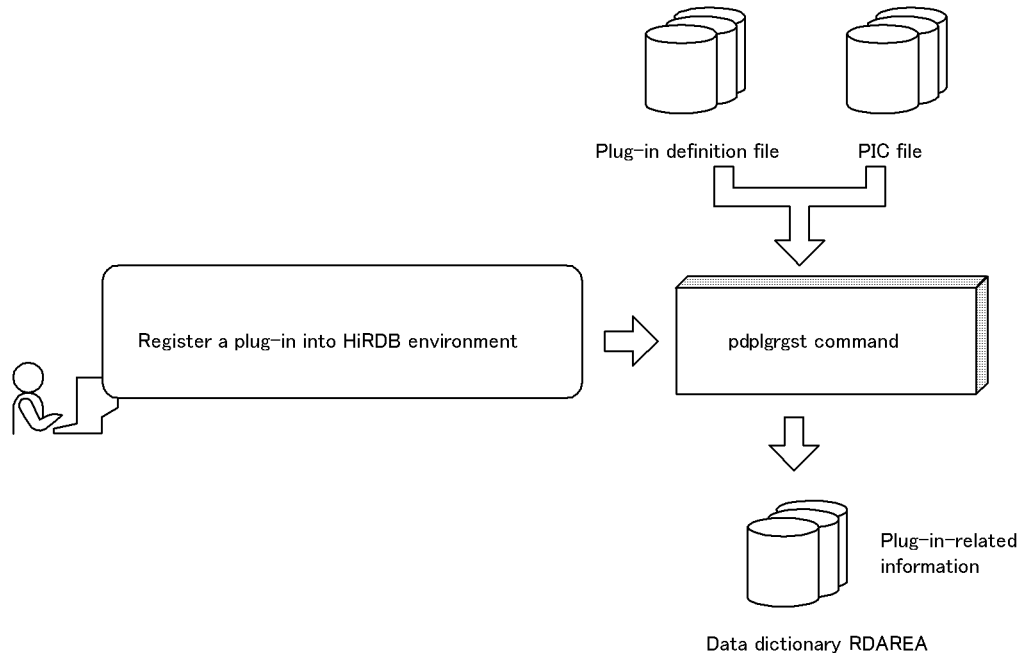
#2: You need to add this RDAREA to make the stored-procedure facility available to HiRDB before the registry facility initialization utility (`pdreginit`) is executed.

(7) Registering plug-ins

Use the `pdplgrgst` command to register your plug-ins in HiRDB. You can enter the `pdplgrgst` command from any server machine.

The following figure illustrates the plug-in registration procedure.

Figure 5–2: Plug-in registration procedure



(a) `pdplgrgst` command input format

Following shows the input format of the `pdplgrgst` command:

```
pdplgrgst plug-in-definition-filename PIC-filename
```

Example of HiRDB Text Search Plug-in

- Data type plug-in
`pdplgrgst_phsgml.adt_phsgml.pic`
 (Current directory: /TSPlugin/_phsgml/etc)
- Index type plug-in
`pdplgrgst_phngram.idx_phngram.pic`
 (Current directory: /TSPlugin/_phngram/etc)

Notes

- To register an index-type plug-in, you need to register the corresponding data type plug-in beforehand.
- Register both data type and index type plug-ins in the same schema.

(b) Owner of a plug-in

The owner of a plug-in (owner of the abstract data type, index type, and function provided by a plug-in) is treated as MASTER. This allows the authorization identifier to be omitted when specifying the plug-in-provided function call processing using SQL statements.

Specifying a user other than MASTER as the plug-in owner

You can specify the user executing the `pdplgrget` command as the plug-in owner instead of MASTER. To do this, specify the `-u` option in the `pdplgrget` command, in which case the `pdplgrget` command executor (the authorization identifier specified in the `PDUSER` operand in the client environment definitions) becomes the owner of the plug-in.

Notes

1. The `pdplgrgst` command executor's schema must have already been defined.
2. If a plug-in provides both abstract data type and index type, be sure to assign the same owner.
3. Only the plug-in owner can delete or upgrade his/her plug-in. To delete or upgrade a plug-in, specify the `-u` option in the `pdplgrgst` command.
4. If you delete a plug-in owner's schema, the plug-in itself is also deleted. In this case, do the following:
 - Delete the `pdplugin` operand from the system common definitions.
 - Execute the `pdplgset -d` command at all server machines to set up plugins.
5. When more than one plug-in includes a function that has the same name and number of parameters, you must first register one of the plug-ins, define the function that calls the function in that plug-in, and then register the other plug-ins. However, an error is returned when the plug-in is registered if the function you define declares an abstract data type for a parameter or a return value, and that function is used in a view definition. If this occurs, you must delete the view table that uses that function, and then register the plug-in again.

(c) Migrating from HiRDB Version 5.0 (with HiRDB Object Option) 01-00**The plug-in owner remains the same.**

With HiRDB Version 5.0 (with HiRDB Object Option) 01-00, the plug-in's owner is the user who executes the `pdplgrgst` command. After upgrading HiRDB to Version 6 06-00 or later, the plugin's owner remains the same (not MASTER).

How to define the owner as MASTER

To define the owner as MASTER, you need to delete the plugin once and then re-register it.

To delete a plug-in, you must first delete the abstract data type, index type, and the tables, view tables and indexes used by the function that are provided by the plug-in. Then, execute the `pdplgrgst` command.

If the authorization identifier is specified in the section where the function provided by the plug-in is called with the SQL statement, you need to delete the authorization identifier or change it to MASTER.

(8) Initializing the registry facility

Some plug-ins require the registry facility. In such cases, use the registry facility initialization utility's (`pdreginit`) `create rdarea` statement to create the following RDAREAs. This operation is not necessary when the registry facility is already being used with plug-ins.

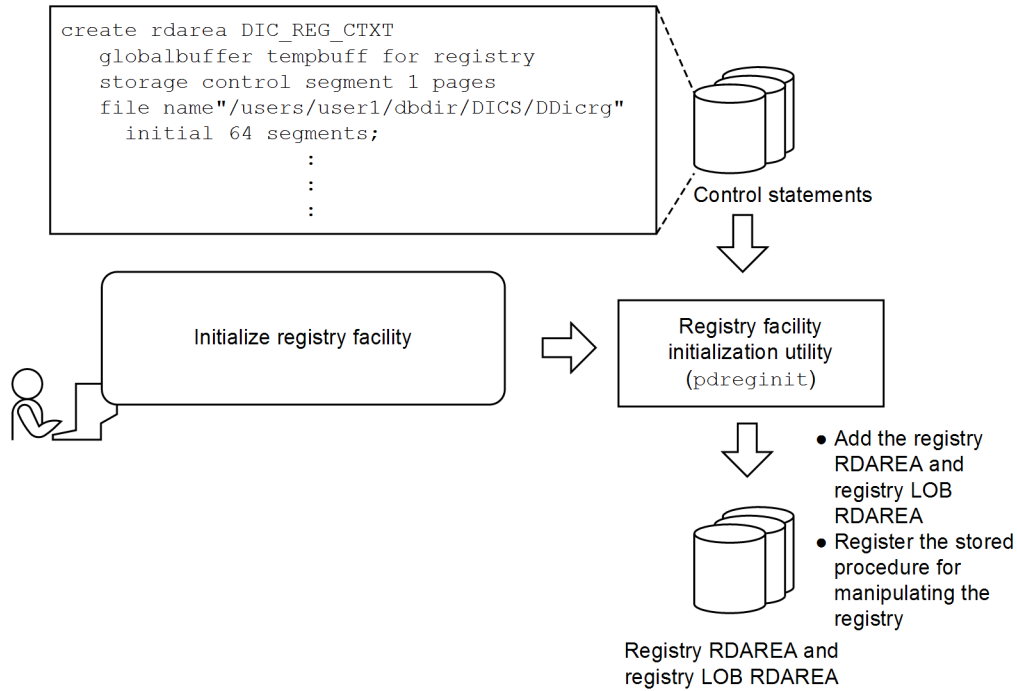
- Registry RDAREA
- Registry LOB RDAREA

Execute the registry facility initialization utility (`pdreginit`) only once before all plug-ins are registered.

The registry RDAREA and registry LOB RDAREA store the registry information. Which of the two RDAREAs is used is determined automatically on the basis of the length of the data to be registered.

The following figure illustrates the procedure for creating the registry RDAREA and the registry LOB RDAREA.

Figure 5–3: Procedure for creating a registry RDAREA and registry LOB RDAREA



(9) Terminating HiRDB

To enable a plug-in, HiRDB must be terminated normally by entering the `pdstop` command. No tables or indexes that use the registered plug-in can be defined until HiRDB has been restarted.

After HiRDB has been terminated, a backup copy should be made of all the updated RDAREAs.

(10) Adding the pdplugin operand

After HiRDB has been terminated normally, add the `pdplugin` operand in the system common definitions. Specify the name of a plug-in to be used in the `pdplugin` operand.

For a HiRDB parallel server configuration, you need to add the `pdplugin` operand in the system common definitions on all server machines; otherwise, you will not be able to start HiRDB.

(11) Starting HiRDB

Use the `pdstart` command to start HiRDB.

(12) Registering registry information

Once the registry facility has been initialized, registry information required by the plug-in must be registered. The plug-in and the registry facility can then be used. For details about how to register registry information, see the applicable plug-in documentation.

5.1.2 Notes on using plug-ins

(1) HiRDB setup/startup conditions and availability of plug-ins

The following table shows the HiRDB (unit) setup/startup conditions and availability of plug-ins.

Table 5–1: HiRDB (unit) setup/startup conditions and availability of plug-ins

Plug-in utilization declaration (pdplugin operand)	Registration of plug-in (pdplgrgst)	Plug-in initialization error	Whether a unit can be started	Availability of plug-in
Not specified	Registered	None	S	--
		Occurred	S	--
	Not registered	None	S	--
		Occurred	S	--
Specified	Registered	None	S	S
		Occurred	S	--
	Not registered	None	S	--
		Occurred	S	--

S: Unit can be started, and plug-in can be used.

--: Unit cannot be started, nor can plug-in be used.

(2) Availability of plug-ins in the event of plug-in initialization error

Plug-ins are initialized automatically during HiRDB startup. If multiple `pdplugin` operands are specified in the system common definition and at least one of them results in a plug-in initialization error, none of the plug-ins in the unit can be used.

(3) Availability of plug-ins depending on the unit

The availability of plug-ins depends on conditions at the unit in the following cases:

- The plug-in utilization declaration (`pdplugin` operand) in the system common definition is not the same from one unit to another.
- An error occurred during plug-in initialization processing and the plug-in is no longer available on that unit.

An SQL statement that calls only the available plug-ins will execute successfully; however, if it attempts to call even one unavailable plug-in, execution of the SQL statement will fail.

5.2 Upgrading plug-ins

This section describes the procedure for upgrading plug-ins installed in HiRDB (data type and index type plug-ins). Upgrading a plug-in means changing the plug-in without deleting the following:

- Tables and view tables that use the data type provided by the plug-in.
- Indexes using the index type provided by the plug-in.
- Functions provided by the plug-in.

For notes about upgrading plug-ins, see *Notes in pdplgrgst (Register and delete plug-ins)* in the manual *HiRDB Version 9 Command Reference*.

This section describes the upgrading procedure.

(1) Making a backup copy

To protect against possible errors, use the database copy utility (`pdcopy`) to make a backup copy of the following RDAREAs. For this purpose, be sure to specify the `-M x` option in the database copy utility (`pdcopy`).

- Master directory RDAREA
- Data dictionary RDAREAs
- Data directory RDAREA
- Data dictionary LOB RDAREAs

For details about making backups, see the *HiRDB Version 9 System Operation Guide*.

(2) Terminating HiRDB

Enter the `pdstop` command to terminate HiRDB normally.

(3) Saving all necessary files

Save all necessary files at the following location:

- `$PDDIR/plugin/plugin-name` directory

For details about the files to be saved, see the applicable plug-in documentation. Save the applicable files on all server machines where HiRDB is set up.

(4) Cancelling setup of the old version of the plug-in

Use the `pdplgset -d` command to cancel setup of the plug-in. This command deletes everything at the following location:

- `$PDDIR/plugin/plugin-name` directory

This command does not delete any file under the `CONF` directory.

Cancel setup on all server machines where HiRDB is set up. For details about how to cancel plug-in setup, see *5.1 Overview of plug-in environment setup*.

(5) Installing the new version of the plug-in

Install the new version of the plug-in at each HiRDB server machine. For details about the installation procedure, see the applicable plug-in documentation.

(6) Setting up the new version of the plug-in

Use the `pdplgset` command to set up the new plug-in. The new plug-in must be set up on all server machines where HiRDB is set up. For details about how to set up plugins, see *5.1 Overview of plug-in environment setup*.

(7) Making appropriate changes to the system common definition

Delete the `pdplugin` operand for the old version of the plug-in from the system common definition. Delete this operand at all the server machines that contain the applicable system common definition.

(8) Restoring files that were saved

Restore the files that were saved in step (3) at all server machines where HiRDB is set up.

(9) Restarting HiRDB

Enter the `pdstart` command to start HiRDB.

(10) Registering the new version of the plug-in

Execute the `pdplgrgst` command, specifying the `-a` option to re-register the plug-in.

(11) Terminating HiRDB

Enter the `pdstop` command to terminate HiRDB normally.

(12) Making appropriate changes to the system common definition

When HiRDB has terminated normally, add the `pdplugin` operand to the system common definition. In the `pdplugin` operand, specify the name of the upgraded plug-in.

For a HiRDB parallel server configuration, add this operand to the system common definition at all server machines; otherwise, you will not be able to start HiRDB.

(13) Starting HiRDB

Enter the `pdstart` command to start HiRDB. The tables and indexes that were defined before the plug-in was upgraded become available again. Any new facilities provided by the new version of plug-in are also available.

5.3 Deleting plug-ins

This section describes the procedure for deleting plug-ins that have been registered into HiRDB. Deleting a plug-in means the following:

- Deleting the plug-in definition information registered in the dictionary.
- Deleting the function, abstract data type, and index type provided by the plug-in.

The plug-in-provided file that contains the plug-in file set is not deleted.

(1) Deleting the database resources that use the facility provided by the plug-in

Before deleting a plug-in, the HiRDB administrator must delete the database resources listed below. The table below lists the SQL statements issued to delete the following resources:

- Tables, view tables, functions, procedures, and abstract data types (when a user-defined abstract data type specifies the abstract data type provided by the plug-in as one of its attributes) that use the abstract data type provided by the plug-in that is to be deleted
- Index using the index type that is provided by the plug-in that is to be deleted
- Function and procedure using the function provided by the plug-in that is to be deleted

Table 5–2: SQL statements for deleting database resources

Database resources to be deleted	SQL statement to be used
Table	DROP TABLE
Index	DROP INDEX
View table	DROP VIEW
Function	DROP FUNCTION
Procedure	DROP PROCEDURE
Abstract data type	DROP DATA TYPE [#]

[#]: The data type provided by a plug-in must not be deleted.

(2) Deleting the registered plug-in

The following command is executed for each plug-in that is to be deleted:

```
pdplgrgst -d plug-in-definition-filename PIC-filename
```

Notes

1. When a data-type plug-in is to be deleted and an index-type plug-in providing the index facility for that data type is also registered, the index-type plug-in must be deleted first.
2. If you are deleting a plug-in with an owner who is not MASTER, note the following:
 - Only a plug-in's owner can delete his/her plug-in. Specify the authorization identifier and password of the plug-in owner in the `PDUSER` operand in the client environment definitions. Then, specify the `-u` option when executing the `pdplgrgst` command.
 - The owner of a plugin registered with HiRDB Version 5.0 (with HiRDB Object Option) is not MASTER. Therefore, you need to take the actions described previously.
 - If you delete a plug-in owner's schema, the plug-in itself is also deleted. For details about the actions to be taken after the plug-in has been deleted, see the sections beginning at (3) as follows.

(3) Deleting the registry

For details about how to delete registry information, see the applicable plug-in documentation.

(4) Terminating HiRDB

The `pdstop` command must be entered to terminate HiRDB normally.

(5) Making appropriate changes to the system common definition

When HiRDB is terminated normally, the `pdplugin` operand must be deleted from the applicable system common definitions.

(6) Canceling setup of the plug-in

The `pdplgset -d` command is used to cancel setup of the plug-in. Before this command is executed, save all necessary files. No files are deleted from the `conf` directory. Plug-in setup cancellation must be performed at all units where HiRDB is set up.

(7) Uninstalling the plug-in

The plug-in must be uninstalled from the server machine. For details about the uninstallation, see the procedure for the applicable plug-in.

6

Creating Databases

This chapter describes the procedures from schema, table, and index creation through data storage.

6.1 Overview of database creation

This section provides the information that you need to be familiar with before creating databases (tables and indexes). The topics covered include:

- Preparing for database creation
- Database creation procedure
- Database update log acquisition procedure
- Notes about loading data to a table containing a unique index
- Loading a large amount of data (data load with the synchronization point specification)
- Loading data into a row-partitioned table (using the parallel loading facility)
- Loading data into a row-partitioned table (creating divided-input data files)
- Data loads that use the automatic numbering facility
- Input data file UOC
- Checking for unneeded RDAREAs

6.1.1 Preparing for database creation

Executor: HiRDB administrator

This section discusses preparations for database creation.

(1) Defining the client environment

Before a database can be created, the following environment variables must be set in the client environment definition. (For details about how to set client environment variables, see the *HiRDB Version 9 UAP Development Guide*):

- PDHOST
- PDUSER
- PDNAMEPORT

(2) Changing the password

If the password of the authorization identifier of the HiRDB administrator is the same character string as the authorization identifier, use the GRANT definition SQL to change the password. To do this, execute the following SQL with the database definition utility (pddef) or HiRDB SQL Executor:

```
GRANT DBA TO identifier-for-the-HiRDB-administrator IDENTIFIED BY new-password;
```

(3) Defining a schema

The CREATE SCHEMA definition SQL is used to define a schema. To define a schema, either the database definition utility (pddef command) is used or an appropriate UAP is created. Only one schema can be defined for each user.

(4) Creating a database (users other than the HiRDB administrator)

Before a user other than the HiRDB administrator can create a database, the HiRDB administrator must grant an appropriate user privilege to this user. Use the GRANT definition SQL to grant the necessary user privilege to the user who will be creating databases. The following privileges are required:

- CONNECT privilege
- Schema definition privilege
- RDAREA usage privilege

For details about user privileges, see the *HiRDB Version 9 System Operation Guide*.

Example

Grant the CONNECT, schema, and RDAREA usage (RDAREA name: RDAREA01) privileges to the user who will create tables (authorization identifier: USER002, password: HIRDB002):

```
GRANT CONNECT TO USER002 IDENTIFIED BY HIRDB002;
GRANT SCHEMA TO USER002;
GRANT RDAREA RDAREA01 TO USER002;
```

(5) Specifying the data conversion facility

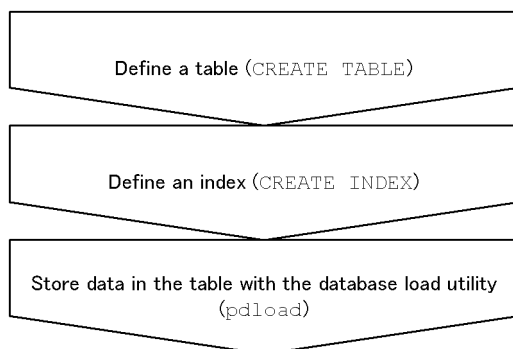
When storing data in a database, you can use a facility to convert the data. Evaluate whether to use the following facilities. For details about these facilities, see the *HiRDB Version 9 System Operation Guide*.

- Space conversion facility
- Facility for conversion to a DECIMAL signed normalized number

6.1.2 Database creation procedure

The following figure shows the procedure for creating a database.

Figure 6–1: Database creation procedure



Note

Execute CREATE TABLE and CREATE INDEX with one of the following methods:

- Database definition utility (pddef)
- HiRDB SQL Executer

6.1.3 Database update log acquisition mode

When storing data in a table using the database load utility (pdload), you can specify a database update log acquisition mode. Use the database load utility's (pdload) -l option to specify a desired database update log acquisition mode.

(1) Types of database update log acquisition mode

There are three different database update log acquisition modes:

- Log acquisition mode
This mode acquires the database update log required for rollback and rollforward. Use this mode when there are relatively few data items.
- Pre-update log acquisition mode
This mode acquires only the database update log required for rollback. Use this mode when there are many data items.

- No-log mode

This mode does not acquire a database update log. Therefore, the data load processing time is the shortest of the three modes. Use this mode when there is only one table per RDAREA (if the table is partitioned, only one row-partitioned table per RDAREA) and any related index is also placed in one RDAREA.

For details about the functionality of these modes, see the *HiRDB Version 9 System Operation Guide*.

(2) Storing data in a user LOB RDAREA

If you are storing data in a user LOB RDAREA, use the `CREATE TABLE`'s `RECOVERY` operand to specify the database update log acquisition mode.

The database update log acquisition mode for user LOB RDAREAs (`RECOVERY` operand of `CREATE TABLE`) may depend on the `-l` option value in `pdload`, as shown in the following table.

Table 6–1: Database update log acquisition mode for user LOB RDAREAs depending on the `-l` option value in `pdload`

Value specified for <code>-l</code> option in <code>pdload</code>	RECOVERY operand value in <code>CREATE TABLE</code>		
	ALL	PARTIAL	NO
a (equivalent to ALL)	ALL	PARTIAL	NO
p (equivalent to PARTIAL)	PARTIAL [#]	PARTIAL [#]	NO
n (equivalent to NO)	NO	NO	NO

ALL: Log acquisition mode

PARTIAL: Pre-update log acquisition mode

NO: No-log mode

For example, if `PARTIAL` is specified in the `RECOVERY` operand of `CREATE TABLE` and the log acquisition method is `NO` in `pdload`, then `NO` (no-log mode) is set for the user LOB RDAREAs.

[#]: For the log that is output by plug-ins, `ALL` (log-acquisition mode) is assumed.

(3) Mode selection considerations

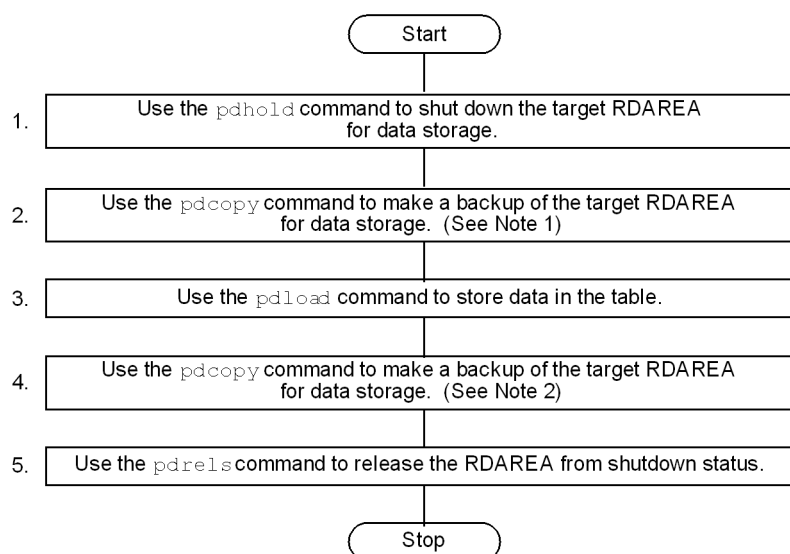
In general, the pre-update log acquisition mode, which is the default mode, should be selected. However, selection of another mode should be considered under the following conditions:

Condition	Mode to be selected
For initial storage, only the target table (or index) for data storage is stored in the RDAREA.	No-log mode
There is large volume of input data, and data storage will take a long time.	
There is only a small amount of input data.	Log acquisition mode

(4) Operational differences

Depending on the mode that is selected, there are differences in the operating procedure during data storage, as shown in the following figure.

Figure 6-2: Differences in operating procedure based on the database update log acquisition mode (data storage)

**Note 1**

This operation is required when the no-log mode is selected. If the `pdload` command should terminate abnormally during operation in the no-log mode, you would use this backup to recover the RDAREA. Note that it is not necessary to make a backup if the conditions described in (5) *When it is not necessary to make a backup prior to data storage* are satisfied.

However, regardless of the update log acquisition mode, a backup should be made when additional data storage is performed in the batch index creation mode on a table for which a plug-in index has been defined. The reason is that in order to perform database recovery after the `pdload` command has terminated abnormally, all the plug-in indexes, including the data portions, must be re-created, which would require a long time for database recovery.

Note 2

This operation is required when the pre-update log acquisition mode or the no-log mode is selected. If a backup is not made at this point, it will not be possible to recover the RDAREA to a subsequent status if it becomes necessary to recover the RDAREA with the `pdrstr` command (target processing after data storage execution cannot be recovered); the RDAREA can be recovered only to its status before data storage was executed.

Supplemental note

If the pre-update log acquisition mode or the no-log mode is selected, the target data storage RDAREA must remain on shutdown status during the steps 1-4 shown in the figure. If the contents of the RDAREA are modified before the backup at step 4 has been made, and if it becomes necessary to recover the RDAREA with the `pdrstr` command, the modified contents will not be recovered; it will be possible to recover the RDAREA only to its status before data storage was executed. If the `pdrstr` command is used to recover the RDAREA, an error will occur during execution of the `pdrstr` command if the system log in the input information contains a log collected in the pre-update log acquisition mode or the no-log mode.

(5) When it is not necessary to make a backup prior to data storage

If data storage is executed in the no-log mode, a backup must have been made prior to execution of data storage. However, if either condition 1 or 2 in the following table is satisfied, it becomes unnecessary to make a backup, because the RDAREA can be returned to its status before execution of data storage even if the `pdload` command terminated abnormally:

No.	Condition		RDAREA recovery method in the event of an error
1	Initial storage	When only the targeted data storage table and indexes of that table are stored in the target data storage RDAREA	If the target data storage RDAREA is reinitialized with the database reconfiguration utility (<code>pdmod</code> command), then once data storage is re-executed the RDAREA can be recovered.
	Data storage executed in the creation mode		

No.	Condition		RDAREA recovery method in the event of an error
2	There exists in the RDAREA targeted for data storage at least one table (index) other than the table (index) targeted for data storage.	When the RDAREA can be recovered to its status before data storage by using a backup and the system log	The RDAREA can be recovered if you use the <code>pdclose</code> command to close the target data storage RDAREA, use the <code>pdlogswap</code> command to swap system log files, and then use the current system log file as input to the database recovery utility (<code>pdrstr</code> command).
	Data storage executed in the addition mode		

Note

With respect to the condition in No. 2, the recovery operation on the RDAREA is easier if a backup has been made, so in general it is recommended that a backup be made. In particular, if `pdload` has terminated abnormally while in the batch index creation mode, indexes cannot be recovered by a rollback even in the log acquisition mode or the in pre-update log acquisition mode. If it is necessary to be able to recover quickly to the status before data storage was attempted in the event of abnormal termination of `pdload`, you should definitely make a backup.

6.1.4 Notes on data storage for a table for which an index with the unique attribute has been defined

The following must be considered when you execute data storage on a table for which a primary key index (`PRIMARY` index) or a unique index (index with the `UNIQUE` specification) has been defined:

- If the input data file contains data with duplicated key values, do not load data in the batch index creation mode.

If you attempt to load data in the batch index creation mode, the system stores data in the table and outputs the index key information to an index information file. At this point, key values are not checked for any duplication. Key value duplication checking occurs later when the index data is stored. If a duplicated key value is detected, index creation processing is rolled back, but the data has already been stored (already committed and cannot go back). In these cases, you need to use a backup copy to restore the RDAREAs.

Therefore, if you are loading data from an input data file that contains data with duplicated key values, be sure to specify the index update mode. This mode updates an index each time data is stored. A duplicated key value is immediately detected and the corresponding data is not stored in the database.

You can specify the batch index creation mode and index update mode using the `-i` option of the database load utility (`pdload`). Note that the default value is the batch index creation mode.

6.1.5 Loading a large amount of data (data loading with the synchronization point specification)

If you plan to load a large amount of data to a table, evaluate the use of data loading with the synchronization point specification.

Normally, with data load processing, a transaction cannot be settled until all data store processing is completed. Therefore, a synchronization point dump cannot be validated while the database load utility is executing. If `HiRDB` terminates abnormally while loading a large amount of data, it takes a long time to restart `HiRDB`. To avoid this, you can set a synchronization point at any interval based on the number of data items during data load processing, thereby enabling transaction settlement. This is called a data load with the synchronization point specification.

To perform data loads with the synchronization point specification, specify a line number of a synchronization point (the number of data items after which a synchronization point is to be set) using the database load utility's `option` statement.

Notes

1. When this facility is used, the overall throughput is reduced by the synchronization point processing, compared to when this facility is not used.

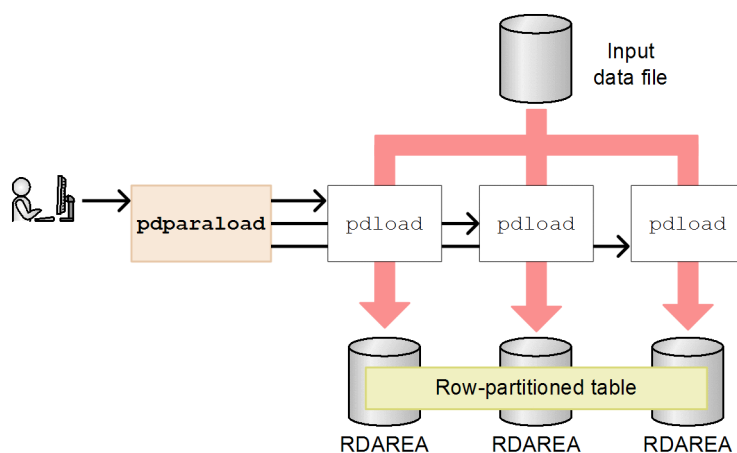
2. If the utility terminates abnormally, the recovery method depends on the termination timing. For details about how to handle abnormal termination, see 6.7 *Handling utility abnormal termination errors during data loading with the synchronization point specification*. Note that the recovery method is complicated if the utility terminates abnormally during a data load in the batch index creation mode.
3. Because data storage begins on a new page for each synchronization point, this facility is applicable only when a large number of pages is stored.

6.1.6 Loading data into a row-partitioned table (using the parallel loading facility)

The parallel loading facility executes, in parallel, data loads from a single input data file to multiple RDAREAs that constitute a row-partitioned table. Executing a `pdparaload` command allows data to be loaded to multiple RDAREAs at once.

The following figure shows an overview of the parallel loading facility.

Figure 6–3: Overview of the parallel loading facility



Description

When the user executes a `pdparaload` command, `pdload` commands equal to the number of RDAREAs that constitute the row-partitioned table are executed automatically. At this time, the `pdparaload` command generates the `pdload` command control statement file.

Each `pdload` command accesses the input data file, extracts the relevant data, and stores it in the row-partitioned table within the RDAREA.

(1) Advantages

Using the parallel loading facility offers the following two advantages.

- Data loads can take less time to process.
When `pdparaload` is executed, multiple `pdload` commands are executed in parallel. For this reason, data loading takes less processing time than loading data by table using a single `pdload` command.
- Operation is easy, since there is a single input data file and a single command.
Through the use of divided-input data files, you can also execute the `pdload` command in parallel and load data by RDAREA. However, this can become complicated because you need to divide the data file and execute the `pdload` command many times. When you use the parallel loading facility, you do not need to divide the input data file by RDAREA. Moreover, you only need to execute the command once.

The parallel loading facility thus offers the dual advantage of fast processing from data loads by RDAREA and easy data loading by table.

(2) Preparing for command execution

Do the following prior to executing parallel loading.

(a) Calculate the resources used

With parallel loading, multiple `pdload` commands execute in parallel. For example, when a table is divided into three RDAREAs for storage, three `pdload` commands are executed simultaneously, so three portions of resources are needed. Factor this in when you calculate the amount of resources that will be necessary.

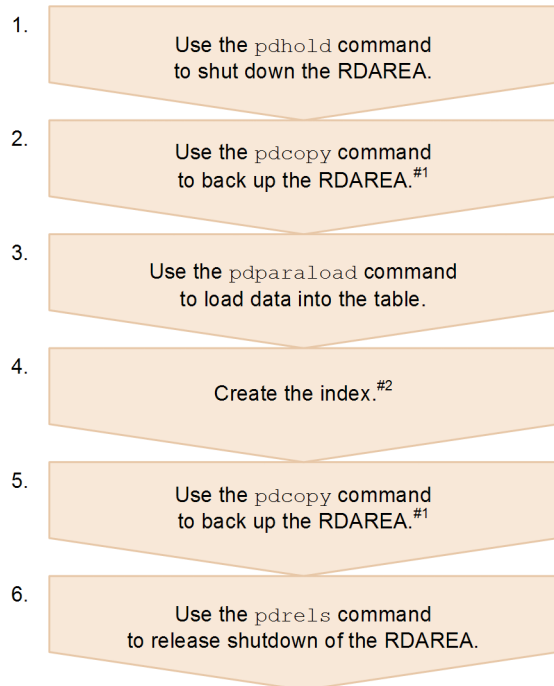
(b) Prepare the input data

Prepare the data to be input to the table as an input data file. The `pdparalload` command uses a single input data file.

(3) Operating procedures

The figure below shows the operating procedures for parallel loading.

Figure 6-4: Parallel loading operating procedure



#1

This step might not be necessary depending on the specification for the database's update log acquisition mode (the `-l` option).

#2

This step is required if `n` (index information output mode) or `x` (index information output suppression mode) was specified as the index creation method (the `-i` option). It is also necessary if `c` (batch index creation mode) was specified and if a non-partitioning key index has been defined.

(4) Limitations

The following limitations apply when data is loaded using the parallel loading facility:

- Data cannot be loaded in tables that define flexible hash partitioning.
- Data loads that specify a synchronization point cannot be performed.
- `NOWAIT` searches cannot be conducted on tables that are executing parallel loading.

- LOB column structure base tables and LOB data cannot be loaded separately.
- Input data files that use tape devices such as EasyMT as the media cannot be used.

`pdparallelload` command options and control statements can be specified in the same manner as `pdload` command options and control statements, with some exceptions. For details about what can be specified, see the section on *pdparallelload* in the manual *HiRDB Version 9 Command Reference*.

(5) Conditions that might affect performance

The performance of the parallel loading facility, such as its allocation of the input data, might be affected by the environment in which data loading is executed. For this reason, in some execution environments data loading might not be any faster even when you use the parallel loading facility. Therefore, try using the parallel loading facility and measure how much time data loading takes before you start actual operations. If this trial shows that using the parallel loading facility will shorten data loading time, use the parallel loading facility. If the facility does not deliver the expected processing performance, load data in table units or divide the input data file and load data in RDAREA units. For details about creating a divided-input data file, see 6.1.7 *Loading data into a row-partitioned table (Creating divided-input data files)*.

The conditions that affect the parallel loading facility's performance are described below.

(a) Row partitioning among servers

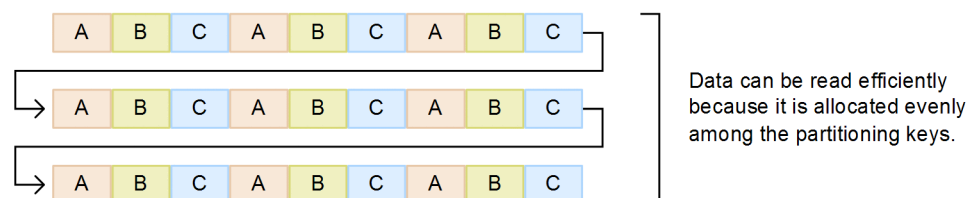
The parallel loading facility is advantageous when you are partitioning rows among servers. Parallel loading can also be executed on tables with row partitioning on a single server. But data loading is faster when it is divided among servers, since processing of a single `pdload` command to store data can monopolize the servers.

(b) Allocation of input data

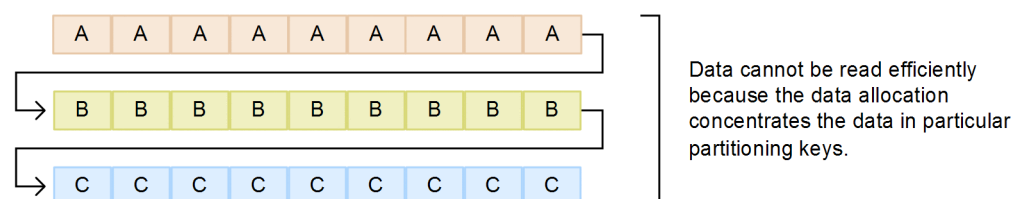
During parallel loading, multiple `pdload` commands read data from a single input data file. Processing time can be shortened at this point by allocating data so as to increase read processing efficiency. As the following figure shows, allocating input data evenly across partitioning keys results in efficient read processing.

Figure 6–5: Allocation of input data vs. read efficiency

• Example of balanced data allocation



• Example of weighted data allocation



Legend: A : Row data corresponding to partitioning condition A
B : Row data corresponding to partitioning condition B
C : Row data corresponding to partitioning condition C

If data from a table that uses row partitioning on one server is unloaded by a `pdorg` command to a single file, the unloaded data file will demonstrate lopsided data allocation, as shown in the figure. For this reason, if these tables are used as input data files, performance might decline despite the use of parallel loading.

(c) Formats for input data

The parallel loading facility allows data loading in all the data formats that can be loaded using the `pdload` command. However, whenever you are working with large quantities of data, we recommend using the binary format for input data. The DAT and fixed-size data formats both require that the format be converted when the data is stored in a database. This might increase CPU usage and decrease the performance of parallel processing of data loading.

(d) LOB creation types

When LOB columns and abstract data type columns that have LOB parameters are defined in the table you are processing, we recommend specifying `f` as the LOB creation type (the `-k` option) for the data load. A file is prepared for each LOB data item as the LOB input file for data loads that specify `f`. For this reason, there is no duplicate reading of LOB data when data loads are executed in parallel. This reduces input and output.

When `d` is specified as the LOB creation type, all LOB data is stored in the input data file. This generates processing to skip reading of LOB data that is not a target of processing when data is loaded in RDAREA units. This increase in processing can weaken performance.

(6) Operating examples

This section provides operating examples that use parallel loading.

- In *Operating example 1*, LOB columns are defined.
- In *Operating example 2*, a non-partitioning key index is defined.

The conditions for these operating examples are as follows.

Table 6–2: Parallel loading operating example conditions

Operating example	Table definition		Index definition		Input data format	Index creation method	Log acquisition mode
	Row partitioning	LOB column	Partitioning key index	Non-partitioning key index			
<i>Operating example 1</i>	Y	Y	Y	N	Binary	Batch index creation mode	Pre-update log acquisition mode
<i>Operating example 2</i>	Y	N	Y	Y	Binary	Batch index creation mode	Pre-update log acquisition mode

Legend

Y: Defined

N: Not defined

(a) Operating example 1 (defining LOB columns)

Data is loaded to TBL2, which defines LOB columns, using the parallel loading facility. For the LOB creation type (`-k` option), `f` is specified and a file is created for each LOB data item.

■ Table and index definitions

Table definition

```
CREATE TABLE "TBL2" (
  col001 int not null,
  col002 varchar(20),
  col003 blob(1M) in ((LOB01), (LOB02), (LOB03)),
  col004 decimal(10,3)
) fix hash hashf by col001
in (RDUSER11, RDUSER21, RDUSER31);
```

Index definition

```
CREATE INDEX "INDX2" ON "TBL2"(col001,col002)
in ((RDUSER12),(RDUSER22),(RDUSER32));
```

■ Data load procedure

1. Use the `pdhold` command to shut down the RDAREA to which you plan to load data.

```
pdhold -r LOB01,LOB02,LOB03,RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32
```

2. Create the `pdparaload` command control statement file.

`source`: Specify the server name, input data file and error information file. For a HiRDB parallel server configuration, always specify the server name.

`lobdata`: Specify the LOB input file.

`idxwork`: Specifies the storage directory for the index information file.

`sort`: Specifies the storage directory of the work file for sorting.

`report`: Specifies the processing results file name.

```
source fes01:/users/data/input_file error=/users/rep/error_file
lobdata /users/data/lob
idxwork bes01 /users/work
idxwork bes02 /users/work
idxwork bes03 /users/work
sort bes01 /users/work
sort bes02 /users/work
sort bes03 /users/work
report file=/users/rep/result_file
```

3. Use the `pdparaload` command to load data.

`-d`: Load data in creation mode.

`-b`: Use a binary input data file.

`-k`: Specify the LOB creation type. Create a file for each LOB data item (f).

`-i`: Specify the batch index creation mode (c).

`-l`: Specify the pre-update log acquisition mode (p).

```
pdparaload -d -b -k f -i c -l p "TBL2" /users/cntl/control_file
```

4. Use the `pdcopy` command to make a backup of the RDAREA.

```
pdcopy -m /users/rdarea/mast/mast01 -M r -p /users/pdcopy/list/list01
-b /users/pdcopy/backup/backup01
-r LOB01,LOB02,LOB03,RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32
```

5. Use the `pdrels` command to release the RDAREA from shutdown status.

```
pdrels -r LOB01,LOB02,LOB03,RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32
```

(b) Operating example 2 (defining a non-partitioning key index)

Load data to TBL3, which defines a non-partitioning key index, using the parallel loading facility. Since a non-partitioning key index is being defined, batch creation of the index using the `pdorg` command is required after you execute the `pdparaload` command. For batch creation of the index, you specify the index information file output by the `pdparaload` command.

■ Table and index definitions

Table definition

```
CREATE TABLE "TBL3"(
  col001    int not null,
  col002    varchar(20),
  col003    char(32),
  col004    decimal(10,3)
) partitioned by col001
in ((RDUSER11) 10000000, (RDUSER21) 20000000, (RDUSER31));
```

Index definition (partitioning key index)

```
CREATE INDEX "INDX3" ON "TBL3" (col001,col002)
in ((RDUSER12),(RDUSER22),(RDUSER32));
```

Index definition (non-partitioning key index)

```
CREATE INDEX "INDX4" ON "TBL3" (col003)
in ((RDUSER13));
```

Data load procedure

1. Use the `pdhold` command to shut down the RDAREA to which you plan to load data.

```
pdhold -r RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32,RDUSER13
```

2. Create the `pdparaload` command control statement file.

source: Specify the input data file and the error information file. For a HiRDB parallel server configuration, always specify the server name.

idxwork: Specify the storage directory for the index information file. After data loading, execute batch creation of the index using the index information file created here.

sort: Specify the storage directory of the work file for sorting.

report: Specify the processing results file name.

```
source /users/data/input_file error=/users/rep/error_file
idxwork /users/work
sort /users/work
report file=/users/rep/result_file
```

3. Use the `pdparaload` command to load data.

-d: Load data in creation mode.

-b: Use a binary input data file.

-i: Specify the batch index creation mode (c).

-l: Specify the pre-update log acquisition mode (p).

```
pdparaload -d -b -i c -l p "TBL3" /users/cntl/control_file
```

4. Create the `pdrorg` command control statement file.

In the index statement, specify the index information file output by the `pdparaload` command.

```
index "INDX4" RDUSER13 "/users/work/INDEX-INDX4-RDUSER13-faaG4MnMf"
index "INDX4" RDUSER13 "/users/work/INDEX-INDX4-RDUSER13-caa34EnEc"
index "INDX4" RDUSER13 "/users/work/INDEX-INDX4-RDUSER13-caa06MnMc"
sort /users/work
report file=/users/rep/result_file2
```

5. With the `pdrorg` command, execute batch creation of the index.

```
pdrorg -k ixmk -l p -t "TBL3" /users/cntl/control_rorg
```

6. Use the `pdcopy` command to make a backup of the RDAREA.

```
pdcopy -m /users/rdarea/mast/mast01 -M r -p /users/pdcopy/list/list01
-b /users/pdcopy/backup/backup01
-r RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32,RDUSER13
```

7. Use the `pdrels` command to release the RDAREA from shutdown status.

```
pdrels -r RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32,RDUSER13
```

(7) What to do when a command error occurs

There are two basic types of errors that can occur when the `pdparaload` command is executed, and two corresponding ways to handle them, as follows.

- Errors when executing data loads in RDAREA units (`pdload` command)

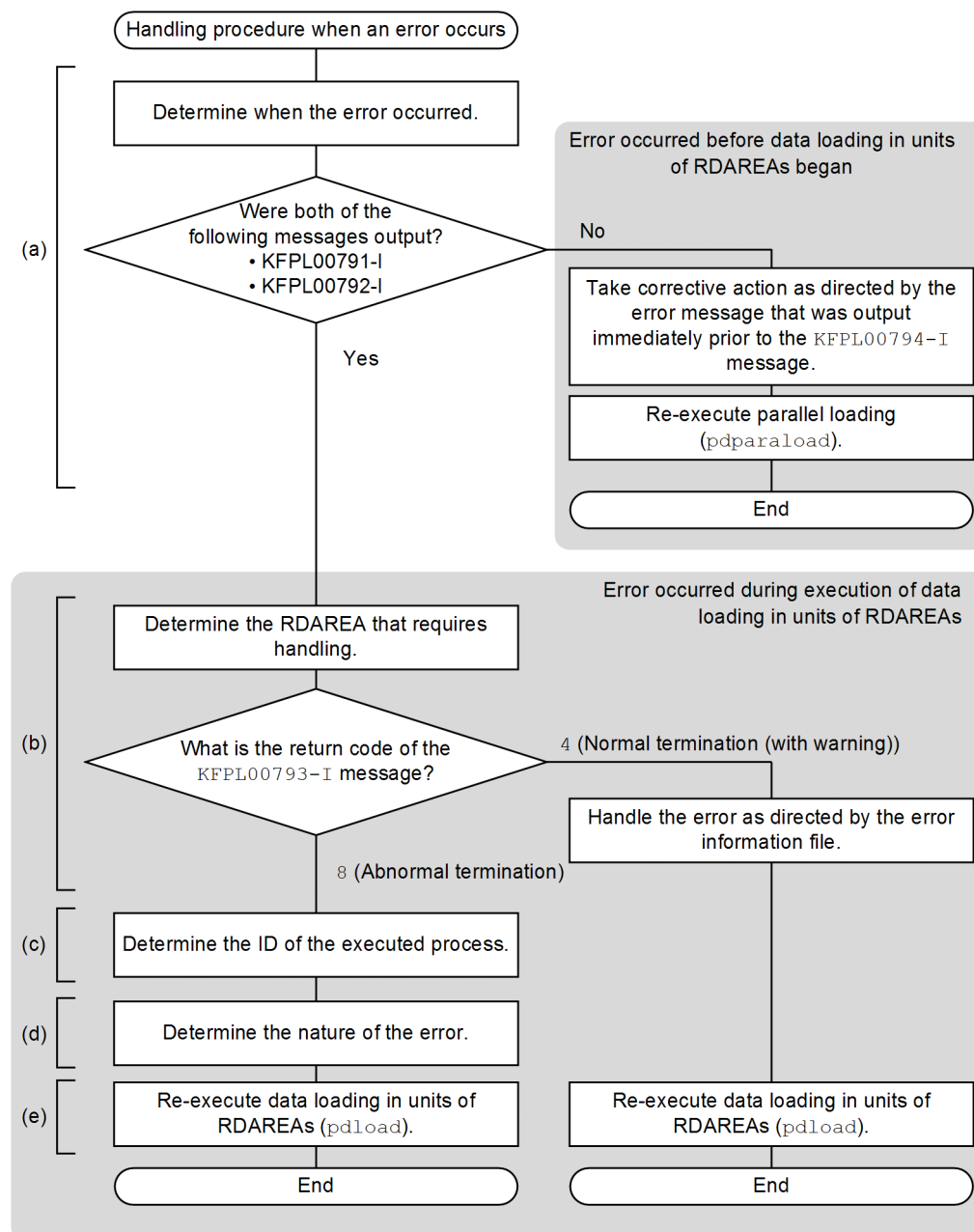
In this case, data fails to load into some RDAREAs. After handling the error for those RDAREAs that experienced errors, re-execute the data load in RDAREA units with the `pdload` command.

- Errors before data loads in RDAREA units (`pdload` command) begin

In this case, data loads to all RDAREAs fail. After handling the error, re-execute the `pdparalload` command.

This section describes details about errors generated when the `pdparalload` command is executed and their handling. The handling procedure when an error occurs when the `pdparalload` command is executed is shown below.

Figure 6–6: Handling procedure when an error occurs during `pdparalload` command execution



Note:

Sections (a) through (e) in the figure correspond to (a) through (e) in the figure.

(a) Identify error timing

Identify when the error occurred. Check whether the following two messages have been output.

- KFPL00791-I
- KFPL00792-I

Both messages have been output

An error occurred during execution of a data load in RDAREA units. This means data loads to some RDAREAs have failed. In this case, handle the error by starting with step (b) of the procedure. Thereafter, re-execute the data load in RDAREA units with the `pdload` command.

One of the messages has not been output

An error occurred before data loading in RDAREA units began. This means data loads to all RDAREAs have failed. In this case, handle as directed by the error message output directly prior to the KFPL00794-I message. Thereafter, re-execute the `pdparaload` command.

(b) Identify RDAREAs that require handling

Extract the KFPL00793-I message from the messages output to the OS console that executed the `pdparaload` command or to the syslog file, and then identify the RDAREAs that require handling. When extracting the message, use the following as search keywords.

- Message ID (KFPL00793-I)
- Authorization identifier
- Table identifier

Message extraction example

```
KFPL00793-I Pdload execution abnormal terminated, table=user1."T1", RDAREA=RDAREA1, return
code=8, pdload process id=1385412 (1212478)
:
KFPL00793-I Pdload execution abnormal terminated, table=user1."T1", RDAREA=RDAREA2, return
code=8, pdload process id=1385433 (1212478)
```

Description

Extract the KFPL00793-I message based on the authorization identifier (`user1`) and the table identifier (`T1`). This can identify the RDAREAs (RDAREA1 and RDAREA2) where the data load failed.

Take one of the following corrective actions for the identified RDAREA, depending on the return code of the KFPL00793-I message.

Return code is 4

Check the error information file output by the `pdload` command and take corrective action for the error. The error information file is specified by the `error` option of the `source` statement of the `pdparaload` command (the RDAREA name is automatically appended to the file name). When nothing is specified, the following directory and file name are used to create the file.

- The error information file storage directory
 1. The directory specified in the `pd_tmp_directory` operand
 2. If the above directory is not specified, the directory specified in environment variable `TMPDIR`
 3. If the above directory is not specified, the `/tmp` directory

- The error information file name

The file name that includes the process ID and message ID (KFPL00793-I) identified in (c)

After corrective action is taken, re-execute the data load in units of RDAREAs using the procedure of step (e).

Return code is 8

Identify the error as instructed in steps (c) and (d) and take corrective action.

Then, re-execute the data load in units of RDAREAs using the procedure of step (e).

(c) Identify the IDs of the processes that were executed

From the message log file, identify the IDs of the pdload control process and server process that the pdparaload command executed. Extract the process IDs by the following procedure.

1. Output the message log from pdparaload command execution start (KFPL00791-I message) to termination (KFPL00794-I message) to a file using the pdcat command.
2. Extract the KFPL00711-I message, which indicates the start of the process, from the content output in step 1. To extract the message, use the following search keywords.
 - Message ID (KFPL00711-I)
 - Authorization identifier
 - Table identifier
 - RDAREA name identified in step (b)

Message extraction example

```
1572976 2010/11/04 15:55:48 0mload1 lod KFPL00711-I pdloadm started, table=user1."T1",
RDAREA=RDAREA1
1728690 2010/11/04 15:55:48 bes1 lod KFPL00711-I pdbes started, table=user1."T1",
RDAREA=RDAREA1
```

Description

Extract the KFPL00711-I message based on the authorization identifier (user1), the table identifier (T1) and the RDAREA name (RDAREA1). This can identify the IDs of the processes (1572976, 1728690) that generated the error.

In this case, these IDs identify the following processes.

- The process name of 1572976 is pdloadm, so it is the pdload control process.
- The process name of 1728690 is pdbes, so it is the back-end server process.

(d) Identify the error content

Identify the nature of the error from the message log file. Extract the error message that requires handling from the message log file acquired in step 1 of (c) using the process IDs identified in step (c) as search keywords.

Message extraction example (pdload control process)

```
1572976 2010/11/04 15:55:48 0mload1 lod KFPL00711-I pdloadm started, table=user1."T1",
RDAREA=RDAREA1
1572976 2010/11/04 15:55:48 0mload1 lod KFPL00704-I Pdload terminated, return code=8
```

Message extraction example (back-end server process)

```
1728690 2010/11/04 15:55:48 bes1 lod KFPL00711-I pdbes started, table=user1."T1",
RDAREA=RDAREA1
1728690 2010/11/04 15:55:48 bes1 lod KFPL00709-I Error information file was created,
file=/tmp/ERROR-4cd258f41728690
1728690 2010/11/04 15:55:48 bes1 lod KFPL00709-I Lobmid file was created, file=/tmp/LOBMID-
T1-4cd258f41728690
1728690 2010/11/04 15:55:48 bes1 lod KFPL00702-I Pdload started, table=user1."T1",
generation=0
1728690 2010/11/04 15:55:48 bes1 lod KFPL00710-I Index information file assigned,
index=user1."T1NX", RDAREA="USER01", file=/tmp/INDEX-T1NX-USER01-GN0-daan_ylid
1728690 2010/11/04 15:55:48 bes1 lod KFPL00723-I 0 rows loaded, table=user1."T1",
RDAREA="USER01"
1728690 2010/11/04 15:55:48 bes1 lod KFPLxxxxx-E YYYYYY
```

Description

Extract messages based on the process IDs (1572976, 1728690). Here, a back-end server process has output the error message (KFPLxxxx-E). Take corrective action as indicated by the error message.

After corrective action is taken, re-execute the data load in units of RDAREAs according to the procedure of step (e).

(e) Re-execute the data load in units of RDAREAs (pdload command)

Re-execute the data load in units of RDAREAs using the `pdload` command. Here, we recommend re-using the `pdload` command control statement generated by the `pdparaload` command. The storage directory and file name of the `pdload` command control statement file generated by the `pdparaload` command are as follows. Create the `pdload` control statement file based on these.

- The `pdload` control statement storage directory
 1. The directory specified in the `pd_tmp_directory` operand
 2. If the above directory is not specified, the directory specified in environment variable `TMPDIR`
 3. If the above directory is not specified, the `/tmp` directory

- The `pdload` control statement file name

`LOD_CTL_authorization-ID_table-ID_RDAREA-NAME`

! Important note

When re-executing the data load in units of RDAREAs, specify `divermsg=off` in the option statement of `pdload`.

When loading data in RDAREA units, error information will be output when there is line data in the input data that does not match the RDAREA storage conditions, and the `pdload` command will terminate with a return code of 4. If `divermsg=off` is specified, output of this error data information is suppressed, and the `pdload` command can terminate with a return code of 0.

The following is an example of the `pdload` command control statements and command lines when re-executing a data load in units of RDAREAs.

Control statement example

```
option divermsg=off
source "RDUSER02" fes01:/users/data/input_file error=/users/rep/error_file_RDUSER02
lobdata /users/data/lob
idxwork bes02 /users/work
sort bes02 /users/work
report file=/users/rep/result_file_RDUSER02
```

Description

- Specifying `divermsg=off` in the option statement suppresses output of error data information.
- Specify the name of the RDAREA that is the target of the `source` statement for data loading in units of RDAREAs.

Command line example

```
pdload -d -b -k f -i c -l p "TBL2" "/users/tmp/LOD_CTL_USER02_ TBL2_RDUSER02"
```

6.1.7 Loading data into a row-partitioned table (Creating divided-input data files)

Data can also be loaded to a row-partitioned table using the parallel loading facility. For details about the parallel loading facility, see *6.1.6 Loading data into a row-partitioned table (using the parallel loading facility)*. If you cannot achieve the desired results using the parallel loading facility, load data using divided-input data files, as described in this subsection.

When you load data into a row-partitioned table, you can reduce the time required for data loading and the length of time the table is in the exclusive mode by dividing the input data files by storage RDAREAs and executing parallel data loading. By specifying the `src_work` statement in the control information file and then executing the database load utility (`pdload`), you can create an input data file for each RDAREA from user-created input data files. The obtained input data files can be used to execute data loading for the individual RDAREAs. A file created by the database load utility (`pdload`) is called a *divided-input data file*. For details about the options and control statements used to create the divided-input data files, see the manual *HiRDB Version 9 Command Reference*.

6.1.8 Data loads that use the automatic numbering facility

Use the sequence generator identifier to number automatically. This is called the *automatic numbering facility*. When data is loaded, sequence numbers generated by the sequence generator identifier can be stored in the table columns. This section describes selection criteria for the acquisition methods and storage methods of sequence numbering.

For details about the automatic numbering facility, see the *HiRDB Version 9 UAP Development Guide*; for details about loading data using the automatic numbering facility, see the manual *HiRDB Version 9 Command Reference*.

(1) Criteria for selecting sequence number acquisition method

There are three methods for acquiring sequence numbers.

Number batch acquisition method:

After data has loaded, uses the sequence generator identifier values to number the sequence as a batch.

Specification unit acquisition method:

Loads data while acquiring sequence numbers at every specified unit.

Buffer unit acquisition method:

Loads data while acquiring sequence numbers in batches equal to the number of lines that can be read into the input buffer.

When selecting a method of acquiring sequence numbers, consider the features listed in the following table.

Table 6–3: Features of sequence number acquisition methods

Item considered	Features		
	Number batch acquisition method	Specification unit acquisition method	Buffer unit acquisition method
Missing numbers under normal circumstances	Does not occur.	Missing numbers occur if the number of lines of data loaded is not a multiple of the specified unit.	Does not occur. ^{#2}
Large quantities of missing numbers during rollbacks	Does not occur.	The current value is not recovered even if a rollback occurs, so large quantities of missing numbers occur.	The current value is not recovered even if a rollback occurs, so large quantities of missing numbers occur.
Communication overhead when requesting numbering from sequence generator identifier ^{#1}	Since there are only as many numbering requests as there are data load commits, the impact of numbering on performance is kept to a minimum.	The impact on performance can be kept to a minimum by keeping the number of numbering requests down by using large acquisition units.	The impact on performance can be kept to a minimum by keeping the number of numbering requests down by using a large input buffer. However, the unit acquired at one time is determined by the line length calculated from the input buffer size and the column definition size, so a long line requires ample memory in the input buffer.
Simultaneous execution with UAPs that use the same sequence generator identifier	Cannot execute simultaneously. The sequence generator identifier is locked during data loading.	Can execute simultaneously.	Can execute simultaneously.
Parallel execution of data loading in RDAREA units	Cannot execute simultaneously. The sequence generator identifier is locked during data loading.	Can execute in parallel.	Can execute in parallel.

^{#1}

For a HiRDB parallel server configuration, communications occur during acquisition of sequence numbers when the server whose database load utility reads input data is different from the server where the sequence generator identifier is defined.

Consequently, frequent numbering requests result in increased communications, which adversely affects data loading performance.

#2

When the sequence number storage method is not total replacement of column data, missing numbers might be generated.

(2) Criteria for selecting a sequence number storage method

There are three methods for storing sequence numbers. The following describes the criteria to use when you select a method.

Total replacement of column data:

All column data in the input data file is replaced by sequence numbers for the corresponding columns that store sequence numbers. Select this method when you are assigning new numbers to all the corresponding column values.

Partial replacement of column data:

Only the column data in the input data file whose data matches specified replacement conditions is replaced by sequence numbers for the corresponding columns that store sequence numbers. Select this method when, for example, the input data file is in DAT or extended DAT format and when you are replacing only the NULL value portion with sequence numbers.

Column data addition:

If the input data file has no data corresponding to the column that stores the sequence numbers, sequence numbers are added as input data. Select this method when you are adding new columns to store numbers. This method cannot be specified when the input data file format is binary.

6.1.9 Input data file UOC

You can use a user-created program to edit data that is to be loaded. The edited data is passed directly to `pdload`. Therefore, programs that edit input files can perform data loading without creating temporary work files.

A program that the user creates to edit data is called the user's own coding (UOC). You can use a UOC to edit input data, such as when the format of a file containing database data is different from the input data file format supported by `pdload`, or when the character encoding used in the database data is not supported by HiRDB.

6.1.10 Deleting unneeded RDAREAs

After creating a database, you should check the `SQL_RDAREAS` table of the data dictionary tables for any user RDAREAs for which no table or index is defined, or for any user LOB RDAREAs for which no LOB column is defined. You can delete any unneeded RDAREAs and save disk space.

For details about how to retrieve data dictionary tables and for details about the `SQL_RDAREAS` table, see the *HiRDB Version 9 UAP Development Guide*. For details about how to delete RDAREAs, see the *HiRDB Version 9 System Operation Guide*.

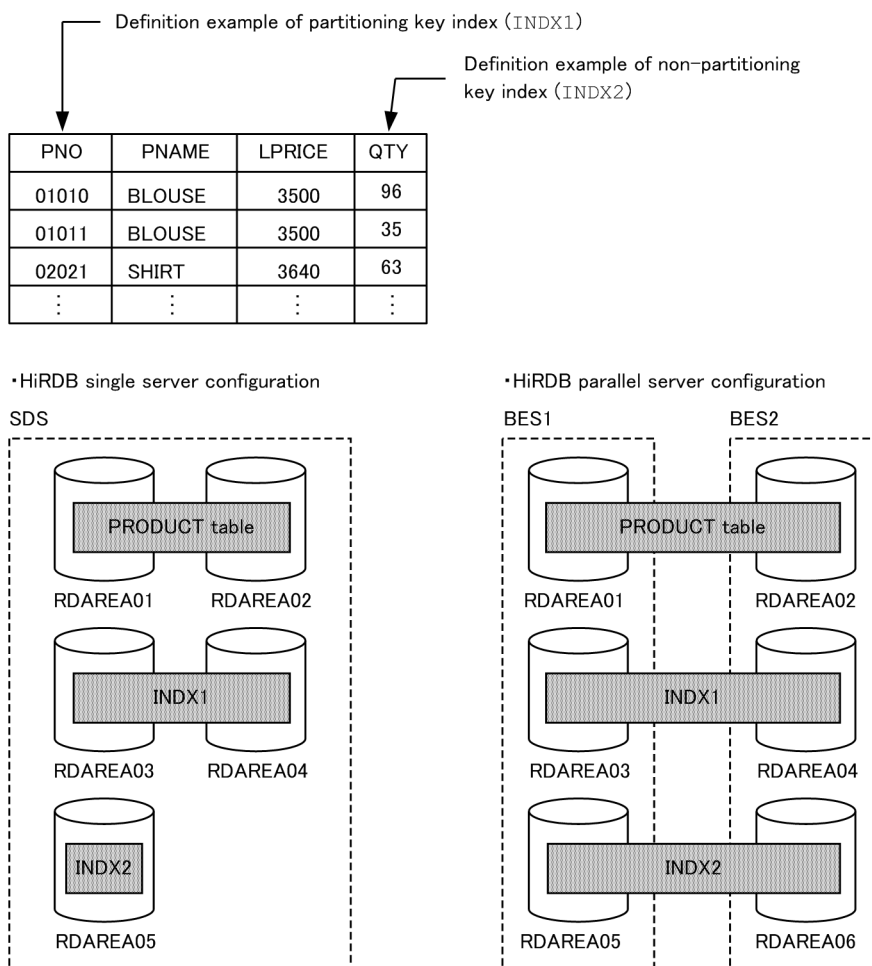
6.2 Creating a row-partitioned table

This section describes the creation of a `PRODUCT` table. The following are the creation conditions:

- Partition the `PRODUCT` table by row. Store the `PRODUCT` table in user RDAREAs `RDAREA01` and `RDAREA02`.
- Define a partitioning key index (`INDX1`) for the `PRODUCT` table. Store `INDX1` in user RDAREAs `RDAREA03` and `RDAREA04`.
- Define a non-partitioning key index (`INDX2`) for the `PRODUCT` table. Store `INDX2` in user RDAREAs `RDAREA05` and `RDAREA06`.
- Store by means of initial storage in `RDAREA01`–`RDAREA06` only the target data storage table (and indexes).
- In executing data storage, use batch creation (the default value) for the indexes.
- Perform data storage in the no-log mode.

For details about the partitioning key index and non-partitioning key index, see *13.3 Index row partitioning*.

Data can also be loaded to a row-partitioned table using the parallel loading facility. For details about the parallel loading facility, see *6.1.6 Loading data into a row-partitioned table (using the parallel loading facility)*.



Explanation:

SDS: Single server

BES: Back-end server

(1) Defining the PRODUCT table

Define the PRODUCT table with CREATE TABLE. The following shows an example:

(a) Key range partitioning

Specification of storage condition:

```
CREATE TABLE PRODUCT
(PNO CHAR(5) NOT NULL,
 PNAME NCHAR(15),
 LPRICE INTEGER,
 QTY INTEGER
) IN ((RDAREA01) PNO<='10000', (RDAREA02));
```

Specification of boundary value:

```
CREATE TABLE PRODUCT
(PNO CHAR(5) NOT NULL,
 PNAME NCHAR(15),
 LPRICE INTEGER,
 QTY INTEGER
) PARTITIONED BY PNO
IN ((RDAREA01) '10000', (RDAREA02));
```

(b) Flexible hash partitioning or FIX hash partitioning

```
CREATE TABLE PRODUCT
(PNO CHAR(5) NOT NULL,
 PNAME NCHAR(15),
 LPRICE INTEGER,
 QTY INTEGER
) [FIX]# HASH HASH6 BY PNO
IN (RDAREA01,RDAREA02);
```

#: This specification is applicable to FIX hash partitioning.

(2) Defining an index

Define an index for the PRODUCT table using CREATE INDEX. The following shows an example:

(a) HiRDB single server configuration

```
CREATE INDEX INDX1 ON TABLE (PNO)
IN ((RDAREA03), (RDAREA04));
CREATE INDEX INDX2 ON PRODUCT (QTY)
IN (RDAREA05);
```

(b) HiRDB parallel server configuration

```
CREATE INDEX INDX1 ON PRODUCT (PNO)
IN ((RDAREA03), (RDAREA04));
CREATE INDEX INDX2 ON PRODUCT (QTY)
IN ((RDAREA05), (RDAREA06));
```

(3) Storing data in the table

To use the database load utility (pdload) to store data in the table.

Procedure

1. Use the pdhold command to shut down the target data storage RDAREAs (RDAREA01–RDAREA05). For a HiRDB parallel server configuration, shut down RDAREA01–RDAREA06.
2. Use the pdload command to load the input data file into the table. Because only the target data storage table and indexes are stored in the RDAREAs, and because this is the initial storage, select the no-log mode as the database update log acquisition mode. For the index creation method, select the batch index creation mode

(the default value). For details about the options of the `pdload` command, see the manual *HiRDB Version 9 Command Reference*.

3. Because the `pdload` command is executed in the no-log mode, make a backup of the target data storage RDAREAs. For details about how to make backups in units of RDAREAs, see the *HiRDB Version 9 System Operation Guide*.
4. Use the `pdrels` command to release the target data storage RDAREAs from shutdown status.

For details about these commands and utilities, and about how to verify the command and utility execution results, see the manual *HiRDB Version 9 Command Reference*.

Supplemental notes

- Because the `pdload` command executes in the no-log mode, the target data storage RDAREAs must remain on shutdown status during steps 1-3.
- In the case of a falsification prevented table, when data storage is performed with the `pdload` command, the `-d` option cannot be specified.
- For details about error handling during batch index creation, see *6.6 Handling errors during batch index creation*.

(4) Checking the data storage status

If you have executed data loading, you should execute the database condition analysis utility (`pddbstat`) next to check the data storage status. This utility enables you to check whether the database has been created exactly as designed. The database condition analysis utility can obtain the following information:

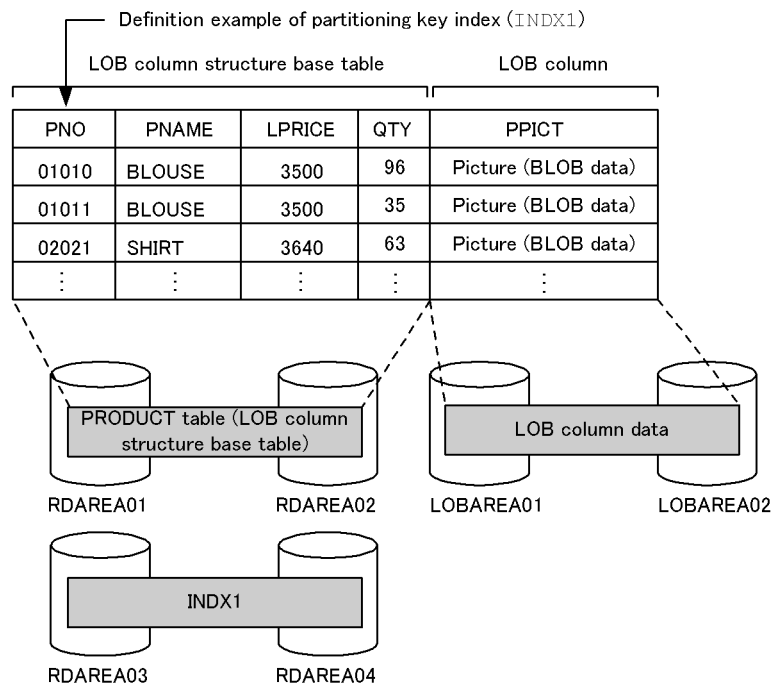
- Data storage status of each user RDAREA
- Data storage status of each table or index

6.3 Creating a table with a LOB column

This section describes the creation of a `PRODUCT` table. The following are the creation conditions:

- Partition the `PRODUCT` table by row. Store the `PRODUCT` table's LOB column structure base table in user RDAREAs `RDAREA01` and `RDAREA02`.
- Store LOB column data in user LOB RDAREAs `LOBAREA01` and `LOBAREA02`.
- Define a partitioning key index (`INDX1`) for the `PRODUCT` table. Store `INDX1` in user RDAREAs `RDAREA03` and `RDAREA04`.
- In `RDAREA01`–`RDAREA02` and `LOBAREA01`–`LOBAREA02`, perform initial storage only for the target data storage table (and index).
- When executing data storage, perform batch creation (the default value) on the index.
- Perform data storage in the no-log mode.

Data can also be loaded into a row-partitioned table using the parallel loading facility. For details about the parallel loading facility, see 6.1.6 *Loading data into a row-partitioned table (using the parallel loading facility)*.



Notes

- Only one LOB column is stored in a user LOB RDAREA. If a table contains multiple LOB columns, they must be stored in separate user LOB RDAREAs.
- For a row-partitioned table that has LOB columns, there must be a one-to-one correspondence between the user LOB RDAREAs for the LOB columns and the user RDAREAs for storing the table.

(1) Defining the `PRODUCT` table

Define the `PRODUCT` table with `CREATE TABLE`. The following shows an example:

(a) Key range partitioning

Specification of storage condition:

```
CREATE TABLE PRODUCT
(PNO CHAR(5),
```

```

PNAME NCHAR(15),
LPRICE INTEGER,
QTY INTEGER,
PPICT BLOB(64K) IN ((LOBAREA01), (LOBAREA02))
) IN ((RDAREA01) PNO<='10000', (RDAREA02));

```

Specification of boundary value:

```

CREATE TABLE PRODUCT
(PNO CHAR(5),
PNAME NCHAR(15),
LPRICE INTEGER,
QTY INTEGER,
PPICT BLOB(64K) IN ((LOBAREA01), (LOBAREA02))
) PARTITIONED BY PNO
IN ((RDAREA01) '10000', (RDAREA02));

```

(b) Flexible hash partitioning or FIX hash partitioning

```

CREATE TABLE PRODUCT
(PNO CHAR(5),
PNAME NCHAR(15),
LPRICE INTEGER,
QTY INTEGER,
PPICT BLOB(6000) IN ((LOBAREA01), (LOBAREA02))
) [FIX]# HASH HASH6 BY PNO
IN (RDAREA01, RDAREA02);

```

#: This specification is applicable to FIX hash partitioning.

(2) Defining an index

Define an index for the `PRODUCT` table using `CREATE INDEX`. The following shows an example:

```

CREATE INDEX INDX1 ON PRODUCT (PNO)
IN ((RDAREA03), (RDAREA04));

```

(3) Storing data in the table

To use the database load utility (`pdload`) to store data in the table.

Procedure

1. Use the `pdhold` command to shut down the target data storage RDAREAs (RDAREA01–RDAREA04 and LOBAREA01–LOBAREA02).
2. Use the `pdload` command to load the input data file into the table. Because only the target data storage table and index are stored in the RDAREAs, and because this is an initial storage, select the no-log mode as the database update log acquisition mode. For the index creation method, select the batch index creation mode (the default value). For details about the options of the `pdload` command, see the manual *HiRDB Version 9 Command Reference*.
3. Because the `pdload` command is executed in the no-log mode, make a backup of the target data storage RDAREAs. For details about how to make backups in units of RDAREAs, see the *HiRDB Version 9 System Operation Guide*.
4. Use the `pdrels` command to release the target data storage RDAREAs from shutdown status.

For details about these commands and utilities, and about how to verify the command and utility execution results, see the manual *HiRDB Version 9 Command Reference*.

Supplemental notes

- Because the `pdload` command executes in the no-log mode, the target data storage RDAREAs must remain on shutdown status during steps 1-3.
- In the case of a falsification prevented table, when data storage is performed with the `pdload` command, the `-d` option cannot be specified.

- For details about error handling during batch index creation, see *6.6 Handling errors during batch index creation*.

(4) Checking the data storage status

If you have executed data loading, you should execute the database condition analysis utility (`pddbstat`) next to check the data storage status. This utility enables you to check whether the database has been created exactly as designed. The database condition analysis utility (`pddbstat`) can obtain the following information:

- Data storage status of each user RDAREA or user LOB RDAREA
- Data storage status of each table or index

(5) Notes

When executing data loading on a table with a LOB column, you can load the LOB column structure base table and LOB data separately.

Set the database update log acquisition mode to the no-log mode, and set the index creation method to batch index creation (the default value).

Procedure

1. Use the `pdhold` command to shut down the target data storage RDAREAs (RDAREA01–RDAREA04 and LOBAREA01–LOBAREA02).
2. Use the `pdload` command to load the input data file into the table (LOB column structure base table and index). At this time, the target data storage RDAREAs are RDAREA01–RDAREA04. Output to the LOB middle file the information required for data storage of the LOB column. For details about the options of the `pdload` command, see the manual *HiRDB Version 9 Command Reference*.
3. Use the `pdload` command to perform data storage in user LOB RDAREAs LOBAREA01–LOBAREA02. Specify the LOB input file and the LOB middle file specified in step 2.
4. Because the `pdload` command is executed in the no-log mode, make a backup of the target data storage RDAREAs. For details about how to make backups in units of RDAREAs, see the *HiRDB Version 9 System Operation Guide*.
5. Use the `pdrels` command to release the target data storage RDAREAs from shutdown status.

For details about these commands and utilities, and about how to verify the command and utility execution results, see the manual *HiRDB Version 9 Command Reference*.

6.4 Creating a table containing a plug-in-provided abstract data type

This section describes the procedure for creating tables that define abstract data types (the `SGMLTEXT` and `XML` types) provided by plug-ins.

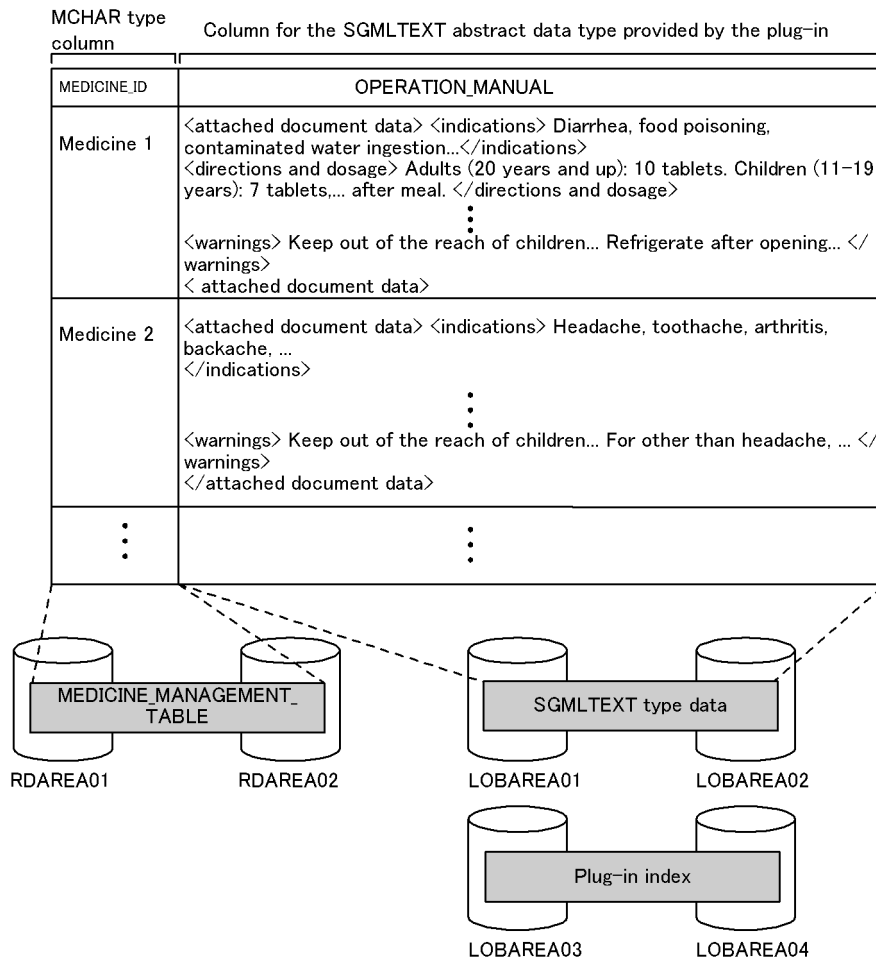
To use the `SGMLTEXT` type, you will need HiRDB Text Search Plug-in; to use the `XML` type, you will need HiRDB XML Extension. For details about environment settings for plug-ins, see *Chapter 5. Setting Up the Plug-in Environment*. The plug-in owner should be `MASTER`.

6.4.1 The `SGMLTEXT` type

This section describes the procedure for creating a table with abstract data type (`SGMLTEXT` type) that is provided by the HiRDB Text Search Plug-in.

A `MEDICINE_MANAGEMENT_TABLE` is created here. The following are the creation conditions:

- Partition the `MEDICINE_MANAGEMENT_TABLE`. Store the LOB column structure base table in user RDAREAs `RDAREA01` and `RDAREA02`.
- Store `SGMLTEXT`-type column data in user LOB RDAREAs `LOBAREA01` and `LOBAREA02`.
- Store the plug-in index in user LOB RDAREAs `LOBAREA03` and `LOBAREA04`.
- In `RDAREA01`–`RDAREA02` and `LOBAREA01`–`LOBAREA04`, perform initial storage only for the target data storage table (and index).
- When executing data storage, perform batch creation (the default value) on the index.
- Perform data storage in the no-log mode.



Explanation:

This example stores MEDICINE_ID (MCHAR type) in the user RDAREAs and OPERATION_MANUAL (SGMLTEXT type) in the user LOB RDAREAs.

(1) Defining the MEDICINE_MANAGEMENT_TABLE

Define the MEDICINE_MANAGEMENT_TABLE with CREATE TABLE. The following shows an example:

(a) Key range partitioning

Specification of storage condition:

```
CREATE TABLE MEDICINE_MANAGEMENT_TABLE (
  MEDICINE_ID MCHAR(15),
  OPERATION_MANUAL SGMLTEXT
  ALLOCATE(SGMLTEXT IN((LOBAREA01),(LOBAREA02))) 1.
  PLUGIN'<DTD>medicine.dtd</DTD>' 2.
) IN((RDAREA01)MEDICINE_ID<='MEDICINE 10',(RDAREA02)); 3. 4.
```

Specification of boundary value:

```
CREATE TABLE MEDICINE_MANAGEMENT_TABLE (
  MEDICINE_ID MCHAR(15),
  OPERATION_MANUAL SGMLTEXT
  ALLOCATE(SGMLTEXT IN((LOBAREA01),(LOBAREA02))) 1.
  PLUGIN'<DTD>medicine.dtd</DTD>' 2.
) PARTITIONED BY MEDICINE_ID 3.
IN((RDAREA01)'MEDICINE 10',(RDAREA02)); 4.
```

Explanation:

1. Specifies the data type provided by the plug-in module.
2. The SGMLTEXT LOB column in MEDICINE_MANAGEMENT_TABLE is divided among and stored in user LOB RDAREAs LOBAREA01 and LOBAREA02.
3. Specifies the plug-in portion. For details about specification, see the applicable plug-in documentation.
4. MEDICINE_MANAGEMENT_TABLE's LOB column structure base table is divided and stored in user RDAREAs RDAREA01 and RDAREA02.

(b) Flexible hash partitioning or FIX hash partitioning

```
CREATE TABLE MEDICINE_MANAGEMENT_TABLE (
  MEDICINE_ID MCHAR(15),
  OPERATION_MANUAL SGMLTEXT
  ALLOCATE(SGMLTEXT IN((LOBAREA01), (LOBAREA02)))
  PLUGIN'<DTD>medicine.dtd</DTD>'
) [FIX] # HASH HASH6 BY MEDICINE_ID
IN (RDAREA01, RDAREA02)
```

1.
2.
3.
4.

#: This specification is applicable to FIX hash partitioning.

Explanation:

1. Specifies the data type provided by the plug-in module.
2. The SGMLTEXT LOB column in MEDICINE_MANAGEMENT_TABLE is divided among and stored in user LOB RDAREAs LOBAREA01 and LOBAREA02.
3. Specifies the plug-in portion. For details about specification, see the applicable plug-in documentation.
4. MEDICINE_MANAGEMENT_TABLE's LOB column structure base table is divided and stored in user RDAREAs RDAREA01 and RDAREA02.

(2) Defining a plug-in index

If you use the index type for data retrieval offered by a plug-in, you can retrieve data easily and at high speed. The index type offered by a plug-in is called *plug-in index*. This section explains how to define a plug-in index using the index type (NGRAM) provided by the HiRDB Text Search Plug-in.

The following example defines a plug-in index for MEDICINE_MANAGEMENT_TABLE using CREATE INDEX:

```
CREATE INDEX PLGINDX1
  USING TYPE NGRAM
  ON MEDICINE_MANAGEMENT_TABLE (OPERATION_MANUAL)
  IN ((LOBAREA03), (LOBAREA04));
```

Explanation:

For the row-partitioned MEDICINE_MANAGEMENT_TABLE, plug-in index PLGINDX1 is divided and stored in user LOB RDAREAs LOBAREA03 and LOBAREA04. OPERATION_MANUAL is specified for the column that constitutes the PLGINDX1 plug-in index.

(3) Storing data in the table

To use the database load utility (pdload) to store data in the table:

Procedure

1. Use the pdhold command to shut down the target data storage RDAREAs (RDAREA01–RDAREA02 and LOBAREA01–LOBAREA04).
2. Use the pdload command to load the input data file into the table. Because only the target data storage table and index are stored in the RDAREAs, and because this is an initial storage, select the no-log mode as the database update log acquisition mode. For the index creation method, select the batch index creation mode (the default value). For the constructor function and the data type information passed to the constructor function, specify a column structure information file. For details about the options of the pdload command, see the manual *HiRDB Version 9 Command Reference*.

3. Because the `pdload` command is executed in the no-log mode, make a backup of the target data storage RDAREAs. For details about how to make backups in units of RDAREAs, see the *HiRDB Version 9 System Operation Guide*.
4. Use the `pdrels` command to release the target data storage RDAREAs from shutdown status.

For details about these commands and utilities, and about how to verify the command and utility execution results, see the manual *HiRDB Version 9 Command Reference*.

Supplemental notes

- Because the `pdload` command executes in the no-log mode, the target data storage RDAREAs must remain on shutdown status during steps 1-3.
- In the case of a falsification prevented table, when data storage is performed with the `pdload` command, the `-d` option cannot be specified.
- For details about error handling during batch index creation, see 6.6 *Handling errors during batch index creation*.

(4) Checking the data storage status

If you have executed data loading, you should execute the database condition analysis utility (`pddbst`) next to check the data storage status. This utility enables you to check whether the database has been created exactly as designed. The database condition analysis utility (`pddbst`) can obtain the following information:

- Data storage status of each user RDAREA or user LOB RDAREA (physical analysis only)
- Data storage status of each registry RDAREA or registry LOB RDAREA (physical and logical analyses)

(5) Loading data in units of RDAREAs into a table for which partitioning conditions are specified with a hash function

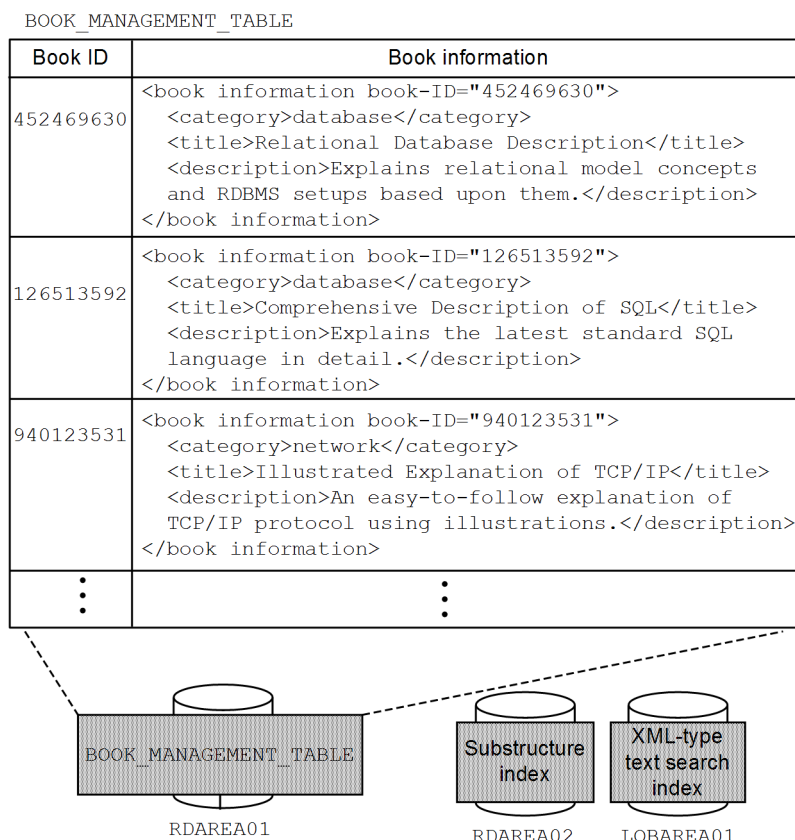
You can create a UAP using a hash function for table partitioning to create an input data file for each RDAREA. Because this makes it possible to check the amount of data to be stored in each RDAREA, you can select a hash function for uniform partitioning. For details about how to create a UAP for using a hash function for table partitioning, see the *HiRDB Version 9 UAP Development Guide*.

6.4.2 The XML type

This subsection describes the procedure for creating a table with the abstract data type (XML type) provided by HiRDB XML Extension.

The following procedure creates a `BOOK_MANAGEMENT_TABLE`. The following are the creation conditions assumed by the example:

- The `BOOK_MANAGEMENT_TABLE` is stored in user RDAREA `RDAREA01`.
- The substructure index is stored in user RDAREA `RDAREA02`.
- The XML-type full-text search index is stored in user LOB RDAREA `LOBAREA01`.
- In `RDAREA01`, `RDAREA02` and `LOBAREA01`, only for the target data load table (and index) is stored in the initial load.
- While data is loading, batch creation (the default value) is performed on the index.
- Data storage is performed in the no-log mode.



(1) Defining the BOOK_MANAGEMENT_TABLE

Define the `BOOK_MANAGEMENT_TABLE` with `CREATE TABLE`. An example follows:

Example of defining a table that includes an XML type column

```
CREATE TABLE BOOK_MANAGEMENT_TABLE (book-ID INTEGER, book-information
XML) IN RDAREA01
```

(2) Defining the index

(a) Substructure index (B-tree)

In XML type columns, a specific substructure can be made into a key, and an index can be defined with that value as its key value. Use this index to reduce the processing time for narrowing down the rows, if a predicate for a structure that defines a substructure index was specified in the XQuery expression of an `XML EXISTS` predicate or an `XMLQUERY` function.

The following lists the predicates within an XQuery expression that can use a substructure index:

- Comparison expressions (`=`, `!=`, `>`, `>=`, `<`, `<=`, `<>`, `eq`, `ne`, `lt`, `le`, `gt`, `ge`) using the substructure that is the key
- The `fn:contains` function, `fn:starts-with` function, or `fn:ends-with` function with the substructure that is the key

For details about index usage conditions, see (4) *Index usage conditions*.

For details about searches using substructure indexes, see the *HiRDB Version 9 UAP Development Guide*.

(b) XML-type full-text search index (n-gram)

N-gram indexes for full-text searches against the value of XML types (`IXXML`) can be defined within XML type columns. Defining a XML-type full-text search index can reduce line narrowing processing time when a predicate that

includes full-text search conditions such as character string matching is stated in the XQuery expression of a `XMLEXISTS` predicate.

Predicates in XQuery expressions that serve as conditions for XML-type full-text searches are:

- Perfectly matching character strings (`xs:string` type) (`=`)
- `fn:contains` function
- `fn:starts-with` function
- `fn:ends-with` function
- `hi-fn:contains` function

For details about index usage conditions, see (4) *Index usage conditions*.

For details about searches that use XML-type full-text search indexes, see the *HiRDB Version 9 UAP Development Guide*.

(3) Storing data in a table

There are two types of input data when data is stored in tables. The data storage methods to use differ according to the type of input data.

1. ESIS-B format, for converting XML language to XML insertion data

In this case, XML language is parsed using the XML conversion command (`phdxmlcnv`) or the XML conversion library (Java library), and XML insertion data (ESIS-B format) is generated. This ESIS-B format data is output in binary format and stored in a table using `pdload` or an `INSERT` statement.

For details about the XML conversion command and XML conversion library, see the manual *HiRDB Version 9 XML Extension*.

2. XML language

In this case, XML language is converted to XML insertion data (ESIS-B format) using the database load utility (`pdload`) or the `XMLPARSE` function, and stored in a table. The conversion from XML language to ESIS-B format data is conducted with `pdload` or the `XMLPARSE` function. To perform the conversion to ESIS-B format with `pdload`, specify the `-G` option.

(a) Loading data

To use the database load utility (`pdload`) to store data in the table:

Procedure

1. Use the `pdhold` command to shut down the target data load RDAREAs (`RDAREA01`, `RDAREA02` and `LOBAREA01`).
2. Use the `pdload` command to load the input data file into the table.
 - To use XML statements directly as the input data, specify the `-G` option.
 - Because only the target data load table and index are stored in the RDAREAs, and because this is an initial load, select the no-log mode as the database update log acquisition mode.
 - For the index creation method, select the batch index creation mode (the default value).
 - Specify the constructor function and the data type information passed to the constructor function in a column structure information file.
 - Set the format of the input data file to binary.

For details about the options of the `pdload` command, see the manual *HiRDB Version 9 Command Reference*.

3. Because the `pdload` command is executed in the no-log mode, make a backup of the target data load RDAREAs. For details about how to make backups in units of RDAREAs, see the *HiRDB Version 9 System Operation Guide*.
4. Use the `pdrels` command to release the target data load RDAREAs from shutdown status.

For details about these commands and utilities, and about how to verify the command and utility execution results, see the manual *HiRDB Version 9 Command Reference*.

Supplemental notes

- Because the `pdload` command executes in the no-log mode, the RDAREAs to which data is being loaded must remain in shutdown status during steps 1 to 3.
- In the case of a falsification prevented table, when data loading is performed with the `pdload` command, the `-d` option cannot be specified.
- For details about error handling during batch index creation, see *6.6 Handling errors during batch index creation*.

(b) Insertion of XML language**Inserting ESIS-B format data into a table or updating ESIS-B format data in a table**

Specify an XML constructor function as the insertion value of an `INSERT` statement or the update value of an `UPDATE` statement, and set the generated ESIS-B format data in its argument.

The following is an example of inserting an XML language (ESIS-B format) value stored in embedded variable `bookinfo` into `BOOK_MANAGEMENT_TABLE`.

Example of XML language (ESIS-B format) insertion

```
INSERT INTO BOOK MANAGEMENT TABLE
VALUES ( 310494321, XML(:bookinfo AS BINARY(102400)) )
```

Inserting XML language into a table or updating XML language in a table

Specify an `XMLPARSE` function as the insertion value of an `INSERT` statement or as the update value of an `UPDATE` statement, and set the XML language in its argument.

The following is an example of inserting an XML language value stored in embedded variable `bookdoc` into `BOOK_MANAGEMENT_TABLE`.

Example of XML language insertion

```
INSERT INTO BOOK MANAGEMENT TABLE
VALUES ( 310494321, XMLPARSE(DOCUMENT :bookdoc AS BINARY(32000)) )
```

(4) Index usage conditions

This section describes the two index usage conditions described in (2).

Substructure index usage conditions

When a substructure index is defined, the index is used when the substructure index usage conditions shown in the following table are met.

Table 6–4: Substructure index usage conditions

USING UNIQUE TAG specified?	Where XQuery specification is located	Operators or functions in XQuery	Substructure index usage conditions [#]
Yes	XMLEXISTS predicate	=	(a) 1, 2, 3 (b) 1, 2, 5, 6, 7, 8, 9
		!=, >, >=, <, <=, <>, eq, ne, gt, ge, lt, le, fn:contains, fn:starts-with, fn:ends-with	(a) 1, 2, 3 (b) 1, 2, 4, 6, 7, 8, 9
	XMLQUERY function	=	(c) 1, 2, 3
No	XMLEXISTS predicate	=	(a) 1, 2, 3 (b) 1, 3, 5, 6, 7, 8, 9
		!=, >, >=, <, <=, <>	(a) 1, 2, 3 (b) 1, 3, 4, 6, 7, 8, 9
		fn:contains	(a) 1, 2, 3

USING UNIQUE TAG specified?	Where XQuery specification is located	Operators or functions in XQuery	Substructure index usage conditions [#]
		fn:starts-with fn:ends-with	(b) 1, 3, 4, 6, 7, 8, 10
	XMLQUERY function	=	(c) 1, 2, 3

#

(a): Indicates the usage conditions common to substructure indexes and XML-type full-text search indexes, as described in (a) below.

(b): Indicates the usage conditions of substructure indexes as described in (b) below.

(c): Indicates the usage conditions of substructure indexes pertaining to XQuery in the XMLQUERY function, as described in (c) below.

The numbers refer to the item numbers in sections (a), (b), and (c) below.

XML-type full-text search index usage conditions

When an XML-type full-text search index is defined, the index is used when the usage conditions for XML -type full-text search indexes shown in the following table are met.

Table 6–5: XML-type full-text search index usage conditions

Where XQuery specification is located	Operators or functions in XQuery	Index usage conditions for XML-type full-text search [#]
XMLEXISTS predicate	fn:contains fn:starts-with fn:ends-with =	(a) 1, 2, 3 (d) 1, 2, 4, 5, 6, 7, 8
	hi-fn:contains	(a) 1, 2, 3 (d) 1, 3, 4, 5, 6, 7, 8

#

(a): Indicates the usage conditions common to substructure indexes and XML-type full-text search indexes, as described in (a) below.

(d): Indicates the usage conditions for XML -type full-text search indexes, as described in (d) below.

The numbers refer to the item numbers in sections (a) and (d) below.

Operators and functions that can use multiple indexes

During a search with XQuery operators or functions that can use both substructure indexes and XML-type full-text search indexes, the index used for evaluation is determined by the operator or function. The table below shows which indexes are used with each operator or function.

Table 6–6: Indexes used to evaluate operators and functions that can use multiple indexes

Item No.	Operator or function	Index used in evaluation
1	=	Substructure index
2	fn:contains	XML-type full-text search index
3	fn:starts-with	Substructure index
4	fn:ends-with	XML-type full-text search index

To specify the index to be used, specify SQL optimization for that index. For details, see *SQL optimization specification for a used index* in the manual *HiRDB Version 9 SQL Reference*.

HiRDB might sometimes not use these indexes because of the estimated access cost. Use the access path display utility (pdvwopt) to determine whether a search that uses an index will be conducted.

For predicates that can use substructure indexes or XML-type full-text search indexes that were specified within the XQuery query of a XMLEXISTS predicate, only a maximum of 255 indexes can be used in evaluation.

(a) Usage conditions common to substructure indexes and XML-type full-text search indexes

The following are usage conditions that are common to substructure indexes and XML-type full-text search indexes. Indexes defined in the manner shown below are used in sample text.

```
create index idx1 on t1(c1) key using unique tag from '/root/elm1/@attr1' as
varchar(10)
```

1. Any XML query context item is specified in the XML query argument of a XMLEXISTS predicate.

Example

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq "ABC"]'
    passing by value c1, 'DEF' as A)
```

Note: The underlined part is the XML query context item.

2. All context item expressions (periods) specified in the XQuery query of the XMLEXISTS predicate are specified in the XQuery predicate.

Example

```
select c2 from t1
  where xmlexists('/root/elm1[_/@attr1 eq "ABC"]' passing by value c1)
```

Note: The underlined part is the context item expression specified in the XQuery predicate.

3. All XQuery Boolean expressions (AND, OR) within the XQuery query of the XMLEXISTS predicate are specified in the XQuery predicate.

Example

```
select c2 from t1
  where xmlexists('/root[elm1/@attr1 = "ABC" or elm1/@attr1 = "DEF"]'
    passing by value c1)
```

Note: The underlined part is the XQuery Boolean expression (OR) specified in the XQuery predicate.

(b) Usage conditions for substructure indexes pertaining to the XQuery of the XMLEXISTS predicate

The following are usage conditions for substructure indexes pertaining to the XQuery of the XMLEXISTS predicate.

Items 1, 2, and 4 to 9 use indexes defined as shown below in the sample text.

```
create index idx1 on t1(c1) key using unique tag from '/root/elm1/@attr1' as
varchar(10)
create index idx4 on t1(c1) key using unique tag from '/root/elm1/elm2' as
varchar(10)
```

Items 3 and 10 use indexes defined as shown in the following sample text.

```
create index idx2 on t1(c1) key from '/root/elm1/@attr1' as varchar(10)
create index idx5 on t1(c1) key from '/root/elm1/elm2' as varchar(10)
```

1. The index substructure specification matches the substructure path specified as a condition in the XQuery query of the XMLEXISTS predicate.

Example

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq "ABC"]' passing by value c1)
```

Note: The underlined part is the matching substructure path.

2. If the substructure index has a USING UNIQUE TAG specification, it is compared to the substructure path specified as a condition in the XQuery query of the XMLEXISTS predicate. This is done using a general comparison, value comparison, fn:contains function, fn:starts-with function, or fn:ends-with function.

Example

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq "ABC"]' passing by value c1)
```

Note: The underlined part is an XQuery comparison expression (value comparison).

3. If the substructure index does not have a USING UNIQUE TAG specification, it is compared to the substructure path specified as a condition in the XQuery query of the XMLEXISTS predicate. This is done using a general comparison, `fn:contains` function, `fn:starts-with` function, or `fn:ends-with` function.

Example

```
select c2 from t1
  where xmlexists('/root/elml[@attr1 = "ABC"]' passing by value c1)
```

Note: The underlined part is an XQuery comparison expression (general comparison).

4. The following describes the case with general or value comparisons separately from the case with `fn:contains`, `fn:starts-with`, or `fn:ends-with` functions.

Using general or value comparisons

The items compared using a general or value comparison are the substructure path specified as a condition in the XQuery query of the XMLEXISTS predicate, and a single XQuery constant or XQuery variable.

Example

```
select c2 from t1
  where xmlexists('/root/elml[@attr1 >= "ABC"]' passing by value c1)
```

Note: The underlined part compares the substructure path to an XQuery constant.

Using the `fn:contains`, `fn:starts-with`, or `fn:ends-with` functions

The first argument of the `fn:contains`, `fn:starts-with`, or `fn:ends-with` function is the substructure path specified as a condition in the XQuery query of the XMLEXISTS predicate, while the second argument is a single XQuery constant or XQuery variable.

Example

```
select c2 from t1
  where xmlexists('/root/elml[fn:starts-with(@attr1, "ABC")]' passing by
value c1)
```

Note: The underlined part compares a substructure path and an XQuery constant.

5. Comparisons with `=` compare the substructure path, specified as a condition in the XQuery query of the XMLEXISTS predicate, to an XQuery sequence concatenation expression comprised of an XQuery constant or XQuery variable at or below the specification of the system common definition `pd_apply_search_ats_num` operand.

Example

```
select c2 from t1
  where xmlexists('/root/elml[@attr1 = ("ABC", "DEF", "GHI")] '
passing by value c1)
```

Note: The underlined part is an XQuery sequence concatenation expression comprised of an XQuery constant at or below the specification of the system common definition `pd_apply_search_ats_num` operand.

6. The data type of the key value specified when the substructure index was defined is the same as, or can be converted to, the data type of the XQuery constant or the value expression passed to the XQuery variable in the XQuery query that is compared to the substructure path specified as a condition in the XQuery query of the XMLEXISTS predicate.

Example

```
select c2 from t1
  where xmlexists('/root/elml[@attr1 = "ABC"]' passing by value c1)
```

Note: The underlined part is string type data that is the same as the VARCHAR type that is the data type of the key value.

7. The XQuery Boolean expression (OR) operand specified in the XQuery query of the XMLEXISTS predicate contains only the condition that allows use of substructure indexes.

Example

```
select c2 from t1
  where xmlexists('/root[elml/@attr1 = "ABC" or elml/@attr1 = "DEF"] '
passing by value c1)
```

Note: The underlined part is the condition that allows use of all substructure indexes.

8. If an XQuery variable was specified as a condition in items 4 or 5 as the value compared with the substructure path, the value expression passed to that XQuery variable is one of the following:

- Constant

- USER value function
- CAST specification whose value expression is a dynamic parameter, an SQL parameter, or an SQL variable
- Scalar subquery that does not make an external reference

Example 1

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq $A]'
    passing by value c1, 'ABC' as A)
```

Note: The underlined part is the value expression passed to an XQuery variable in an XQuery query (constant).

Example 2

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq $A]'
    passing by value c1, cast(? as varchar(256)) as A)
```

Note: The underlined portion is the value expression passed to the XQuery variable in an XQuery query (CAST specification whose value expression is a dynamic parameter, an SQL parameter, or an SQL variable).

9. The substructure path that is compared using a value or general comparison is specified in the following format. Alternatively, the substructure index has a USING UNIQUE TAG specification, and the substructure path that includes the first argument of the fn:contains, fn:starts-with, or fn:ends-with function is specified in the following format.

```
substructure-path:: = [XML-namespace-declaration]... substructure-path-
expression
XML-namespace-declaration::={declare namespace prefix = XML-namespace-URI;
    | declare default element namespace XML-namespace-URI;}
substructure-path-expression:: = [/step-expression...]/step-expression
step-expression:: = {[child:: | attribute:: | @ ]} qualifier-name |
context-item-expression}
context-item-expression:: = period
period::= .
qualifier-name:: = [prefix:]local-name
```

Example using value or general comparison

The following three examples use value comparisons or general comparisons. In these examples, the index is used even when another value comparison or general comparison is specified in the value comparison operator eq. See items 2 and 3 for index usage conditions pertaining to whether USING UNIQUE TAG has been specified.

Example 1: @ or attribute:: and a qualifier name are specified in a step expression (@ and attribute:: have the same meaning).

```
select c2 from t1
  where xmlexists('/root/child::elm1[@attr1 eq "ABC"]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 2: child:: is specified in a step expression, or child:: is not specified and a qualifier name is specified.

```
select c2 from t1
  where xmlexists('/root/child::elm1[child::elm2 eq "ABC"]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 3: A context item expression is specified in a step expression (only when usage condition 2 applies to the value comparison).

```
select c2 from t1
  where xmlexists('/root/child::elm1/elm2[. eq "ABC"]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Examples using XQuery functions

The following examples, 4 through 6, use XQuery functions. In these examples, the index is used even when an XQuery function other than fn:starts-with is specified.

Example 4: @ or attribute:: and a qualifier name are specified in a step expression (@ and attribute:: have the same meaning).

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(@attr1 , "ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 5: `child::` is specified in a step expression, or `child::` is not specified and a qualifier name is specified.

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(elm2 , "ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 6: A context item expression is specified in a step expression.

```
select c2 from t1
  where xmlexists('/root/elm1/@attr1[fn:starts-with( . , "ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

10. With a substructure index that has no USING UNIQUE TAG specification, the substructure path that contains the first argument of the `fn:contains`, `fn:starts-with`, or `fn:ends-with` function is specified in the following format. The first argument is also specified in the following step expression end format.

```
substructure-path:: = [XML-namespace-declaration]... substructure-path-expression
XML-namespace-declaration::={declare namespace prefix = XML-namespace-URI;
  | declare default element namespace XML-namespace-URI;}
substructure-path-expression:: = [/step-expression...]/step-expression-end
step-expression:: = {[child:: | attribute:: | @ ]} qualifier-name |
context-item-expression
step-expression-end:: = {[attribute:: | @ ]} qualifier-name | context-item-expression
context-item-expression:: = period
period::= .
qualifier-name:: = [prefix:]local-name
```

In the following examples, the index is used even when another XQuery function is specified in the `fn:starts-with` function.

Example 1: `@` or `attribute::` and a qualifier name are specified in a step expression end (`@` and `attribute::` have the same meaning).

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(@attr1 , "ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 2: A context item expression is specified in a step expression end.

```
select c2 from t1
  where xmlexists('/root/elm1/@attr1[fn:starts-with( . , "ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

(c) Usage conditions for substructure indexes pertaining to the XQuery of the XMLQUERY function

This section shows usage conditions for substructure indexes pertaining to the XQuery of the XMLQUERY function.

An index defined as follows is used in the examples of items 1 and 2.

```
create index idx1 on t1(c1) key using unique tag from '/root/elm1' as
varchar(10)
```

1. The SQL code fulfills all the following conditions.

- It is a SELECT statement or code from an INSERT statement to a SELECT statement.
- There is a single main query SELECT expression.
- That main query SELECT expression is XMLQUERY (the XMLSERIALIZE argument can be XMLQUERY).
- That XMLQUERY XML query argument is a single XML query variable, and the value expression passed to the variable is XMLAGG.

- The XMLAGG argument specified in that XML query variable is an independent column specification.
- The code specifies no table join.
- The code specifies no set operation.
- The code specifies no subqueries.
- The code specifies no set function.
- The code specifies no GROUP BY clause.
- The code specifies no HAVING clause.
- The code specifies a WHERE clause, but specifies AND 255 times or fewer.

See example 2 for SQL examples.

- The XQuery specified in the XMLQUERY function satisfies all the following conditions.
 - It is a path expression that uses an XML query variable as a route.
 - There is only one XQuery predicate specification at the outermost level.
 - A comparison is performed between the predicate of b. and =, using a general comparison.
 - The comparison of c. is a path expression comparison whose route is an XML column specific substructure and an XQuery variable.

Example

```
select xmlserialize(xmlquery('$VAR1/root[elm1 = $VAR1/root[elm2 = "ABC"]/
elm1]/elm1'
  passing by value xmlagg(c1) as VAR1 empty on empty) as varchar(32000))
from t1
```

Note: The underlined part is the location that matches the above conditions.

- A substructure index is defined with the same data type as the path expression in 2d. whose route is an XML column specific substructure and an XQuery variable (these indexes can be identical).

(d) Usage conditions for XML-type full-text search indexes

The following are usage conditions for XML -type full-text search indexes.

- A full-text search index is defined in the XML type column that is the search target.

Example

```
create index idx3 using type ixxml on t2(c1) in (LOB1)
```

- The substructure path that contains the first argument of the fn:contains, fn:starts-with, or fn:ends-with function is specified in the following format. The first argument is specified in the following text step expression end or attribute step expression end format. Alternatively, the substructure path that is the subject of an = comparison is specified in the following format, and the substructure path specified in the XQuery predicate is specified in the following text step expression end or attribute step expression end format.

```
substructure-path:: = [XML-namespace-declaration]... substructure-path-
expression
XML-namespace-declaration:: = {declare default element namespace
  "http://www.w3.org/XML/1998/namespace";
  |declare namespace prefix = XML-namespace-URI;#
  |declare default element namespace XML-namespace-URI;#}
substructure-path-expression:: = [{/ | //}#}step-expression ...]
  {/ | //}#}{text-step-expression | attribute-step-expression-end}
step-expression:: = {[child::] name-test | context-item-expression}
text-step-expression:: = [{child::|descendant::}]text-test
  /text-step-expression-end
text-step-expression-end:: = context-item-expression
attribute-step-expression-end:: = {[attribute:: | @] name-test
  | [{attribute:: | @}] attribute-test}
context-item-expression:: = period
period:: = .
name-test:: = {qualifier-name|*#|prefix:*#|*:local-name#}
qualifier-name:: = [prefix:]local-name
```

#: Can be specified if the HiRDB XML Extension version is 08-04 or later.

The index is used even if another XQuery function is specified in the fn:contains function in the following example.

Example 1: @ or attribute:: and a name test are specified in the first argument (@ and attribute:: have the same meaning).

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(@attr1,"ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 2: @ or attribute:: and an attribute test are specified in the first argument (@ and attribute:: have the same meaning).

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(@attribute(),"ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 3: Only an attribute test is specified in the first argument.

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(attribute(),"ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 4: A context item expression is specified in the first argument.

```
select c2 from t2
  where xmlexists('/root/child::elm1/text()[fn:contains(., "ABC")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

3. The substructure path that contains the first argument of the `hi-fn:contains` function is specified in the following format. The first argument is also specified in the text step expression, text step expression end, or attribute step expression end format shown below. Also, the version of HiRDB XML Extension is 08-04 or later.

```
substructure-path:: = [XML-namespace-declaration]... substructure-path-expression
XML-namespace-declaration:: = {declare namespace prefix = XML-namespace-URI;
  | declare default element namespace XML-namespace-URI;}
substructure-path-expression:: = [{/ | //}step-expression ...]
  {/ | //}{text-step-expression | attribute-step-expression-end}
step-expression:: = {[child::] name-test | context-item-expression}
text-step-expression:: = {[child::]descendant::}text-test
  [/text-step-expression-end]
text-step-expression-end:: = context-item-expression
attribute-step-expression-end:: = [{attribute:: | @} name-test
  | [{attribute:: | @}] attribute-test}
context-item-expression:: = period
period:: = .
name-test:: = {qualifier-name | * | prefix:* | *:local-name}
qualifier-name:: = [prefix:]local-name
```

Example 1: A text test is specified in the first argument.

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(text(),""ABC AND DEF"")]'
    passing by value c1)

select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(descendant::text(),""ABC AND
  DEF"")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 2: A text test and a context item expression are specified in the first argument.

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(text()/., ""ABC AND DEF"")]'
    passing by value c1)

select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(descendant::text()/., ""ABC
  AND DEF"")]'
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 3: A context item expression is specified in the first argument.

```
select c2 from t2
  where xmlexists('/root/elm1/text() [hi-fn:contains(., ""ABC AND DEF"")]')
    passing by value c1)
```

```
select c2 from t2
  where xmlexists('/root/elm1/ descendant::text() [hi-fn:contains(., ""ABC AND DEF"")]')
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 4: @ or attribute:: and a name test are specified in the first argument (@ and attribute:: have the same meaning).

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(@attr1, ""ABC AND DEF"")]')
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 5: @ or attribute:: and an attribute test are specified in the first argument (@ and attribute:: have the same meaning).

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(@attribute(), ""ABC AND DEF"")]')
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

Example 6: Only an attribute test is specified in the first argument.

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(attribute(), ""ABC AND DEF"")]')
    passing by value c1)
```

Note: The underlined part is the substructure path that matches the above format.

4. The length of the character string specified in the XQuery query of the XMLEXISTS predicate is 32,000 bytes or less.

Example

```
select c2 from t2
  where xmlexists('/root/elm1[fn:contains(@attr1, "ABCDEF")]'
    passing by value c1)
```

Note: The underlined part is a character string of 32,000 bytes or less.

5. The value compared to the substructure path specified in the format of 2 or 3 is a character string XQuery constant.

Example

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(@attr1, "ABC")]'
    passing by value c1)
```

Note: The underlined part is the character string XQuery constant.

6. The sum of the length of the /, //, @, and local name specified in the substructure path expression of the XQuery query of the XMLEXISTS predicate (except when the first character of the substructure path expression is /) is 1,024 bytes or less.

Example

```
select c2 from t2
  where xmlexists('/root/elm1[fn:contains(@attr1, "ABCDEF")]'
    passing by value c1)
```

Note: The underlined part is a substructure path expression of 1,024 bytes or less.

7. There is no more than one // specified in the substructure path expression specified in the XQuery query of the XMLEXISTS predicate.

Example

```
select c2 from t2
  where xmlexists('/root/elm1[fn:contains(@attr1, "ABCDEF")]'
    passing by value c1)
```

Note: The underlined part is the substructure path expression with no more than one // specification.

8. None of the following plug-in options has been specified in the full-text search index defined in the XML type column that is the search target.

- DELcode=file-name
- NOindex=file-name
- ENGLISH
- ENGLISH_STANDARD

For details about plug-in options, see the manual *HiRDB Version 9 XML Extension*.

Example

```
create index idx6 using type ixxml on t2(c1) in (LOB1)
PLUGIN 'SAMECASE=ON, SAMEWIDE=ON, SAMEY=ON, SAMED=ON, DELcode=ON'
```

Note: The underlined part is the full-text search index that specified only options that can use the index.

(e) If no index is used

In the following situations, no substructure index or XML-type full-text search index is used.

In items 1 and 3 below, an index defined as follows is used in the examples.

```
create index idx1 on t1(c1) key using unique tag from '/root/elml/@attr1' as
varchar(10)
```

In items 2 and 4 below, an index defined as follows is used in the examples.

```
create index idx3 using type ixxml on t2(c1) in (LOB1)
```

1. If an XQuery that evaluates whether '/root[elml/@attr1' is "ABC" or "DEF" is specified in the XQuery query of the XMLEXISTS predicate

In the following examples, not all of the XQuery Boolean expressions (AND, OR) in the XQuery query of the XMLEXISTS predicate are specified in the XQuery predicate. Consequently, no index is used. With this specification method, XQuery Boolean expressions are not specified directly in the XQuery query of the XMLEXISTS predicate argument (except within the XQuery predicate), so the XQuery query result will always be a Boolean value of TRUE or FALSE. For this reason, the XQuery query result is not a NULL sequence, so the XMLEXISTS predicate result is always TRUE and will not be the intended result (the XMLEXISTS predicate is only FALSE when the XQuery query result is a NULL sequence; otherwise, it is TRUE).

Example: Before change

```
select c2 from t1
  where xmlexists('/root[elml/@attr1 = "ABC"] or /root[elml/@attr1 = "DEF"]'
    passing by value c1)
```

Note: The underlined part is the XQuery Boolean expression (OR) not specified in the XQuery predicate.

If the index is changed as follows, it will be used.

Example: After change

```
select c2 from t1
  where xmlexists('/root[elml/@attr1 = "ABC" or elml/@attr1 = "DEF"]'
    passing by value c1)
```

Note: The underlined part is the XQuery Boolean expression (OR) specified in the XQuery predicate.

2. When an XQuery that searches text nodes under '/root/elml' is specified in an XQuery query of an XMLEXISTS predicate

In the following example, the text step expression end of the substructure path that includes the first argument of the contains function does not match the format shown in item 2 of (4)(c), so the index is not used.

Example: Before change

```
select c2 from t2
  where xmlexists('/root[fn:contains(elml/text(), "ABC")]') passing by value
c1)
```

Note: The underlined part is a substructure path that does not match the format shown in item 2 of (4)(c).

If the index is changed as follows, it will be used.

Example: After change


```
select c2 from t2
  where xmlexists('/root/elm1/text()[fn:contains( _, "ABC")]'
    passing by value c1)
```

Note: The underlined part is a substructure path that matches the format shown in item 2 of (4)(c).

3. If an XQuery that evaluates whether '/root/elm1/@attr1/' is "ABC" is specified in the XQuery query of the XMLEXISTS predicate

In the following example, the substructure path expression of the substructure path specified as a condition in the XQuery query of the XMLEXISTS predicate does not match the format shown in item 9 of (4)(b). Consequently, no index is used.

Example: Before change

```
select c2 from t1
  where xmlexists('/$A/root/elm1[@attr1 eq "ABC"]'
    passing by value c1, c1 as A)
```

Note: The underlined part is a substructure path that does not match the format shown in item 9 of (4)(b).

If the index is changed as follows, it will be used.

Example: After change

```
select c2 from t1
  where xmlexists('/_/root/elm1[@attr1 eq "ABC"]'
    passing by value c1, c1 as A)
```

Note: The underlined part is a substructure path that matches the format shown in item 9 of (4)(b).

4. If many XMLEXISTS predicates are joined by Boolean operators (OR, AND)

Due to calculated access costs, HiRDB judges that not using the index is the optimal access path, and the index is not used to evaluate XMLEXISTS predicates. If the hi-fn:contains function was specified, an evaluation by the index alone is impossible, so an SQL error results.

Example: Before change

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(text(), ""01ABC"")]'
    passing by value c1)
    or xmlexists('/root/elm1[hi-fn:contains(text(), ""02ABC"")]'
    passing by value c1)
    ... (Omitted) ...
    or xmlexists('/root/elm1[hi-fn:contains(text(), ""30ABC"")]'
    passing by value c1)
```

Note: The underlined part is the condition that specifies 30 XMLEXISTS predicates that use the index if they are specified alone.

When the used index specifies SQL optimization as shown below, the index is used.

Example: After change

```
select c2 from t2 with index(idx3,idx3)
  where xmlexists('/root/elm1[hi-fn:contains(text(), ""01ABC"")]'
    passing by value c1)
    or xmlexists('/root/elm1[hi-fn:contains(text(), ""02ABC"")]'
    passing by value c1)
    ... (Omitted) ...
    or xmlexists('/root/elm1[hi-fn:contains(text(), ""30ABC"")]'
    passing by value c1)
```

Note: The underlined part is an index SQL optimization specification that specifies an index of the type required for execution using multiple indexes.

6.5 Creating a table containing a user-defined abstract data type

6.5.1 Defining an abstract data type

The user can use an abstract data type and routines to define and use any desired data type with a complicated structure and a desired data manipulation method.

(1) Definition method

The CREATE TYPE definition SQL is used to define a data type with a desired structure (an abstract data type). CREATE TYPE defines a data structure and a data manipulation method. This section explains how to define the abstract data type t_EMPLOYEE with the data structure shown below and then define a data manipulation method as a function:

Data structure

The data consists of NAME, SEX, POSITION, EMPLOYMENT_DATE, ID_PHOTO, and SALARY.

Data manipulation

- Calculate SERVICE_YEARS from the current date and EMPLOYMENT_DATE.
- Calculate BONUS_FACTOR according to SERVICE_YEARS.
- Calculate the employee's bonus by multiplying SALARY times BONUS_FACTOR.

Example

```

CREATE TYPE t_EMPLOYEE (
    PUBLIC      NAME      NCHAR(16),
                SEX       CHAR(1),
                POSITION   NCHAR(10),
    PRIVATE    EMPLOYMENT_DATE date,
    PUBLIC     ID_PHOTO   BLOB(64K),
    PROTECTED  SALARY     INTEGER,

    PUBLIC FUNCTION t_EMPLOYEE (p_NAME NCHAR(16),
                                p_SEX CHAR(1),
                                p_POSITION NCHAR(10),
                                p_EMPLOYMENT_DATE date,
                                p_ID_PHOTO BLOB(64K),
                                p_SALARY INTEGER)
        RETURNS t_EMPLOYEE
    BEGIN
        DECLARE d_EMPLOYEE t_EMPLOYEE;
        SET d_EMPLOYEE=t_EMPLOYEE ();
        SET d_EMPLOYEE..NAME=p_NAME;
        SET d_EMPLOYEE..SEX=p_SEX;
        SET d_EMPLOYEE..POSITION=p_POSITION;
        SET d_EMPLOYEE..EMPLOYMENT_DATE
            =p_EMPLOYMENT_DATE;
        SET d_EMPLOYEE..ID_PHOTO =p_ID_PHOTO;
        SET d_EMPLOYEE..SALARY=p_SALARY;
        RETURN d_EMPLOYEE;
    END,

    PUBLIC FUNCTION SERVICE_YEARS (p t_EMPLOYEE)
    RETURNS INTEGER
    BEGIN
        DECLARE working_years INTERVAL YEAR TO DAY;
        SET working_years=CURRENT_DATE - p.. EMPLOYMENT_DATE;
        RETURN YEAR(working_years);
    END,

    PROTECTED FUNCTION BONUS_FACTOR (p t_EMPLOYEE)
    RETURNS FLOAT
    BEGIN
        DECLARE rate FLOAT;

```

```

        SET rate=SERVICE_YEARS (p)*0.2/30;
        RETURN rate;
    END,

    PUBLIC FUNCTION BONUS (p t_EMPLOYEE)
    RETURNS INTEGER
    BEGIN
        DECLARE bonus INTEGER;
        SET bonus=p..SALARY*BONUS_FACTOR (p);
        RETURN bonus;
    END
)

```

Explanation:

1. Defines the data structure. This example defines abstract data type `t_employee`.
2. Attribute `EMPLOYMENT_DATE` of the `t_EMPLOYEE` type is used to access the bonus. Encapsulation level `PRIVATE` is specified for this attribute, because there is no need to reference or modify it directly from the outside. For details about the encapsulation level, see *12.17 Table containing an abstract data type*.
3. Attribute `SALARY` of the `t_EMPLOYEE` type is used to calculate the bonus. This attribute also need not be referenced or modified directly from the outside. However, encapsulation level `PROTECTED` is specified for this attribute because its subtype is commonly referenced. For details about the encapsulation level, see *12.17 Table containing an abstract data type*.
4. Defines a user-defined constructor function.
5. Generates a value (instance) and declares an SQL variable to be used as the function's return value with the `t_EMPLOYEE` type.
6. Uses the system-provided default constructor function to generate a value (instance) whose attributes are all `NULL`. The default constructor function has the same name as the `t_EMPLOYEE` type with no argument.
7. For the value specified in item 6 above, assign the value of each attribute using an assignment statement specifying the component. The assignment statement can be used to set the value obtained from the constructor function's argument or to set the data processed using that value.
8. The `RETURN` statement returns a newly generated value (instance). The data type of the return value must be `t_EMPLOYEE`, because the constructor function has the same name as the abstract data type and the type is determined by the `RETURNS` clause.
9. This is a data manipulation function. It returns the employee's `SERVICE_YEARS`. This value is calculated from the current date and `EMPLOYMENT_DATE`. This function accesses the `EMPLOYMENT_DATE` attribute for which `PRIVATE` is specified as the encapsulation level.
10. This is another data manipulation function. It returns the employee's `BONUS_FACTOR`. `SERVICE_YEARS` is used to calculate this value.
11. This is another data manipulation function. It returns the employee's `BONUS`. This value depends on `SERVICE_YEARS` and is obtained by multiplying `SALARY` by `BONUS_FACTOR`.

(2) Definition method using inheritance

Following is an example of defining the subtype `t_OPERATOR` with the supertype being the `t_EMPLOYEE` abstract data type:

Example

```

CREATE TYPE
CREATE TYPE t_OPERATOR UNDER t_EMPLOYEE
( PUBLIC CHARGE_CLIENT NCHAR(15),
  PUBLIC FUNCTION BONUS (p t_OPERATOR) RETURNS INTEGER
  BEGIN
      DECLARE salebonus INTEGER;
      SET salebonus=TOTAL_CLIENTS (...) *1000+P..SALARY*BONUS (p);
      RETURN salebonus;
  END
)

```

(3) Null value for the abstract data type

If values are specified with the `INSERT` data manipulation SQL, the values for the entire abstract data type are set to null.

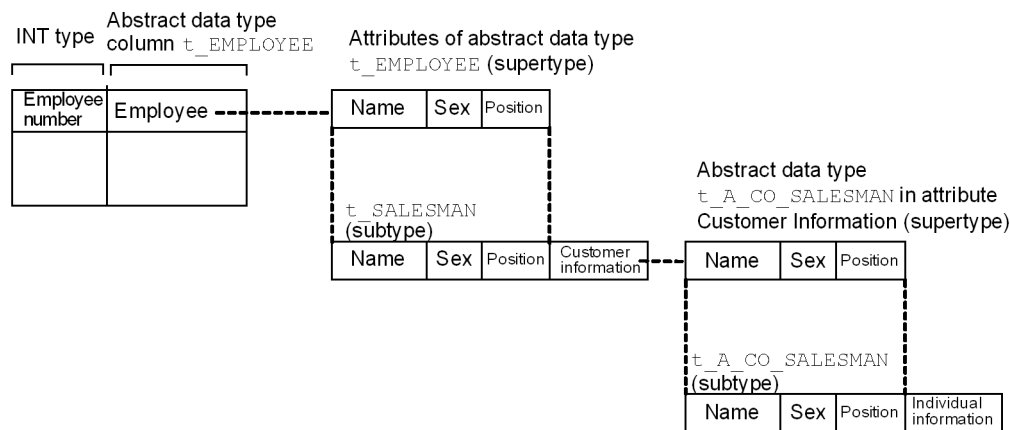
(4) Procedure for deleting the subtype of an abstract data type

If an abstract data type is not specified directly in the table definition, but its parent abstract data type (supertype) is specified as a column type, then the value of the abstract data type (subtype) may have been stored in the table due to substitutability. Care must be taken when an abstract data type (subtype) is deleted.

The procedure for deleting a subtype is described as follows, based on a table containing an abstract data type using substitutability, as shown in the following figure.

Figure 6–7: Example of table containing abstract data type using substitutability

- Staff table



1. Delete `STAFF_TABLE`.
2. Delete subtype `t_OPERATOR` of `t_EMPLOYEE`.
3. Delete `t_EMPLOYEE`.
4. Delete subtype `t_A_COMPANY_STAFF` of `t_A_COMPANY_STAFF`.
5. Delete `t_A_COMPANY_STAFF`.

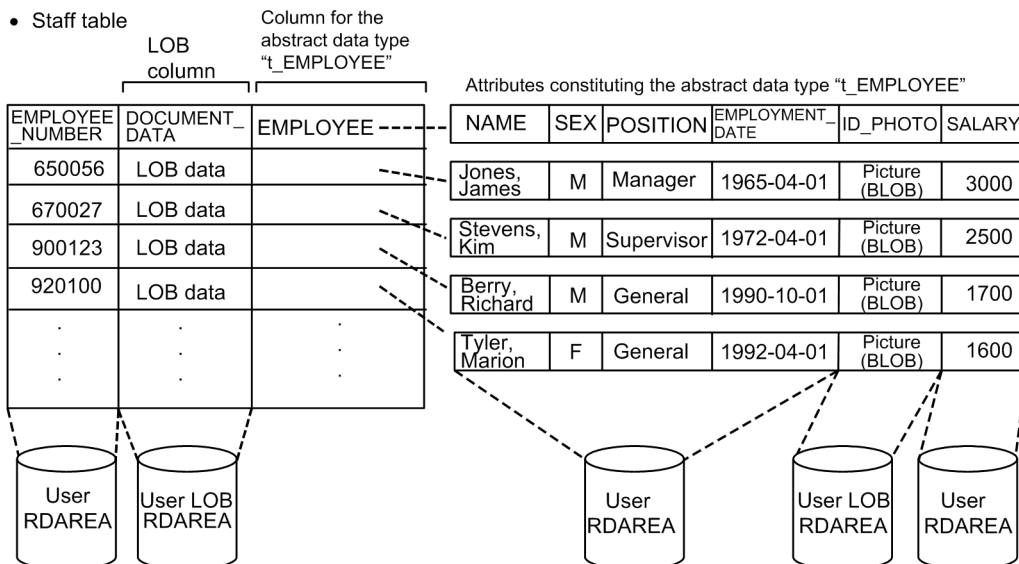
See (5) as follows for the subtypes of the abstract data type that cannot be deleted.

(5) Notes

1. If a constructor function is used to generate values, the abstract data type as a whole is not null, even if the value of each attribute constituting the abstract data type is null.
2. If an abstract data type and its supertype are defined in a table, the abstract data type's subtypes cannot be deleted.
3. If an abstract data type and its supertype are specified as attributes of another abstract data type, the abstract data type's subtypes cannot be deleted.
4. When a subtype is defined and the parent of the data type being created is one of the following, the corresponding stored procedure and stored function become invalid:
 - Data type specified in the SQL parameter of the stored procedure and stored function.
 - Data type of the function's return value.
 - Data type of the argument and return value of the function that is invoked from the stored procedure and stored function.
 - Data type specified in the stored procedure and stored function (including any intermediate data type if the abstract data type is accessed with a component specified).

6.5.2 Defining a table

The RDAREA storage unit depends on the data type of the columns that constitute the table. The explanations below are based on the example of a `STAFF_TABLE` table that consists of `EMPLOYEE_NUMBER`, `DOCUMENT_DATA` (LOB data), and abstract data type `t_EMPLOYEE`. For a table containing abstract data type columns, the portion without the abstract data type columns is called the abstract data type column structure base table.



Note

- Only 1 LOB column from a table is stored in a single user LOB RDAREA. If a table has multiple LOB columns, each must be stored in a separate RDAREA. Non-LOB columns do not need to be distributed into multiple user RDAREAs.
- For each LOB column in a row partitioned table, there must be a 1:1 relationship between user LOB RDAREAs and user RDAREAs where the table is stored.

Explanation:

`STAFF_TABLE` is divided among and stored in user RDAREAs `RDAREA01` and `RDAREA02` on disks A and B, respectively. `DOCUMENT_DATA` of this `STAFF_TABLE` (LOB column) is stored in user LOB RDAREAs `LOBAREA01` and `LOBAREA02`, and the abstract data type (LOB attribute) `ID_PHOTO` is stored in user LOB RDAREAs `LOBAREA03` and `LOBAREA04`.

(1) Key range partitioning

Specification of storage conditions:

```
CREATE TABLE STAFF_TABLE
(EMPLOYEE_NUMBER CHAR(6),
 DOCUMENT_DATA BLOB(64K) IN ((LOBAREA01), (LOBAREA02)),
 EMPLOYEE t_EMPLOYEE ALLOCATE (ID_PHOTO
 IN ((LOBAREA03), (LOBAREA04)))
) IN ((RDAREA01) EMPLOYEE_NUMBER <= 700000, (RDAREA02));
```

Specification of boundary value:

```
CREATE TABLE STAFF_TABLE
(EMPLOYEE_NUMBER CHAR(6),
 DOCUMENT_DATA BLOB(64K) IN ((LOBAREA01), (LOBAREA02)),
 EMPLOYEE t_EMPLOYEE ALLOCATE (ID_PHOTO
 IN ((LOBAREA03), (LOBAREA04)))
) PARTITIONED BY EMPLOYEE_NUMBER
IN ((RDAREA01) 800000, (RDAREA02));
```

(2) Flexible hash partitioning or FIX hash partitioning

```
CREATE TABLE STAFF_TABLE
(EMPLOYEE_NUMBER CHAR(6),
 DOCUMENT_DATA BLOB(64K) IN ((LOBAREA01), (LOBAREA02)),
 EMPLOYEE_t EMPLOYEE ALLOCATE (ID_PHOTO
 IN ((LOBAREA03), (LOBAREA04)))
) [FIX] # HASH HASH6 BY EMPLOYEE_NUMBER
IN (RDAREA01, RDAREA02);
```

#: This specification is applicable to FIX hash partitioning.

6.5.3 Defining an index

This example defines an index for the `EMPLOYEE_NUMBER` column. Note that you cannot define an index for an abstract data type column.

Example

```
CREATE INDEX INDX1 ON STAFF_TABLE (EMPLOYEE_NUMBER)
IN ((RDAREA03), (RDAREA04));
```

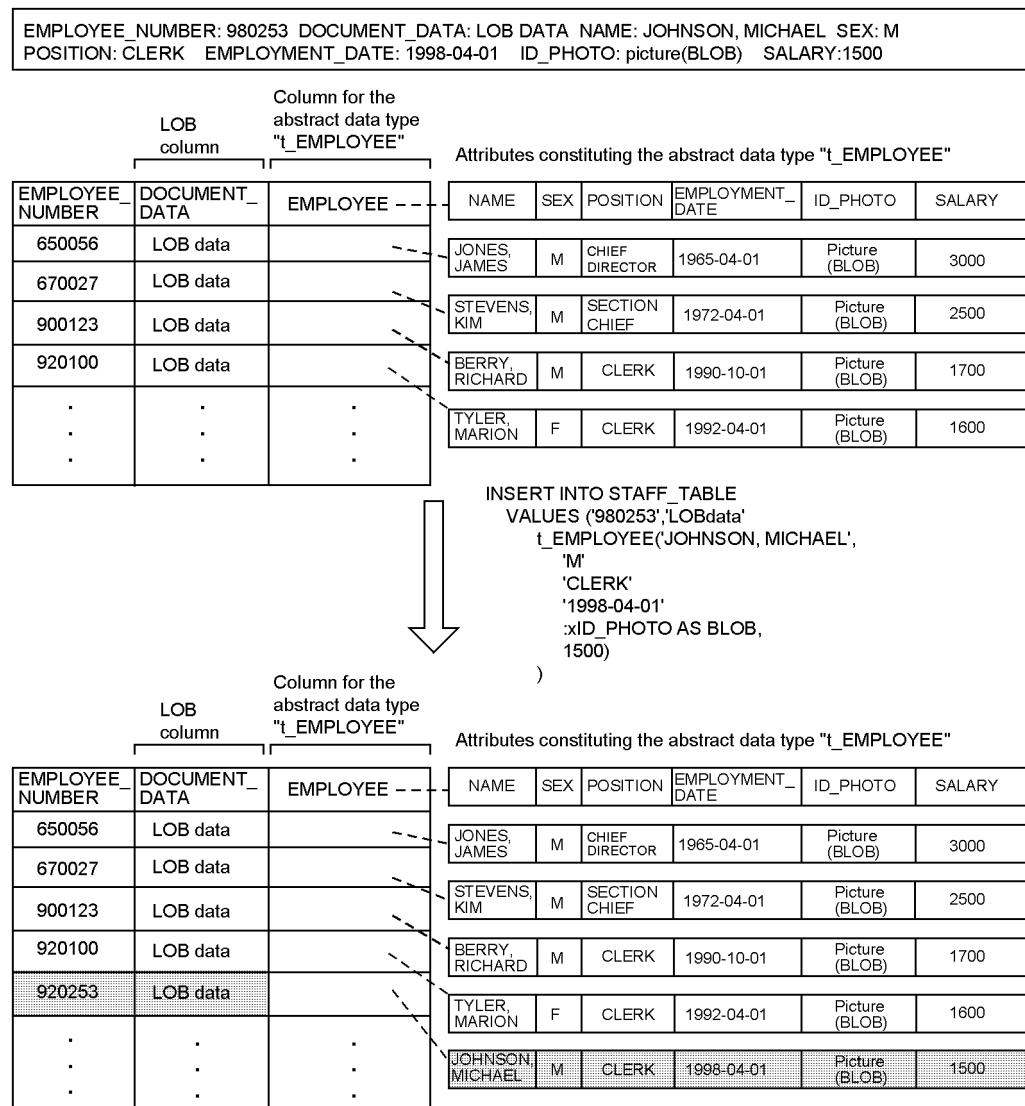
Explanation:

For the row-partitioned `STAFF_TABLE`, partitioning key index `INDX1` is divided and stored in user `RDAREAs` `RDAREA03` and `RDAREA04`. `EMPLOYEE_NUMBER` is specified for the columns that constitute the `INDX1` index.

6.5.4 Storing data in a table

To store data in a table containing a user-defined abstract data type, use the `INSERT` statement of the data manipulation SQL. You cannot use the database load utility (`pdload`) to load this data. To insert data, use the `INSERT` statement to insert a value that is generated by a defined function. The following figure shows the procedure for inserting data into a table that contains an abstract data type column.

Figure 6–8: Procedure for inserting data in a table that contains an abstract data type column



Note: xID_PHOTO assumes that the actual picture has been set by an embedded variable of BLOB type.

6.5.5 Database update log acquisition methods

(1) Database update log acquisition methods

There are three database update log acquisition methods:

1. Log acquisition mode

This mode acquires a database update log required for rollback and rollforward. It is used to create additional data or to reorganize data when there are not many data items.

2. Pre-update log acquisition mode

This method acquires only the database update log required for rollback. It is used to create, add, or reorganize data when there are many data items.

3. No-log mode

This mode does not acquire a database update log. It is used to create or reorganize data when there is only one table per RDAREA (if the table is partitioned, only one row-partitioned table per RDAREA) and any related index is also placed in one RDAREA.

(2) Specification of the database update log acquisition method

There are two ways to specify the database update log acquisition method:

- With `PDDBLOG` in the client environment definition^{#1}
- With the `RECOVERY` operand of `CREATE TABLE`^{#2}

#1: This way is used to specify the database update log acquisition method for a UAP that updates user RDAREAs.

#2: This way is used to specify the database update log acquisition method for a UAP that updates user LOB RDAREAs.

Note

The database update log acquisition method for user LOB RDAREAs (`RECOVERY` operand of `CREATE TABLE`) may depend on a specification in the client environment definition, as explained in the following table.

Table 6–7: Database update log acquisition method for user LOB RDAREAs depending on a specification in the client environment definition

Client environment definition PDDBLOG	RECOVERY operand value in CREATE TABLE		
	ALL	PARTIAL	NO
ALL	ALL	PARTIAL	NO
NO	NO	NO	NO

ALL: Log acquisition mode

PARTIAL: Pre-update log acquisition mode

NO: No-log mode

For example, if `PARTIAL` is specified in the `RECOVERY` operand of `CREATE TABLE` and the log acquisition method is set to `NO` in the client environment definition, then `NO` (no-log mode) is set for the user LOB RDAREAs.

6.5.6 Checking the data storage status

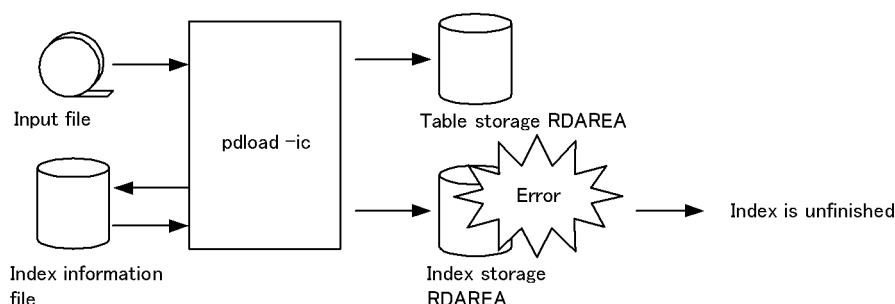
When data is inserted into a table containing an abstract data type column, the database condition analysis utility (`pddbst`) should be executed first to check the data storage status. This utility can check whether the database has been created exactly as designed.

The database condition analysis utility (`pddbst`) can obtain information about the data storage status (physical analysis only) of each RDAREA.

6.6 Handling errors during batch index creation

If an error occurs during batch index creation, data may have been stored successfully in the table, but the index may not have been created. This section explains the procedure for recovering from such a situation. The following figure shows the status of the index in the event of an error during batch index creation using the database load utility (pdload).

Figure 6–9: Status of index in the event of an error during batch index creation by database load utility (pdload)



Note: In the no-log mode, the table and index storage RDAREAs are placed on no-log shutdown status. If the KFPL703-I message is output, data has been stored in the table. If the index storage RDAREA is separate from the table storage RDAREA in such a case, the table storage RDAREA can be released from the shutdown status.

6.6.1 When data was loaded in log acquisition mode or pre-update log acquisition mode

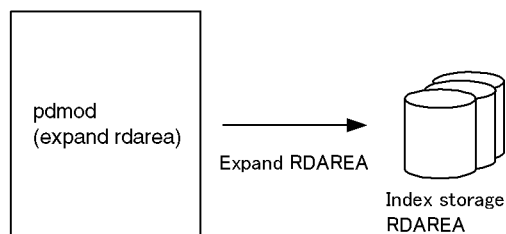
This section explains the procedure for handling errors that may occur during data loading in the log acquisition mode or pre-log acquisition mode.

If the table has a plug-in index, this procedure assumes that the applicable plug-in provides the batch plug-in index creation partial recovery facility. If your plug-in does not have the batch plug-in index creation partial recovery facility, see 6.6.2 *When data was loaded in no-log mode*.

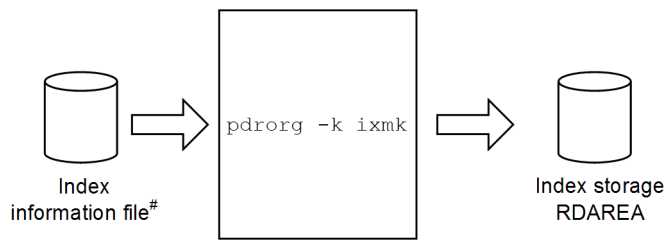
Restore the index storage RDAREAs according to the cause of error. The following shows the procedure:

(1) Error due to shortage of space in index storage RDAREA

1. Expand the index storage RDAREA.

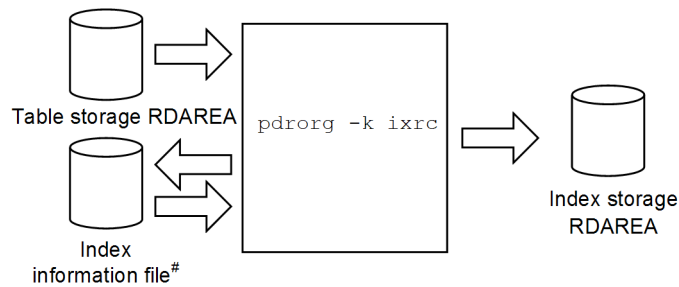


2. Create the index. Use the index information file created by the database load utility (pdload) to create the index in the batch mode.



#: The key information for the additional keys that are loaded and the existing data is output to this file. In the case of a plug-in index, only the key information for the additional data that is loaded is output.

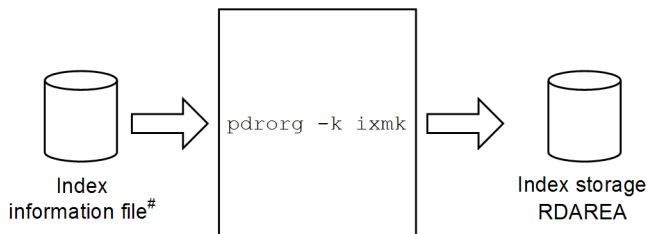
If the index information file created by the database load utility (pdload) is not available, re-create the index with the database reorganization utility (pdrorg).



#: This file is created by the database reorganization utility (pdrorg) and is used to output the key information for the additional keys that are loaded and the existing data.

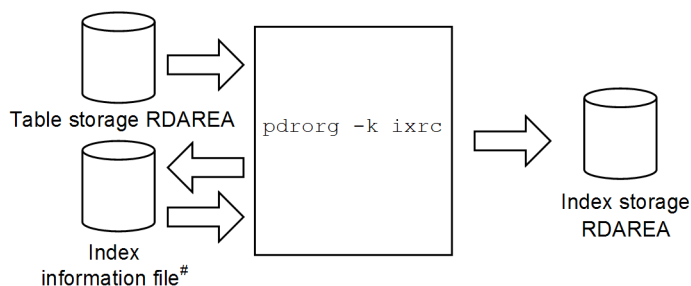
(2) Sorting error (message KFPL15062-E is output) or an error due to forced termination of the utility by the pdcancel command

1. Create the index. Use the index information file created by the database load utility (pdload) to create the index in the batch mode.



#: This file contains the key information for the additional keys that are loaded and the existing data. In the case of a plug-in index, only the key information for the additional data that is loaded is output.

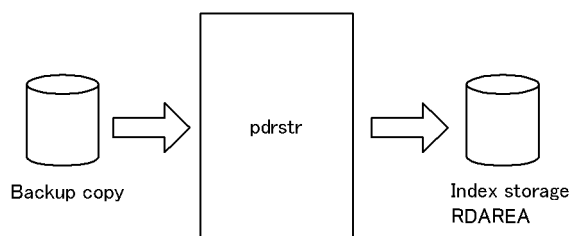
If the index information file created by the database load utility (pdload) is not available, re-create the index with the database reorganization utility (pdrorg).



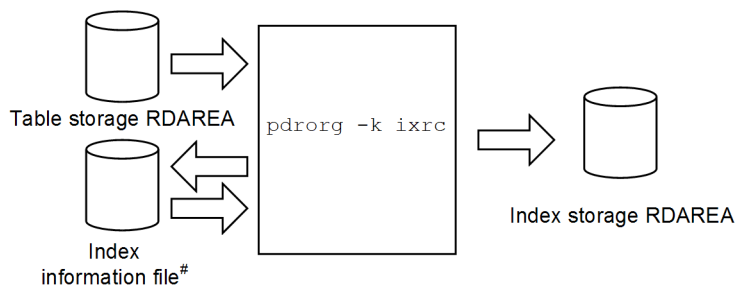
#: This file is created by the database reorganization utility (`pdrorg`) and is used to output the key information for the additional keys that are loaded and the existing data.

(3) Error due to a disk failure

1. Replace the faulty disk and use a backup copy to restore the status before the utility executed. To prevent accesses, place the RDAREAs in shutdown status until the index storage executed in step 2 has been completed.



2. Re-create the index with the database reorganization utility (`pdrorg`).



#: This file is created by the database reorganization utility (`pdrorg`) and is used to output the key information for the additional keys that are loaded and the existing data.

6.6.2 When data was loaded in no-log mode

This section explains the procedure for handling errors that may occur during data loading in no-log mode or when your plug-in does not provide the batch plug-in index creation partial recovery facility.

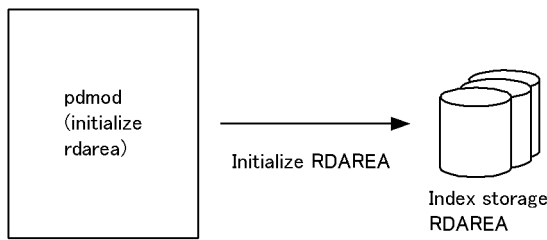
Restore the index storage RDAREAs according to the cause of error. The following shows the procedure:

(1) Error due to shortage of space in index storage RDAREA

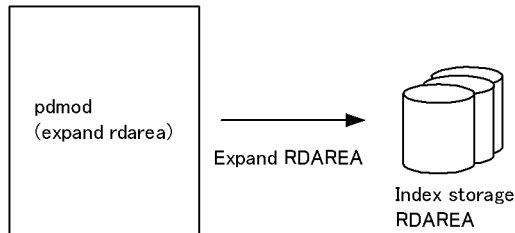
1. Reinitialize the index storage RDAREA.

You can also recover from a backup. In such a case, until index storage has been completed, place the RDAREA in shutdown status in order to prevent accesses. In the following case, recovery must be performed using a backup:

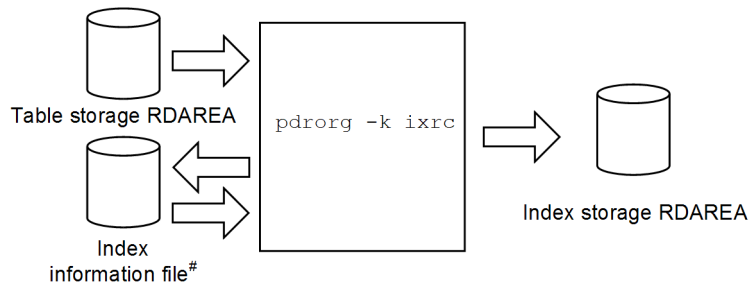
- The relevant index is stored in an RDAREA with different tables, indexes, or falsification prevented tables.



2. Expand the index storage RDAREA.



3. Re-create the index with the database reorganization utility (pdorg).



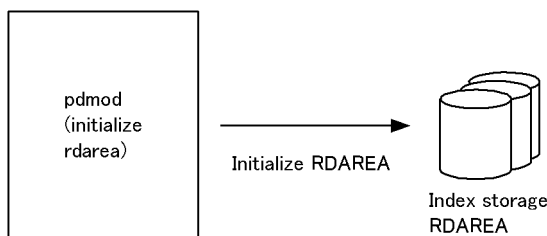
#: This file is created by the database reorganization utility (pdorg) and is used to output the key information for the additional keys that are loaded and the existing data.

(2) Sorting error (message KFPL15062-E is output) or an error due to forced termination of the utility by the pdcancel command

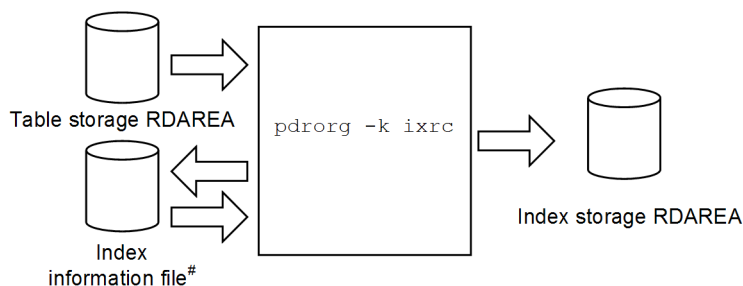
1. Reinitialize the index storage RDAREA.

You can also recover from a backup. In such a case, until index storage has been completed, place the RDAREA in shutdown status in order to prevent accesses. In the following case, recovery must be performed using a backup:

- The relevant index is stored in an RDAREA with different tables, indexes, or falsification prevented tables.



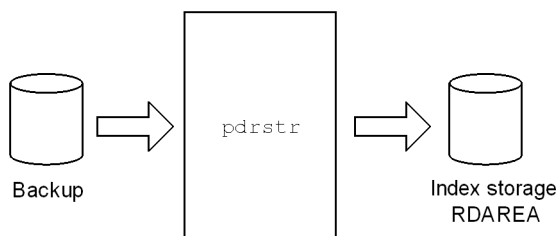
2. Re-create the index with the database reorganization utility (pdorg).



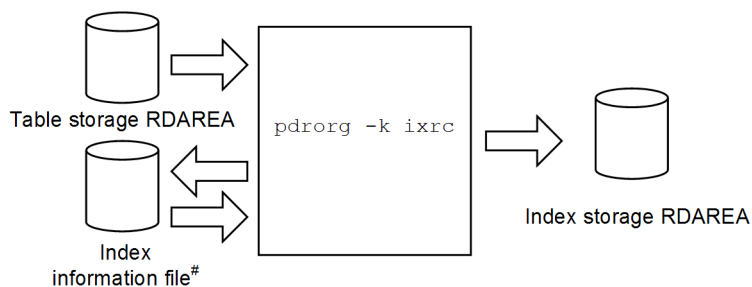
#: This file is created by the database reorganization utility (`pdrorg`) and is used to output the key information for the additional keys that are loaded and the existing data.

(3) Error due to a disk failure

1. Replace the faulty disk and use a backup copy to restore the status before the utility executed.



2. Re-create the index with the database reorganization utility (`pdrorg`).



#: This file is created by the database reorganization utility (`pdrorg`) and is used to output the key information for the additional keys that are loaded and the existing data.

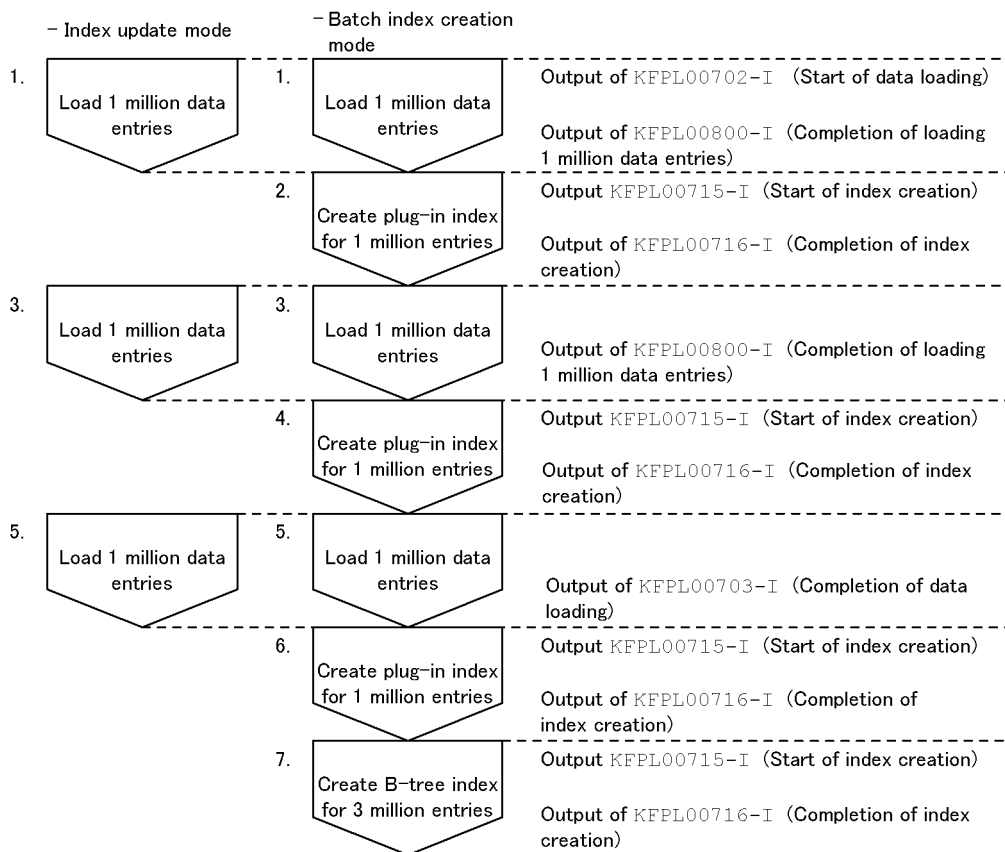
6.7 Handling utility abnormal termination errors during data loading with the synchronization point specification

This section describes the error handling procedure in the event the database load utility terminates abnormally during data loading with the synchronization point specification.

6.7.1 Overview of error handling procedure

The procedure depends on the abnormal termination timing. The following figure shows the error handling procedure in the event of abnormal termination of a utility during data loading with synchronization point specification.

Figure 6–10: Error handling procedure in the event of abnormal termination of a utility during data loading with the synchronization point specification



Explanation:

- The total number of entries subject to data loading is three million with one million line numbers with synchronization points.
- If the utility terminates abnormally in step 1, 3, or 5, reexecute data loading.
- If the utility terminates abnormally in step 2 or 4, re-create the plug-in index using the database reorganization utility's batch index creation facility (`-k ixmk`), and then reexecute data loading.
- If the utility terminates abnormally in step 6, re-create the plug-in index using the database reorganization utility's batch index creation facility (`-k ixmk`), and then create the B-tree index using the database reorganization utility's index re-creation facility (`-k ixrc`).

- If the utility terminates abnormally in step 7 and an index information file has already been created (the KFPL00710-I message is output), create the B-tree index using the database reorganization utility's batch index creation facility (-k ixmk). If the index information file has not been created, create the B-tree index using the database reorganization utility's index re-creation facility (-k ixrc).

6.7.2 Example

This example assumes that the database load utility terminated while loading three million entries of data in the batch index creation mode with one million line numbers with synchronization points.

(1) Checking the messages

The following messages are output:

```
KFPL00800-I Loading until 2000000th row committed
KFPL00710-I Index information file assigned, index=k87m271."INDX01",
RDAREA="LOB02", file=/pdrorg/INDX01_2
KFPL00715-I Index load started at bes2, index=k87m271."INDX01", RDAREA="LOB02"
```

Explanation:

- The KFPL00800-I message indicates that two million data entries have already been loaded.
- The KFPL00715-I message indicates that the creation of the plug-in index has begun, but a completion message corresponding to this message (KFPL00716-I) has not been output.

This indicates that the utility terminated abnormally while creating the plug-in index for one to two million entries.

(2) Using the pdrorg command to create the plug-in index in batch mode

Use the database reorganization utility to create the plug-in index for one million (one to two million) entries in batch mode.

```
pdrorg -k ixmk -t TABLE1 /pdrorg/rorg01
```

Explanation:

-k: Specifies ixmk to create the plug-in index in batch mode.

-t: Specifies the name of the table.

/pdrorg/rorg01: Specifies the name of the control statement file for the pdrorg command.

The contents of the control statement file are shown as follows. The index information file that is specified in the control statement file is indicated in the KFPL00710-I message, which was output in (1).

```
index INDX01 LOB02 /pdrorg/INDX01_2
```

(3) Reexecuting data loading

```
pdload TABLE1 /pdload/load01
```

Explanation:

There is no need to change the option specification.

(4) Making a backup of the target data storage RDAREAs

Make a backup of the target data storage RDAREAs. For details about making backups in units of RDAREAs, see the *HiRDB Version 9 System Operation Guide*.

(5) Using the `pdrels` command to release the target data storage RDAREAs from shutdown status

Enter the `pdrels` command to release the table and index storage RDAREAs from shutdown status.

```
pdrels -r DATA01,DATA02,DATA03,INX01,INX02,INX03,LOB01,LOB02,LOB03
```

After the command has executed, the execution results should be checked for errors. For details about how to check command execution results, see the manual *HiRDB Version 9 Command Reference*.

7

Linking to Other Products

This chapter describes how to link HiRDB to other products.

7.1 Linking to the replication facility

This section describes the information that you need to specify to use HiRDB's replication facility (HiRDB Datareplicator and HiRDB Dataextractor).

7.1.1 Linking to HiRDB Datareplicator

HiRDB Datareplicator enables the user to extract data automatically and incorporate it into a HiRDB database when another HiRDB database is updated. To use HiRDB Datareplicator, you specify the following operands in the HiRDB system common definitions:

- `pd_rpl_init_start` operand
This operand specifies whether to use the HiRDB Datareplicator linkage facility from the time of HiRDB startup.
- `pd_rpl_hdepath` operand
This operand specifies the name of the HiRDB Datareplicator directory where data is extracted. This directory name must be the one that has been specified in the `HDEPATH` environment variable for the HiRDB Datareplicator where data is extracted.
- `pd_log_rpl_no_standby_file_opr` operand
This operand specifies the desired operation in the event a swap request is issued while the HiRDB Datareplicator linkage facility is being used, and none of the system log files can be swapped because extraction of system log information at the HiRDB Datareplicator has not been completed.

For details about the system environment definition and how to use HiRDB Datareplicator to perform data replication, see the *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide*.

Notes

- If you are using a HiRDB facility not supported by HiRDB Datareplicator, you might not be able to use the data linkage facility. For details, see the *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide*. For the most recent, detailed information, see the online manuals published on the HiRDB website.
- When a recovery-unnecessary front-end server is used
Because a recovery-unnecessary front-end server cannot execute import processing using the two-phase commitment method for the synchronization point processing method (enabled when `fxa_sqle` is specified in the import system definition `commitment_method` operand) of the target HiRDB Datareplicator, you need to use a front-end server other than the recovery-unnecessary front-end server. For details, see *9.1.4 Recovery-unnecessary front-end server*.

7.1.2 Linking to HiRDB Dataextractor

HiRDB Dataextractor enables the user to extract data from a mainframe or HiRDB database in batch mode and store it sequentially in a HiRDB database. HiRDB Dataextractor has the following features:

- Data in a central database can be incorporated by batch-mode processing into a departmental database at a specified point in time. This capability enables tables to be created specifically for the departmental database or to refresh all data.
- A portion of the data can be extracted from the central database to create a departmental database suitable for each application.

For details about HiRDB Dataextractor, see the *HiRDB Dataextractor Version 8 Description, User's Guide and Operator's Guide*.

7.2 Linking with an OLTP system

This section describes the procedure for using the X/Open XA interface to link HiRDB to an OLTP system. The topics covered include:

1. OLTP products supported for linking
2. HiRDB XA library
3. Example of HiRDB system configuration with OLTP linkage
4. Transaction transfer
5. Registering HiRDB in the transaction manager
6. Information to be registered in the transaction manager
7. Example of registering in the transaction manager
8. Modifying the registration information in the transaction manager
9. Methods for re-establishing connection between the transaction manager and HiRDB
10. Monitoring HiRDB using the TP1/Resource Manager Monitor facility
11. Notes

7.2.1 OLTP products supported for linking

HiRDB supports the following OLTP products for linking:

- OpenTP1
- TPBroker for C++
- TUXEDO
- WebLogic Server
- OpenTP1/Server Base Enterprise Option (referred to hereafter as TP1/EE)

Some of the OLTP products may not be supported, depending on the type of HiRDB in use. The following table shows OLTP linkage support depending on the HiRDB type.

Table 7–1: OLTP linkage support depending on the HiRDB type

HiRDB type	OLTP product type				
	OpenTP1	TPBroker for C++	TUXEDO	WebLogic Server	TP1/EE
HP-UX	S	S	N	S	N
Solaris	S	S	S	S	N
AIX	S	N	N	N	S
Linux	S	N	N	N	S

S: Supported.

N: Not supported.

7.2.2 HiRDB XA library

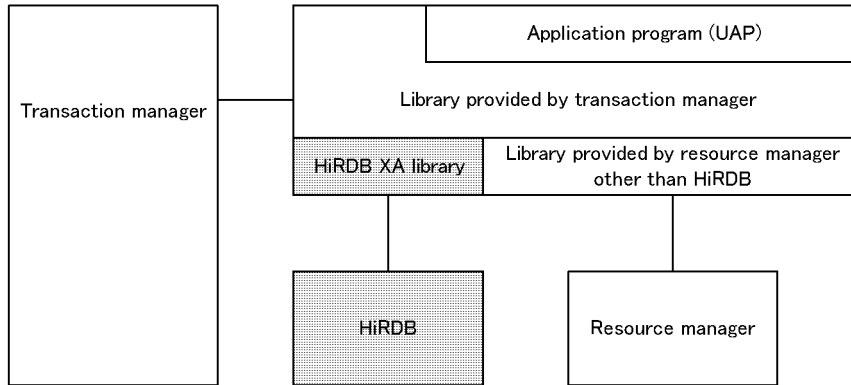
The X/Open XA interface is an X/Open standard specification that stipulates the interface between a transaction manager (TM) and a resource manager (RM) in a distributed transaction processing (DTP) system. The XA interface enables the transaction manager to control the resource manager's transaction processing. In order for the transaction

manager to control the resource manager's transaction processing, a library provided by the resource manager and a library provided by the transaction manager must be linked to an application program.

HiRDB provides the HiRDB XA library to enable a transaction manager to control HiRDB transaction processing. This HiRDB XA library complies with the XA interface specifications based on the X/Open DTP software architecture.

The following figure shows the relationship between HiRDB and the X/Open DTP model.

Figure 7–1: Relationship between HiRDB and X/Open DTP model



Note that if connection is established from a UAP that is using the X/Open XA interface to a recovery-unnecessary front-end server, the SQL statement returns an error. Specify `PDFESHOST` and `PDSERVICEGRP` in the client environment definition and connect to a front-end server that is not the recovery-unnecessary front-end server.

(1) Functions supported by the HiRDB XA library

The following table shows the functions supported by the HiRDB XA library.

Table 7–2: Functions supported by HiRDB XA library

Function	Description
Transaction transfer	Executes transaction commit processing by a process other than the one at the time the UAP accessed HiRDB (UAP here refers to the user application program that established the connection with HiRDB using the HiRDB XA library). Whether to use the transaction transfer function depends on the <code>PDXAMODE</code> client environment definition operand. For details about the transaction transfer function, see <i>7.2.4 Transaction transfer</i> .
Single-phase optimization	Optimizes two-phase commitment control to one phase. When single-phase optimization is used, the transaction completion types of the transaction manager and HiRDB do not always match. For details, see <i>7.2.11(3) Notes on single-phase optimization</i> .
Read only	When a HiRDB resource has not been updated by a prepare request, enables the transaction manager to optimize the processing without issuing a commit request at the second phase.
Dynamic transaction registration	Enables HiRDB to register dynamically a transaction immediately before executing a UAP.
Multi-connection facility	Executes multiple <code>CONNECTs</code> for HiRDB servers separately from one process. For details about the multi-connection facility in the X/Open XA interface environment, see the <i>HiRDB Version 9 UAP Development Guide</i> .

Note:

The HiRDB XA library does not provide asynchronous XA calls (facility that enables the transaction manager to call the HiRDB XA library asynchronously).

(2) XA interface supporting multi-thread (HP-UX 11.0 and Solaris only)

An XA interface supporting multi-thread enables you to link Object Transaction Service (OTS) with TPBroker for C++ and HiRDB.

The multi-thread libraries support only C and C++ languages. They do not support COBOL.

To use an XA interface supporting multi-thread, link a dedicated HiRDB client library. A HiRDB client library version earlier than 05-06 does not support multi-thread. You can use a library for multi-thread with any HiRDB server that supports existing HiRDB clients. The library for multi-thread enables connection to be shared between threads.

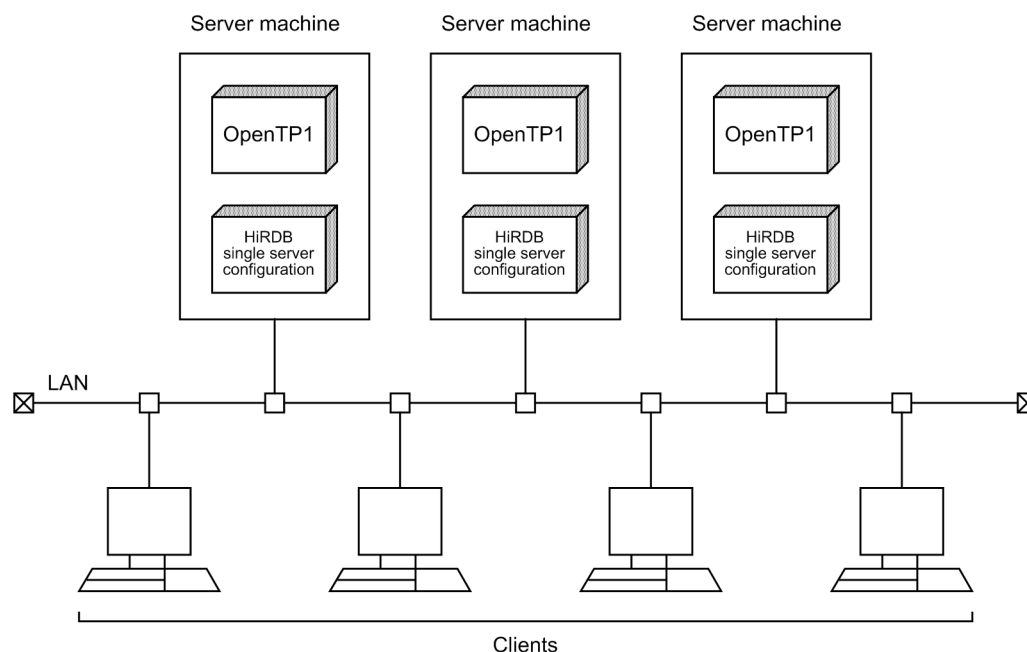
7.2.3 Example of HiRDB system configuration with OLTP linkage

This section describes a HiRDB system linked with OLTP by way of an example using OpenTP1.

(1) Linking with a HiRDB single server configuration

You can execute the update processing of multiple HiRDB single server configurations as a single transaction by linking OLTP (OpenTP1) to the HiRDB single server configurations. If you partition a database by key ranges, the OLTP system (OpenTP1) running on each server machine can distribute processing to the linked HiRDB single server configurations. This enables transaction processing to be performed at high speed. When integrating multiple HiRDB single server configurations, consider linking your system to an OLTP system. The following figure shows linkage of a HiRDB single server configuration with an OLTP system (OpenTP1).

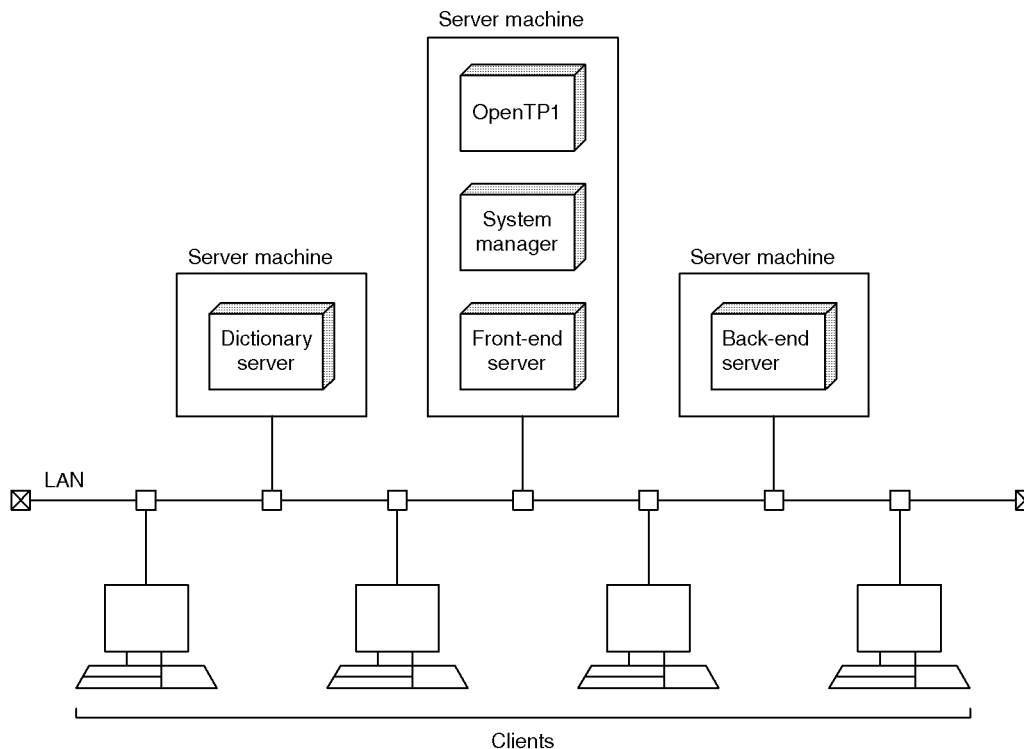
Figure 7-2: Linking a HiRDB single server configuration with an OLTP system (OpenTP1)



(2) Linking with a HiRDB parallel server configuration

When a HiRDB parallel server configuration is linked with an OLTP system (OpenTP1), transaction processing can be achieved at high performance and high workload. The following figure shows linkage of a HiRDB parallel server configuration with an OLTP system (OpenTP1).

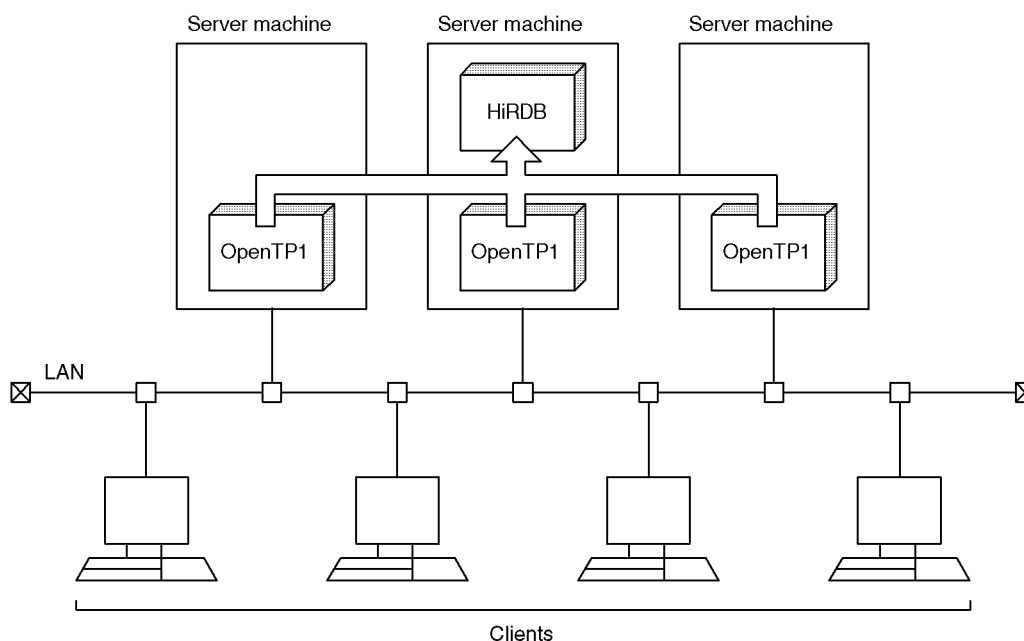
Figure 7–3: Linking a HiRDB parallel server configuration with an OLTP system (OpenTP1)



(3) Linking between multiple OLTPs (OpenTP1) and a single HiRDB

In this type of linking, multiple OLTPs (OpenTP1) and one HiRDB communicate using the client/server method. Different OLTPs (OpenTP1) can connect to one HiRDB at the same time. To do this, you must set a unique OLTP identifier (client environment definition `PDTMID`) for each OLTP (OpenTP1). The following figure shows linkage between multiple OLTPs (OpenTP1) and a single HiRDB.

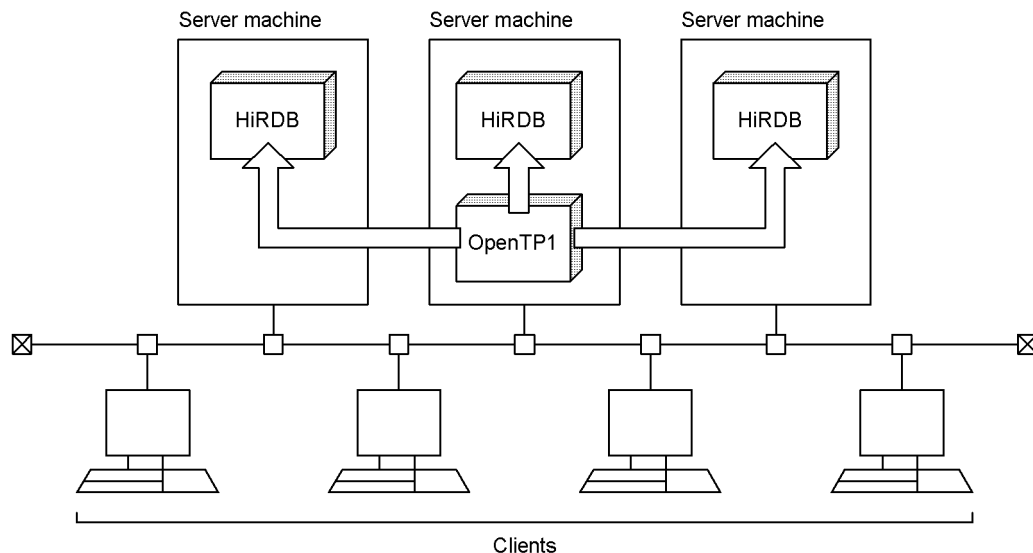
Figure 7–4: Linking between multiple OLTPs (OpenTP1) and a single HiRDB



(4) Linking between one OLTP (OpenTP1) and multiple HiRDBs

In this type of linking, one OLTP (OpenTP1) is linked to multiple HiRDBs. SQL statements can be executed by connecting to HiRDBs on different server machines at the same time. In such a case, you need to use the multi-connection facility. The following figure shows linkage between one OLTP (OpenTP1) and multiple HiRDBs.

Figure 7–5: Linking between one OLTP (OpenTP1) and multiple HiRDBs

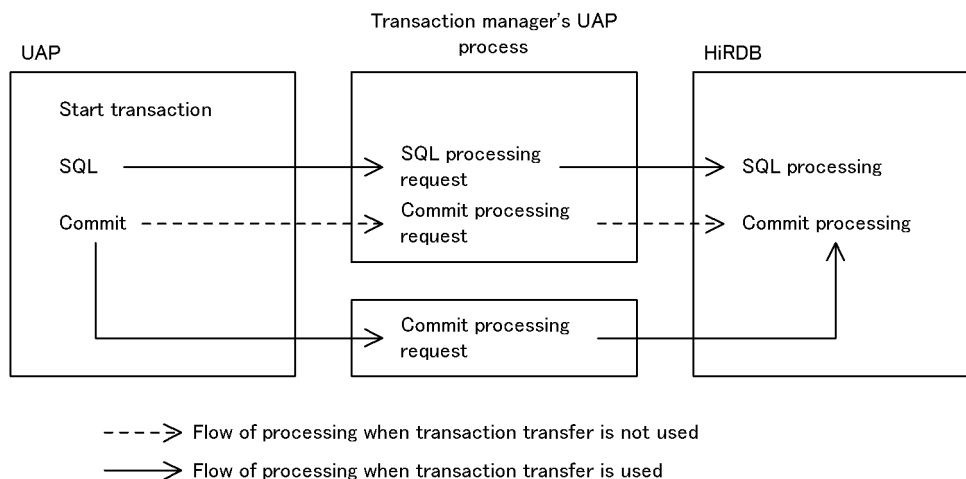


For details about the multi-connection facility, see the *HiRDB Version 9 UAP Development Guide*.

7.2.4 Transaction transfer

For a UAP that connects to HiRDB using the HiRDB XA library, you can execute transaction commit processing by a process other than the one used at the time the UAP accessed HiRDB. This is called *transaction transfer*. The following figure gives an overview of transaction transfer.

Figure 7–6: Overview of transaction transfer



Advantages

When transaction transfer is used, the transaction manager's UAP processing can accept the next service request without having to wait for transaction completion. This makes it possible to execute a UAP with fewer processes. However, the number of server processes used by HiRDB and the number of lock-release waits[#] may increase.

#: Compared to when transaction transfer is not used, accesses to HiRDB for the next service request are more likely to be placed in lock-release wait status until the transaction is completed.

Criteria

When the transaction manager is to use the transaction transfer function, HiRDB must also use the transaction transfer function.

When the transaction manager is not to use the transaction transfer function, HiRDB must not use the transaction transfer function.

The following should be noted when the transaction manager can set whether the transaction transfer function is to be used:

- The transaction transfer function should be used when the UAP processing workload is greater than the HiRDB access workload.

Operating procedure

To use this function, 1 must be set in the `PDXAMODE` operand of the client environment definition; to not use this function, either 0 must be set in this operand or specification of the operand must be omitted.

For details about the `PDXAMODE` operand, see the *HiRDB Version 9 UAP Development Guide*.

Notes

1. If the information indicating whether the transaction transfer function is to be used does not match between the transaction manager and HiRDB, transactions may not be settled, HiRDB may terminate abnormally, or control may be returned to the transaction manager with an error.
2. When this function is used, the scope of `LOCK TABLE UNTIL DISCONNECT` in the `LOCK` statement changes. For details about the scope of `LOCK TABLE UNTIL DISCONNECT`, see the *HiRDB Version 9 UAP Development Guide*.

(1) OpenTP1 used as the transaction manager

When this function is used, HiRDB supports OpenTP1's commit optimization and prepare optimization. Therefore, this function should be used when the `-d` option is omitted from OpenTP1's `trnstring` operand. When the `-d` option is specified, this function should not be used.

The following table shows the relationship between the transaction service definition's `trnstring` operand in the OpenTP1 system definition and HiRDB's `PDXAMODE` operand.

Table 7–3: Relationship between OpenTP1's `trnstring` operand and HiRDB's `PDXAMODE` operand

<code>trnstring</code> operand specification	<code>PDXAMODE</code> operand value
<code>-d</code> option omitted	1
<code>-d</code> option specified	0

Note

- If the `trnstring` operand's value does not match the `PDXAMODE` operand's value, HiRDB cannot settle transactions. In this case, HiRDB returns to OpenTP1 an XA function error return code (–6).
- The `-d` option is supported by TP1/Server Base Version 03-03 or subsequent.

For details about the `trnstring` operand, see the manual *OpenTP1 System Definition*. For details about commit optimization and prepare optimization, see the manual *OpenTP1 Programming Guide*.

7.2.5 Registering HiRDB in the transaction manager

To link your HiRDB to OLTP, you need to register the HiRDB in the transaction manager. You can use each transaction manager's commands and functions to register HiRDB in the transaction manager:

- OpenTP1: Use the `trnlncrm` command to register HiRDB.
- TPBroker for C++: Use the `tslncrm` command to register HiRDB.

- TUXEDO: Register HiRDB in `$TUXDIR/udataobj/RM`. `$TUXDIR` indicates the absolute path name of the directory that contains the TUXEDO system software.
- WebLogic Server: Register HiRDB using the driver class name and the provider for the WebLogic Server JDBC connection pool.
- TP1/EE: Use the `eeetrmkobj` command to register HiRDB.

(1) Dynamic registration and static registration

There are two ways to register HiRDB as the source manager in the transaction manager:

- Dynamic registration
- Static registration

You cannot use both dynamic and static registration with a single transaction manager.

For a WebLogic Server, only static registration can be used.

(a) Dynamic registration

If you dynamically register HiRDB in the transaction manager, HiRDB is placed under the control of the transaction manager when the UAP issues the first SQL statement within a transaction. This method reduces the transaction manager's transaction control overhead for HiRDB when the UAP accesses multiple resources including HiRDB, or when the application program may not access HiRDB at all.

(b) Static registration

If you statically register HiRDB in the transaction manager, HiRDB is placed under the control of the transaction manager when a transaction is started, whether the UAP issues any SQL statements. When OpenTP1 is used as the transaction manager and the connection between a UAP and HiRDB is broken (due to abnormal termination of a unit or server process, or other similar problem), OpenTP1 re-establishes connection at the time a transaction is started. Therefore, there is no need to restart the UAP.

(2) Differences between dynamic and static registration

The following table shows the differences between dynamic and static registration.

Table 7–4: Differences between dynamic and static registration

Timing	Dynamic registration	Static registration
When transaction is started	Performs no management.	<ul style="list-style-type: none"> • Checks to see if connection is being established. • Starts managing transaction under transaction manager's control. • Establishes connection.^{#1}
When first SQL is issued within transaction	<ul style="list-style-type: none"> • Starts managing transaction under transaction manager's control. • Starts HiRDB transaction. • Processes SQL. • Establishes connection.^{#1} 	<ul style="list-style-type: none"> • Starts HiRDB transaction. • Processes SQL statements.
Number of times communication is established between transaction manager and HiRDB during transaction	Number of SQL statements + number of commit processing communications + 1 (communication for establishing connection) ^{#1}	Number of SQL statements + number of commit processing communications + 1 (for transaction startup processing) + 1 (communication for establishing connection) ^{#1}
Reconnection method if connection between transaction manager and HiRDB is broken during processing ^{#2}	Reestablishes connection automatically the next time transaction is started. ^{#3}	Reestablishes connection automatically the next time transaction is started. ^{#4}

#1

Applicable when the XA interface supporting multi-thread is used.

#2

Connections broken as a result of network errors cannot be detected. However, if the transaction manager is TPBroker for C++ or Weblogic Server, connections are established when a transaction starts, so reconnection is possible.

#3

If the transaction manager is OpenTP1/Server Base, specifying `transaction` in the OpenTP1/Server Base operand `trn_rm_open_close_scope` enables reconnection even when a connection is broken as a result of a network error.

#4

If the transaction manager is OpenTP1/Server Base, reconnection is possible even in the event of a connection broken as a result of a network error. Also, when the transaction manager is TP1/EE, specifying `YES` in the HiRDB client environment definition `PDXAAUTORECONNECT` enables reconnection when a connection is broken as a result of a network error.

7.2.6 Information to be registered in the transaction manager

For details about how to register HiRDB as the resource manager in the transaction manager, see the applicable transaction manager documentation. Specification of information in the transaction manager is explained as follows.

For a WebLogic Server, perform the operations listed beginning in (6) below.

(1) RM switch name

HiRDB's RM switch name determines whether dynamic or static registration is used. The following shows the HiRDB RM switch name (`xa_switch_t structure-name`):

- Dynamic registration: `pdtxa_switch`
- Static registration: `pdtxa_switch_y`

(2) RM name

The RM name(resource manager name) defined in the RM switch (`xa_switch_t structure`) is `HiRDB_DB_SERVER`.

(3) Open character string

If you are using the multi-connection facility, specify the open character string to be used when the transaction manager opens the resource manager with `xa_open`. If you are not using the multi-connection facility, there is no need to specify an open character string. If the transaction manager is TP1/EE, specify the open character string for a single HiRDB that has been registered. For the TUXEDO or WebLogic Server transaction manager, you cannot use the multi-connection facility.

To use the multi-connection facility, register multiple HiRDBs in the transaction manager and specify the open character string for each HiRDB. For the open character string, specify the following information:

- The absolute path name of the file containing the environment variable settings that take effect at the destination
- Environment variable group ID

Use one of the following formats:

- `"environment variable group identifier + environment variable setup file name"`
- `"environment variable group identifier * environment variable setup file name"`

An open character string in any other format is ignored. The environment variable group ID always consists of four bytes, and the open character string cannot be longer than 256 bytes.

The following shows an example of registering the open character string for the OpenTP1, TPBroker for C++, or TP1/EE transaction manager.

(a) OpenTP1

Register the open character string using the `trnstring` operand in OpenTP1's transaction service definition. This example registers two HiRDBs in OpenTP1. The registration conditions are as follows:

Resource manager	Environment variable group ID	Environment variable setup file name
HiRDB1	HDB1	/usr/conf/HiRDB/HiRDB11.ini
		/usr/conf/HiRDB/HiRDB12.ini
HiRDB2	HDB2	/usr/conf/HiRDB/HiRDB21.ini
		/usr/conf/HiRDB/HiRDB22.ini

The following shows an example of registering the open character string:

```
trnstring -n HiRDB_DB_SERVER -i H1 -o "HDB1*/usr/conf/HiRDB/HiRDB11.ini"
-o "HDB1*/usr/conf/HiRDB/HiRDB12.ini"
trnstring -n HiRDB_DB_SERVER -i H2 -o "HDB2*/usr/conf/HiRDB/HiRDB21.ini"
-o "HDB2*/usr/conf/HiRDB/HiRDB22.ini"
```

Explanation:

- n: Specifies the name of the resource manager.
- i: Specifies the resource manager extension.
- o: Specifies the character string for the `xa_open` function for transaction service.
This is the open character string that is used by OpenTP1's transaction service process. The format is *environment-variable-group-ID * environment-variable-setup-filename*.
- O: Specifies the character string for the `xa_open` function for the user server.
This is the open character string that is used by the user server process. The format is *environment-variable-group-ID + environment-variable-setup-filename*
 - Specify the same environment variable group ID for `-o` and `-O`.
 - Specify the same environment variables in the file that are specified for `-o` and `-O`.

Note

You can select the HiRDB to be connected from the user service using the `trnrmid` operand in OpenTP1's user service definition. The following example connects to HiRDB1 and HiRDB2:

```
trnrmid -n HiRDB_DB_SERVER -i H1,H2
```

(b) TPBroker for C++

Register the open character string using the `xa_open_string_info` operand in TPBroker for C++'s resource manager definition. This example registers two HiRDBs in TPBroker for C++. The registration conditions are as follows:

Resource manager	Environment variable group ID	Environment variable setup file name
HiRDB1	HDB1	/usr/conf/HiRDB/HiRDB11.ini
		/usr/conf/HiRDB/HiRDB12.ini
HiRDB2	HDB2	/usr/conf/HiRDB/HiRDB21.ini
		/usr/conf/HiRDB/HiRDB22.ini

The following shows an example of registering the open character string:

```
tsdefvalue /OTS/RM/HiRDB_DB_SERVER_1/DMN/xa_open_string_info 1
-s "HDB1*/usr/conf/HiRDB/HiRDB11.ini"
tsdefvalue /OTS/RM/HiRDB_DB_SERVER_1/xa_open_string_info 2
```

```
-s "HDB1+usr/conf/HirDB/HirDB12.ini"

tsdefvalue /OTS/RM/HirDB DB SERVER 2/DMN/xa_open_string_info 1
-s "HDB2*usr/conf/HirDB/HirDB21.ini"
tsdefvalue /OTS/RM/HirDB DB SERVER 2/xa_open_string_info 2
-s "HDB2+usr/conf/HirDB/HirDB22.ini"
```

Explanation:

1. For `/OTS/RM/RM-name/DMN/xa_open_string_info`, specify the open character string that is used by TPBroker for C++'s recovery process. Separate the environment variable group ID and environment variable setup file name by an asterisk (*).
 2. For `/OTS/RM/RM-name/xa_open_string_info`, specify the open character string that is used by the application program process and settlement process. Separate the environment variable group ID and environment variable setup file name by a plus sign (+).
- If the *RM-name* is the same, specify the same environment variable group ID.
 - If the *RM-name* is the same, specify the same environment variable content for each environment variable setup file.
 - If the TPRMINFO environment variable is specified for the settlement process, specify the character string specified in `/OTS/RM/RM-name/xa_open_string_info` as the open character string for `/OTS/RM/RM-name/ (TPRMINFO-value) /xa_open_string_info`. If the multi-connection facility is used, specify 'TPRMINFO=' as the default in `/OTS/completion_process_env` even when TPRMINFO is not specified for the settlement process. The following shows an example:

```
tsdefvalue /OTS completion_process_env -a 'TPRMINFO='
```

(c) TP1/EE

Register the open character string using the `trnstring` operand in the TP1/EE transaction-related definition. This example registers two HiRDBs in TP1/EE. The registration conditions are as follows:

Resource manager	Environment variable group ID	Environment variable setup file name
HiRDB1	HDB1	/usr/conf/HirDB/HirDB11.ini
		/usr/conf/HirDB/HirDB12.ini
HiRDB2	HDB2	/usr/conf/HirDB/HirDB21.ini
		/usr/conf/HirDB/HirDB22.ini

The following shows an example of registering the open character string:

```
trnstring -n HirDB_DB_SERVER -i H1 -o "HDB1*usr/conf/HirDB/HirDB11.ini" \
-o "HDB1+usr/conf/HirDB/HirDB12.ini"
trnstring -n HirDB_DB_SERVER -i H2 -o "HDB2*usr/conf/HirDB/HirDB21.ini" \
-o "HDB2+usr/conf/HirDB/HirDB22.ini"
```

Explanation:

- n: Specifies the name of the resource manager.
- i: Specifies the resource manager extension.
- o:
 - Specifies the open character string that is used by TP1/EE's recovery and monitoring threads.
 - The format is *environment-variable-group-ID*environment-variable-setup-file-name*.
- O:
 - Specifies the open character string that is used by the process thread.
 - The format is *environment-variable-group-ID+environment-variable-setup-file-name*.
- In -o and -O, specify the same environment variable group ID.
- Set the same environment variables in the files that are set in -o and -O.

(4) Close character string

There is no need to specify a character string to enable the transaction manager to close the resource manager using `xa_close`.

(5) RM-related object name

For the RM-related object name, specify the library name listed in the following table. The suffix for the shared library depends on the platform. The suffix is `.so` in Solaris and Linux, and `.a` in AIX. The archive library name is common to all platforms.

Table 7–5: Library names for RM-related object names

Library type		Library name
Shared library	Single thread	<code>libzclty.sl</code> (<code>libzcltx.sl</code>)
	Single thread (Multi-connection facility supported)	<code>libzcltys.sl</code> ^{#1} (<code>libzcltxs.sl</code>)
	Multi-thread (Multi-connection facility supported) ^{#2}	<code>libzcltyk.sl</code> (<code>libzcltxk.sl</code>)
Archive library	Single thread	<code>libcltya.a</code> (<code>libcltxa.a</code>)

Notes

1. When using dynamic registration, you may specify the library name enclosed in parentheses.
2. Select either a shared library or archive library, whichever is appropriate to the purpose. It is more advantageous to select the shared library because it has the following benefits:
 - The object size of the UAP is small.
 - There is no need to relink when upgrading HiRDB.

#1: Specify this library name when linking to TUXEDO.

#2: Applicable to HI-UX 11.0 and Solaris.

(6) Client environment definition

To enable the transaction manager to control HiRDB transaction processing, the HiRDB client environment definition must be specified in the transaction manager definitions. For details about how to specify the client environment definition in an OLTP environment, see the *HiRDB Version 9 UAP Development Guide*.

(a) OpenTP1

If the transaction manager is OpenTP1, the client environment definition must be specified in the `putenv` format in the following OpenTP1 system definitions:

- System environment definition
- User service default definition
- User service definition
- Transaction service definition

For details about these definitions, see the manual *OpenTP1 System Definition*.

To connect to multiple OpenTP1s, be sure to specify the following client environment definition:

- `HiRDB_PDTMID` or `PDTMID`

(b) TPBroker for C++

Specify the client environment definition in TPBroker for C++'s system definition.

(c) TUXEDO

Specify the client environment definition in the file that was specified with the `ENVFILE` parameter in the TUXEDO configuration file (`UBBCONFIG` file). For details about the TUXEDO configuration file, see the *TUXEDO* documentation.

(d) WebLogic Server

The WebLogic Server process environment variables must contain the specifications for the client environment definition.

(e) TP1/EE

If the transaction manager is TP1/EE, you must specify the client environment definition in the following system definitions for the OpenTP1 that is in the TP1/EE execution environment:

- User service default definition
- User service definition

For details about these definitions, see the manual *OpenTP1 System Definition*.

If you connect to multiple TP1/EEs, make sure that the following client environment definition is specified:

- `PDTMID`

(7) JDBC drivers (limited to use with WebLogic Server)

When HiRDB is registered, the following JDBC driver package name and driver class name must be specified:

- Package name: `JP.co.Hitachi.soft.HiRDB.JDBC`
- Driver class name: `JdbhXADataSource`

7.2.7 Example of registering in the transaction manager

(1) OpenTP1

To register HiRDB in OpenTP1, use OpenTP1's `trnlncrm` command. The following shows examples of `trnlncrm` command specification:

(a) Dynamic registration

```
trnlncrm -a HiRDB_DB_SERVER -s pdtxa_switch -o /HiRDB/client/lib/libzclty.sl
```

Explanation:

- a: Specifies the RM name.
- s: Specifies the RM switch name (name of the XA switch structure). The RM switch name depends on the registration method (dynamic or static).
- o: Specifies the RM-related object name (shared library's file name).

(b) Static registration

```
trnlncrm -a HiRDB_DB_SERVER -s pdtxa_switch_y -o /HiRDB/client/lib/libzclty.sl
```

Explanation:

- a: Specifies the RM name.
- s: Specifies the RM switch name (name of the XA switch structure). The RM switch name depends on the registration method (dynamic or static).
- o: Specifies the RM-related object name (shared library's file name).

(2) TPBroker for C++

To register HiRDB in TPBroker for C++, use TPBroker for C++'s `tslnkrm` command. The following shows examples of `tslnkrm` command specification:

(a) Dynamic registration

```
tslnkrm -a HiRDB_DB_SERVER_1 -s pdtxa_switch -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
tslnkrm -a HiRDB_DB_SERVER_2 -s pdtxa_switch -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
```

Explanation:

- a: Specifies the RM name.
- s: Specifies the RM switch name (name of the XA switch structure). The RM switch name depends on the registration method (dynamic or static).
- o: Specifies the RM-related object name (shared library's file name).
- r: Indicates dynamic registration.
- m: Enables OTS daemon operation with multi-thread.

(b) Static registration

```
tslnkrm -a HiRDB_DB_SERVER_1 -s pdtxa_switch_y -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
tslnkrm -a HiRDB_DB_SERVER_2 -s pdtxa_switch_y -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
```

Explanation:

- a: Specifies the RM name.
- s: Specifies the RM switch name (name of the XA switch structure). The RM switch name depends on the registration method (dynamic or static).
- o: Specifies the RM-related object name (shared library's file name).
- r: Indicates static registration.
- m: Enables OTS daemon operation with multi-thread.

(3) TUXEDO

Use the `$TUXDIR/udataobj/RM` file to register HiRDB in TUXEDO. `$TUXDIR` indicates the absolute path name of the directory that contains the TUXEDO system software. The following shows examples of RM file specification:

(a) Dynamic registration

```
HiRDB_DB_SERVER:pdtxa_switch:-L/HiRDB/client/lib -lzcltys
```

(b) Static registration

```
HiRDB_DB_SERVER:pdtxa_switch_y:-L/HiRDB/client/lib -lzcltys
```

(4) WebLogic Server

Register HiRDB using the driver class name and the provider for the WebLogic Server JDBC connection pool. An example specification follows:

(a) Static registration

```
Driver-class-name:JP.co.Hitachi.soft.HiRDB.JDBC.JdbhXADatasource
Property(key=value):user=authorization-identifier
                    password=password
```

```
XAOpenString=name-defined-by-transaction-data-source
dataSourceName=name-specified-by-JDBC-connection-pool
Description=name-defined-by-transaction-data-source
```

Explanation

When HiRDB is registered, the following JDBC driver package name and driver class must be specified:

- *Package-name*: JP.co.Hitachi.soft.HiRDB.JDBC
- *Driver-class-name*: JdbhXADataSource

(5) TP1/EE

To register HiRDB in TP1/EE, use TP1/EE's `eetrnmkobj` command. The following shows examples of `eetrnmkobj` command specification:

(a) Dynamic registration

```
eetrnmkobj -r HiRDB DB SERVER -o seigyo -s pdtxa_switch \
-O /HiRDB/client/lib/libzcltyk.sl -i /HiRDB/include
```

Explanation:

- r: Specifies the RM name.
- o: Specifies the name of the resource manager linkage object.
- s: Specifies the RM switch name (name of the XA switch structure). The RM switch name depends on the registration method (dynamic or static).
- O: Specifies the RM-related object name (shared library's file name).
- i: Specifies the HiRDB-provided header path.

(b) Static registration

```
eetrnmkobj -r HiRDB DB SERVER -o seigyo -s pdtxa_switch_y \
-O /HiRDB/client/lib/libzcltyk.sl -i /HiRDB/include
```

Explanation:

- r: Specifies the RM name.
- o: Specifies the name of the resource manager linkage object.
- s: Specifies the RM switch name (name of the XA switch structure). The RM switch name depends on the registration method (dynamic or static).
- O: Specifies the RM-related object name (shared library's file name).
- i: Specifies the HiRDB-provided header path.

7.2.8 Modifying the registration information in the transaction manager

To modify registration information in the transaction manager (from static registration to dynamic registration or vice versa), or to change a library to be specified for an RM-related object name, use the following procedure to reregister HiRDB in the transaction manager. Because a WebLogic Server can use static registration only, the registration cannot be modified.

(1) OpenTP1

To modify registration information in the transaction manager when using OpenTP1:

1. Use OpenTP1's `trnlncrm` command to reregister HiRDB in the transaction manager.
2. Use OpenTP1's `trnmkobj` command to re-create the object file for transaction control.

3. Relink all UAPs with HiRDB's XA library on the basis of the object file for transaction control re-created in step 2 and the information described in *7.2.6 Information to be registered in the transaction manager*. Otherwise, UAP operation cannot be guaranteed.

(2) TPBroker for C++

To modify registration information in the transaction manager when using TPBroker for C++:

1. Use TPBroker for C++'s `tslnkrm` command to reregister HiRDB in the transaction manager.
2. Use TPBroker for C++'s `tsmkobj` command to re-create the object file for transaction control.
3. Relink all UAPs with HiRDB's XA library on the basis of the object file for transaction control re-created in step 2 and the information described in *7.2.6 Information to be registered in the transaction manager*. Otherwise, UAP operation cannot be guaranteed.

(3) TUXEDO

To modify registration information in the transaction manager when using TUXEDO:

1. Use `$TUXDIR/udataobj/RM` to reregister HiRDB in the transaction manager.
2. Use TUXEDO's `builtdtms` command to re-create the transaction manager server's load module on the basis of the information described in *7.2.6 Information to be registered in the transaction manager*.
3. Use TUXEDO's `buildserver` command to re-create the server's load module on the basis of the information described in *7.2.6 Information to be registered in the transaction manager*.
4. Use TUXEDO's `buildclient` command to re-create the client module on the basis of the information described in *7.2.6 Information to be registered in the transaction manager*.

(4) TP1/EE

To modify registration information in the transaction manager when using TP1/EE:

1. Use TP1/EE's `eetrmkobj` command to reregister HiRDB in the transaction manager.
2. Use TP1/EE's `eetrmkobj` command to re-create the resource manager linkage object file.
3. Relink all UAPs with HiRDB's XA library on the basis of the resource manager linkage object file re-created in step 2 and the information described in *7.2.6 Information to be registered in the transaction manager*. Otherwise, UAP operation cannot be guaranteed.

7.2.9 Methods for re-establishing connection between the transaction manager and HiRDB

(1) Using an application program

If the connection is broken, terminate the running UAP and then restart it. Connection should be re-established automatically.

If restarting the application program is not desired, `tx_open` must be reissued when the error indicating the broken connection is returned to the application program. The service can be resumed without having to terminate the application program. The following is an example of coding for reissuing `tx_open`.

Example

```
int connection = 1;
void service(char *in_data, long *in_len, char *out_data, long *out_len) {
    if (connection == 0) {
        tx_close();
        tx_open();          .....Reissue tx_open when connection is cut
    }
    tx_begin();
    EXEC SQL INSERT INTO .....;          .....Issue SQL statement
    if (SQLCODE == 0) {
```

```

        tx_commit();
        *out_data = "OK";
    } else {
        tx_rollback();
        *out_data = "NG";
        if (SQLCODE == -563 || SQLCODE == -722) {
            connection = 0;
            .....Store broken connection
        }
    }
}

```

(2) When the OLTP product to be linked is TPBroker for C++ or WebLogic Server

If you are using an XA interface supporting multi-thread, connection with HiRDB is established when a transaction is started, and connection is cut off when the transaction is terminated. Therefore, even if the connection is broken during communication, it will be re-established when the next transaction is started.

(3) Using OpenTP1 facilities

If you used dynamic registration, specify `transaction` in the OpenTP1/Server Base operand `trn_rm_open_close_scope`. With this specification, OpenTP1/Server Base establishes or breaks the connection to HiRDB when the transaction starts or terminates. Therefore, even if the connection is broken during communication, it will be re-established when the next transaction is started.

If you used static registration, the transaction manager checks the connection with HiRDB when a transaction is started. If the connection has been broken, it is re-established automatically, and the transaction is started. For TP1/EE, YES must be specified for the HiRDB client environment definition `PDXAAUTORECONNECT`.

(4) Re-establishing connection with the client library that supports HiRDB's XA interface

If connection with HiRDB is broken before the first SQL statement for accessing HiRDB is executed since the transaction manager started the transaction, the HiRDB client library re-establishes connection when the SQL statement is executed. However, the connection will not be re-established if the connection is broken as a result of a network error, since these cannot be detected.

7.2.10 Monitoring HiRDB using the TP1/Resource Manager Monitor facility

This section discusses important points about using the TP1/Resource Manager Monitor (RMM) with HiRDB when OpenTP1 is linked as OLTP. For details about how to operate RMM, see the manual *OpenTP1 System Operation*.

(1) Notes on creating a command for obtaining process IDs subject to monitoring

When a command for obtaining the process IDs of the processes subject to monitoring is created, the processes to be monitored must be specified. For HiRDB, `_scd` must be specified as the process to be monitored. Following is an example of the command for HiRDB (shell script) provided by the RMM service.

```

#Watched Processes
PROCESSES="_scd"

```

To improve this command's reliability, the coding shown below should be added. When this is done, indeterminate process IDs will not be obtained before completion of HiRDB startup. This coding should be inserted before the line where process ID is obtained in the shell script (before the comment line `#These Lines Are The Description Of Get Process_ID Process`).

```

#System status check
get_STATUS='$PDDIR/bin/pdls 2>/dev/null | \
    /usr/bin/awk'{print$4}' | /bin/grep -v STATUS'
for i in $(get_STATUS)
do
    if[${i}!="ACTIVE"]
    then

```

```

        exit 2
    fi
done

```

(2) Notes on using RMM operations with a HiRDB parallel server configuration

In the case of a HiRDB parallel server configuration, a series of commands should be specified for the resource manager subject to monitoring in the definition of the RM subject to monitoring only for the OpenTP1 located on the same host as the unit where the system manager is located.

7.2.11 Notes

(1) Notes about SQL

1. The transaction manager has permission to establish connection with or disconnect from the resource manager. Do not specify an SQL statement in a UAP that establishes connection with or disconnects from the resource manager. The transaction manager also has permission to adjust and monitor the progress of transactions. Therefore, do not specify an SQL statement in a UAP that rolls back or commits a transaction. This means that statements such as `EXEC SQL COMMIT WORK` and `EXEC COMMIT WORK RELEASE` will result in an error.
2. A definition SQL statement will result in an error. A definition SQL statement such as `CREATE TABLE` automatically instructs a commit; therefore, do not specify a definition SQL statement in a UAP.

(2) Notes about libraries for multi-thread

A single transaction cannot establish connection with multiple HiRDB servers separately by using multi-thread. There is only one server process for a HiRDB server connecting from a transaction, even in multithreaded environments. This means that a single transaction can execute only one thread at any one time. Executing more than one SQL statement at one time using multiple threads within the same transaction is not permitted.

(3) Notes on single-phase optimization

HiRDB applies single-phase optimization supported by the transaction manager. The transaction manager can request single-phase commit for a transaction branch if HiRDB is the only shared resource in the transaction branch that was modified. When the transaction manager uses single-phase optimization to request a single-phase commit, once the result of the transaction branch has been decided, HiRDB will delete the information from the transaction branch and return a response to the transaction manager.

With transaction managers that use single-phase optimization, it is not necessary to remember stable storage for global transactions, and even if a failure occurs it is not necessary to know of that result. Therefore, if all of the following conditions are satisfied, the transaction completion status might not match between the transaction manager and HiRDB:

- The connection uses a transaction manager that employs single-phase optimization and XA interface
- The transaction manager uses single-phase optimization for commitment control of the modified system's transaction
- The transaction manager's UAP terminates abnormally during commitment processing

Under these conditions, the result of the HiRDB transaction branch cannot be determined from the result of the failure that was generated by the transaction manager. Therefore, the transaction completion type might not match between the transaction manager and HiRDB.

To avoid this, when exercising commitment control on the transaction of a modified system, do not have the transaction manager use single-phase commit.

(4) Considerations when WebLogic Server is used

- Commit or rollback deletes the preprocessing and the cursor. Re-execute from the preprocessing.

- If commit or rollback is performed on a `ResultSet` object with the cursor still open, the HiRDB transaction might remain uncompleted. If a `ResultSet` object is used, you must execute the `ResultSet.close()` method to close the cursor before a commit or rollback.

(5) Considerations when the rapid system switchover facility is used

Caution must be exercised when all of the following conditions are satisfied:

- A HiRDB/Parallel Server is using the rapid switchover facility with the unit where the system manager is installed
- Linkage is with an OLTP product that uses an API (such as OpenTP1 or TPBroker for C++) that complies with X/Open
- The HiRDB client version is 06-02-/A or earlier
- The primary system specified in the OLTP product's `PDHOST` client environment variable is a standby system in wait completion status

In such a case, when an OLTP product performs uncompleted transaction recovery processing, it is possible that the X/Open-compliant API will return an error and the transaction will not be recovered. When this occurs, upgrade the HiRDB client to version 06-02-/B or later. If the HiRDB client cannot be upgraded soon, for some reason such as you do not want to stop a running application, perform a system switchover of the primary HiRDB system (unit) from a standby system to a running system. However, this is a temporary measure; you should upgrade the HiRDB client version as soon as possible.

7.3 Linking to the inner replica facility

The inner replica facility enables construction of a database system that is capable of providing uninterrupted service. In order to use the inner replica facility, HiRDB Staticizer Option is required.

(1) HiRDB system definition specifications

The following operands are specified in the HiRDB system definition:

- `pd_inner_replica_control` operand
Specifies the maximum number of inner replica groups.
- `pd_inner_replica_lock_shift` operand
Specifies whether UAPs and commands are to be executed concurrently when the inner replica facility is used.
- `pd_lv_mirror_use` operand
Specifies whether the open attribute for a replica RDAREA is to be `SCHEDULE`.

When the inner replica facility is used together with updatable online reorganization, the following operands are specified:

- `pd_max_reflect_process_count` operand
Specifies the number of `pdorend` reflection processes secured in reflection processing. If this operand is omitted, updatable online reorganization cannot be performed.
- `pd_log_org_reflected_logpoint` operand
Specifies whether the status of system log files that have completed reflection processing on all update logs is to be changed.
- `pd_log_org_no_standby_file_opr` operand
When all system log files are in overwrite permitted status for online reorganization, specifies the HiRDB processing when system log files have been swapped.

In addition to the operands above, you should also consider the specifications of the following operands:

- `pd_max_rdarea_no`
- `pd_max_file_no`
- `pd_assurance_index_no`

(2) Environment definition specifications

For details about the system environment definitions in conjunction with using the inner replica facility, see the *HiRDB Version 9 Staticizer Option Description and User's Guide*.

7.4 Linking to JP1

When linked with JP1, HiRDB can centrally manage all events in the entire system, including HiRDB, and start jobs automatically using events as job start triggers. HiRDB can link to the following JP1 products:

- JP1/Base (JP1/System Event Service for JP1 Version 6 or earlier)
- JP1/Integrated Management - Manager (JP1/Integrated Manager for JP1 Version 7 or earlier)
- JP1/Automatic Job Management System 3 (for JP1 Version 8 or earlier, JP1/Automatic Job Management System 2; for JP1 Version 5 or earlier, JP1/Automatic Operation Monitor)

In this manual, JP1/Integrated Management - Manager is called *JP1/IM* and JP1/Automatic Job Management System 3 is called *JP1/AJS3*.

64-bit mode Solaris, HiRDB, AIX HiRDB and Linux (EM64T) HiRDB cannot be linked with JP1.

7.4.1 Reporting events to JP1/Base

Events, such as the start and termination of HiRDB, can be reported to the JP1/Base (JP1/System Event Service for JP1 Version 6 or earlier) that manages JP1 events. JP1/Base manages the reported HiRDB events as JP1 events. This makes it possible to use JP1/IM to manage events and to execute jobs automatically using JP1/AJS3 linkage. For details about event monitoring using JP1/IM, see *7.4.2 Managing events by JP1/IM*. For details about automatic job execution using JP1/AJS3 linkage, see *7.4.3 Automatic job execution using JP1/AJS3 linkage*.

For details about JP1/Base, see the JP1 manual. Because the manual name depends on the JP1 version, see the correct manual for the version you are using:

- For JP1 Version 8 or later
Job Management Partner 1/Base User's Guide
- For JP1 Version 7i or earlier
Job Management Partner 1/Base
- For JP1 Version 5 or earlier
JP1/System Event Service

(1) Sending event notice

To send HiRDB events to JP1/Base, specify the following operands:

- `pd_jp1_use` operand: Y
- `pd_jp1_event_level` operand: 1 or 2

When 1 is specified in the `pd_jp1_event_level` operand, only the basic attributes are sent. When 2 is specified in this operand, extended attributes are also sent.

Important note

In the following cases, only 1 can be specified in the `pd_jp1_event_level` operand; if 2 is specified, events cannot be sent:

- Events are to be sent to JP1/System Event Service.
 - HiRDB in use is a 64-bit mode version for HP-UX.
-

(2) HiRDB events that can be sent

The following table shows HiRDB events that can be sent to JP1/Base.

Table 7–6: HiRDB events that can be sent to JP1/Base

Event	Event ID#1	Message	Detail information #2	Registration timing	Value of pd_jp1_event_level		Extended attribute class
					1	2	
Start	0x00010C00	KFPS05210-I, KFPS05219-I#3, or KFPS05260-I#5	"start"	After the system start completion message (KFPS05210-I, KFPS05219-I, or KFPS05260-I) is output	Y	Y	Information
	0x00010C80						
Normal termination	0x00010C01	KFPS01850-I#3	"end_normal"	After HiRDB system shutdown (immediately before completion of HiRDB termination or immediately after output of the KFPS01850-I message)	Y	Y	Information
	0x00010C81						
Planned termination	0x00010C01	KFPS01850-I#3	"end_plan"		Y	Y	Information
	0x00010C81						
Forced termination	0x00010C01	KFPS01850-I#3	"end_force"		Y	Y	Information
	0x00010C81						
Message log output	0x00010C03	HiRDB message	NULL	After message log is output	Y#4	Y#4	Information
	0x00010C83						
Abnormal termination	0x00010C02	KFPS01821-E	NULL	<ul style="list-style-type: none">Upon HiRDB error terminationAfter error termination is reported to the cluster software when system switchover is used in server mode	N	Y	Error
	0x00010C82						
Definition change	0x00010C04	KFPS04666-I	NULL	At completion of HiRDB system normal start	N	Y	Notice
	0x00010C84						
RDAREA full	0x00010C05	KFPH00213-W	NULL	At output of RDAREA-full error message (KFPH00213-W)	N	Y	Warning
	0x00010C85						
RDAREA expansion error	0x00010C06	KFPX14229-E	NULL	Upon pdmod (with expand specified) execution error	N	Y	Error
	0x00010C86						
Log file free space warning	0x00010C07	KFPS01160-E or KFPS01162-W	NULL	At output of a system log file free-space warning message	N	Y	Warning
	0x00010C87						

Legend:

NULL: Null; nothing is sent.

Y: This event is sent.

N: This event is not sent.

#1: The top code is for a HiRDB single server configuration; the bottom code is for a HiRDB parallel server configuration.

#2: The detail information is provided in text format. Following is the data format:

ssss mm...mm \0

ssss: System identifier (0-4 bytes)

mm...mm: Character string indicated in the *Detail information* column

#3: When `pd_jp1_event_level=1` is specified, the message is null.

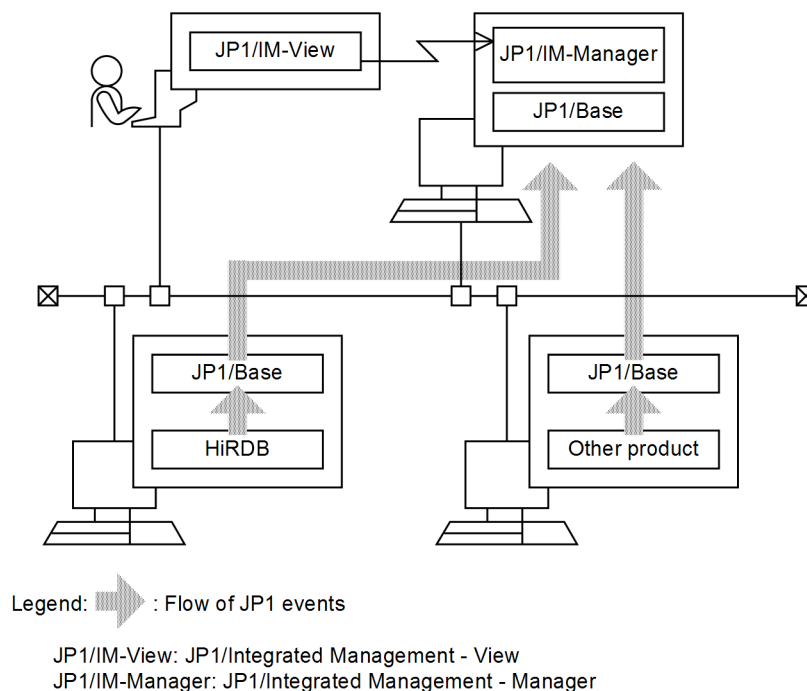
#4: When `pd_jp1_event_msg_out=N` is specified, the event is not sent.

#5: This message is displayed only when a log applicable site is started using the `pdstart -1` command in cases in which the log-only synchronous method is used for Real Time SAN Replication processing.

7.4.2 Managing events by JP1/IM

JP1/IM optimizes (filters) the JP1 events managed by JP1/Base and centrally manages the events that occur in the system as JP1 events. By sending HiRDB events to JP1/Base, you can have JP1/IM manage them in the same manner as with other products' events. The user can check events using windows provided by JP1/IM. The following figure gives an overview of event monitoring by JP1/IM.

Figure 7-7: Overview of event monitoring by JP1/IM



For an overview of event monitoring by JP1/Integrated Management, see the applicable manual for the JP1 version being used.

- For JP1 Version 9
Job Management Partner 1/Integrated Management - Manager Overview and System Design Guide
- For JP1 Version 8
Job Management Partner 1/Integrated Management - Manager System Configuration and User's Guide
- For JP1 Version 7i
Job Management Partner 1/Integrated Manager - Console

(1) Preparations for displaying HiRDB-specific extended attributes by JP1/IM

For JP1/IM to display HiRDB-specific extended attributes, you must copy the *event attribute definition file* provided by HiRDB to the following directory:

- For JP1/Integrated Management - Manager (JP1 Version 7i or earlier JP1/Integrated Manager - Central Console):
installation-directory/conf/console/attribute

The event attribute definition file is stored in the `sample` directory under the HiRDB installation directory. The following table shows the names of event attribute definition files for each OS.

Table 7–7: Name of the event attribute definition file for each OS

OS	Name of the event attribute definition file
HP-UX	HITACHI_HIRDB_HP_attr_ja.conf
AIX	HITACHI_HIRDB_AIX_attr_ja.conf
Solaris	HITACHI_HIRDB_SOL_attr_ja.conf
Linux	HITACHI_HIRDB_LIN_attr_ja.conf

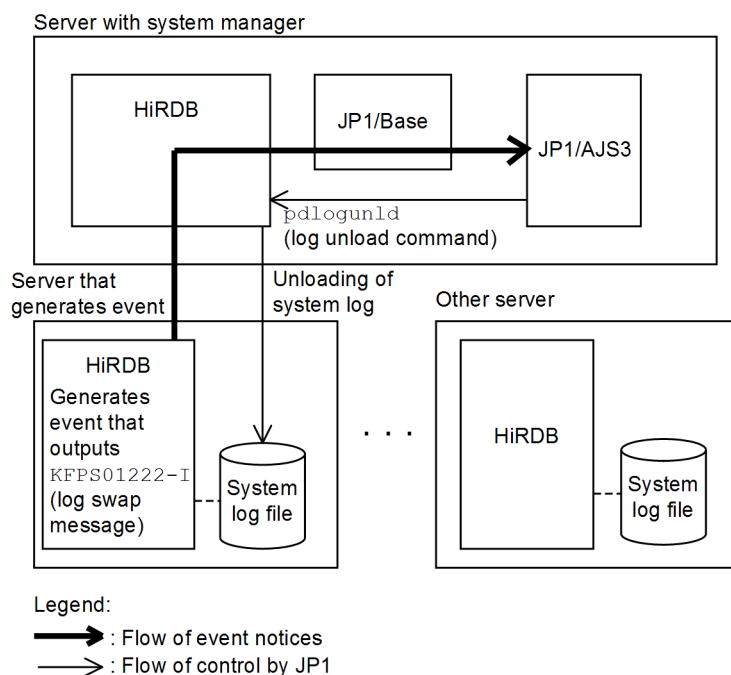
For a multi-HiRDB, copy the most recent version of the event attribute definition file in the running HiRDBs.

7.4.3 Automatic job execution using JP1/AJS3 linkage

For a HiRDB parallel server configuration, operations such as unloading system log files at each server may become complex. In this case, when a HiRDB event is sent to JP1/Base, jobs can be executed automatically by JP1/AJS3 (for JP1 Version 8 or earlier, JP1/AJS2; for JP1 Version 5 or earlier, JP1/Automatic Operation Monitor) using the sent event as the trigger, allowing HiRDB operations to be automated.

The following figure shows the automatic control achieved by JP1 linkage when system log files are unloaded.

Figure 7–8: Automatic control achieved by JP1 linkage when system log files are unloaded



For details about JP1/AJS3, see the JP1 manuals.

8

Designing a HiRDB single server configuration

This chapter describes the design considerations for a HiRDB single server configuration, its HiRDB file system areas, and its system files, and provides notes on the placement of RDAREAs.

8.1 System design for a HiRDB single server configuration

This section describes the system design considerations and the system configuration for a HiRDB single server configuration.

8.1.1 System design

(1) Memory used by a HiRDB single server configuration

This subsection describes the memory used by a HiRDB single server configuration.

A HiRDB single server configuration uses the following memory.

- Shared memory
- Process private memory

(a) Storage requirements

The storage space required by a HiRDB single server configuration must be estimated. For details about the storage requirements for a HiRDB single server configuration, see *15.1 Estimating the memory size required for a HiRDB single server configuration*.

(b) Page fixing of shared memory

With HiRDB, the following shared memory can be fixed in actual memory.

- Shared memory for unit controllers
- Shared memory for global buffers
- Shared memory used by dynamically changed global buffers
- Shared memory for in-memory data buffers

Fixing shared memory in actual memory reduces the number of page I/Os, stabilizing performance.

Prerequisites

The following table shows the prerequisites for page fixing of shared memory by OS.

OS	Prerequisites
HP-UX	None
Solaris	None
AIX	Must be 64-bit mode
Linux	None

Operating environment settings

AIX requires you to set operating system parameters. For details, see *20.3(1) Specifying parameters unique to AIX*.

Page fixing methods

This subsection describes shared memory page fixing methods for each type of shared memory.

- Shared memory for unit controllers
Specify `fixed` in the `pd_shmpool_attribute` operand of the system common definition or unit control information definition.
- Shared memory for global buffers

Specify *fixed* in the `pd_dbbuff_attribute` operand of the system common definition or unit control information definition.

- Shared memory used by dynamically changed global buffers

Specify *fixed* in the `pd_dbbuff_attribute` operand of the system common definition or unit control information definition. This fixes shared memory used by global buffers dynamically changed by the `pdbufmod` command in actual memory.

- Shared memory for in-memory data buffers

Specify *fixed* in the `pdmemdb` command `-p` option.

! Important note

When contiguous areas cannot be secured in actual memory, shared memory pages cannot be fixed. HiRDB operation when page fixing fails is shown below.

OS	HiRDB operation when page fixing fails			
	Unit controller shared memory	Global buffer shared memory	Dynamically changed global buffer shared memory	In-memory data buffer shared memory
HP-UX	Y	N	N	N
Solaris	Y	N	N	N
AIX	Y [#]	Y [#]	Y [#]	Y [#]
Linux	Y	N	N	N

Legend:

Y: Shared memory is secured without fixing pages, and processing continues.

N: HiRDB or the command terminates abnormally.

#

In AIX, system calls terminate normally even when page fixing fails. This means that you cannot tell from HiRDB that page fixing failed. Use the following procedure to check whether pages were fixed.

1. While HiRDB is running, execute the `pdls -d mem` command to check the identifier of the following shared memory segment:

- For shared memory for unit controllers, the shared memory segment with `MANAGER` displayed under `SHM-OWNER`.

- For other types of shared memory, the shared memory segment with a character string consisting of the unit name in parentheses or the HiRDB server name displayed under `SHM-OWNER`.

2. Execute the OS's `ipcs -s` command, and then check the `SID` value of the shared memory that has the identifier of the shared memory segment you checked in step 1.

3. Execute the OS's `svmon` command on the `SID` value you checked in step 2, and then check whether the number of actual memory pages of the shared memory in question matches the number of fixed pages.

(2) Installing a utility special unit

It may not be possible to install a desired input/output device (such as MT drives) to be used by utilities at the HiRDB single server configuration machine because of I/O device restrictions. In such a case, the I/O device can be installed on a separate server machine and can be used by the utilities via a LAN. A server machine on which only input/output devices are installed for use by utilities is called a *utility special unit*.

A utility special unit is used by the following utilities:

- Database load utility (`pdload`)
- Database reorganization utility (`pdrorg`)
- Dictionary import/export utility (`pdexp`)
- Database copy utility (`pdcopy`)
- Database recovery utility (`pdrstr`)

For details about how to use a utility special unit with execution of utilities, see the manual *HiRDB Version 9 Command Reference*.

When an input/output device cannot be installed at the HiRDB single server configuration machine, installation of a utility special unit should be considered.

One utility special unit can be shared among multiple HiRDB single server configurations.

8.1.2 System configuration

The following figure shows an example of a system configuration for a HiRDB single server configuration. Figure 8-2 shows system configurations with a utility special unit installed.

The HiRDB single server configuration is defined in the HiRDB system definition. For details about definition examples of the system configurations, see the manual *HiRDB Version 9 System Definition*.

Figure 8–1: System configuration for a HiRDB single server configuration

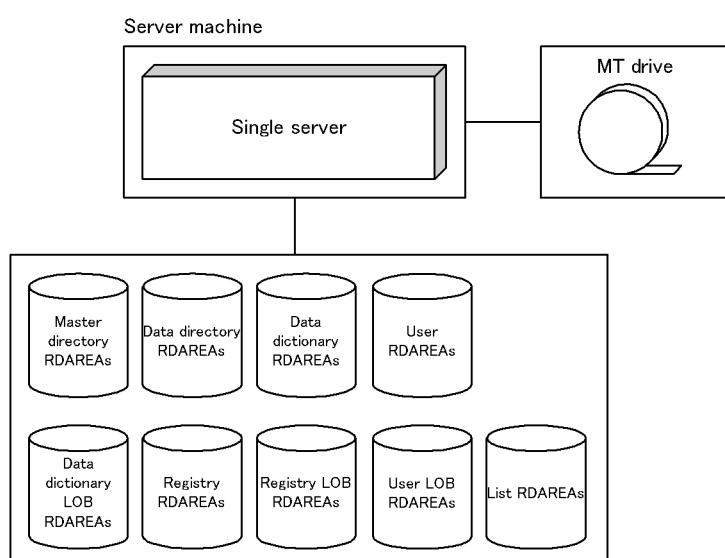
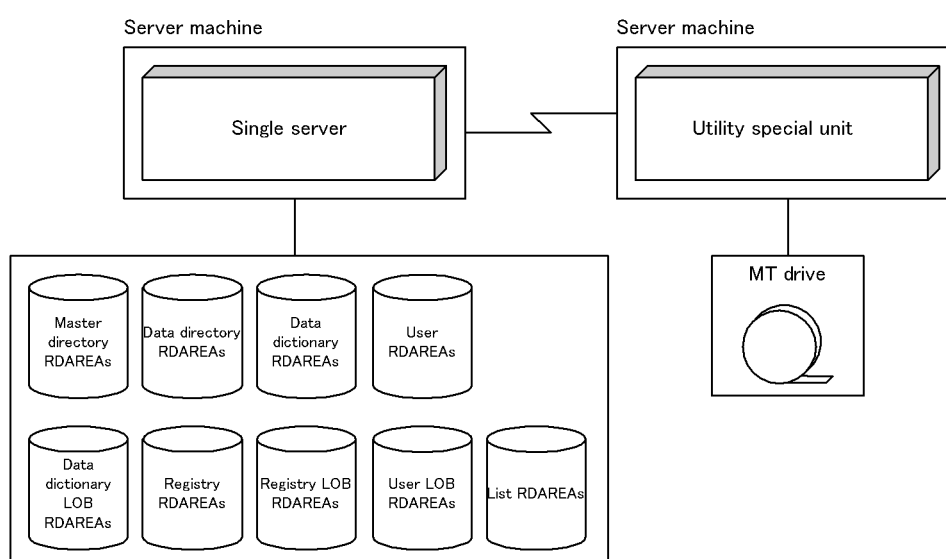


Figure 8–2: System configuration for a HiRDB single server configuration with a utility special unit installed



8.2 Designing HiRDB file system areas

When constructing a HiRDB system, you must create HiRDB file system areas where HiRDB files are created. This section discusses the design considerations for creating HiRDB file system areas.

Separate HiRDB file system areas should be created for the types of items listed below, so that contention between input/output operations on files with different purposes or access characteristics can be avoided. If regular files are used, write performance can be improved by specifying the purpose explicitly, and an appropriate device can be allocated according to the purpose. Separate file system areas should be created for:

- RDAREAs
- System files
- Work table files
- Utilities
- RDAREAs for lists (list RDAREAs)

8.2.1 Designing HiRDB file system areas for RDAREAs

This section discusses the design considerations for HiRDB file system areas in which RDAREAs are to be created.

(1) Design for improved reliability

1. For update processing, character special files are more reliable than regular files. Regular files may become inaccessible if the OS terminates abnormally. Thus, character special files should be used for HiRDB file system areas for user RDAREAs that satisfy the following characteristics:
 - User RDAREAs for storing tables that are used primarily for update processing.
 - User RDAREAs for storing particularly important data.
2. Calculate the size of the required HiRDB file system area and create a HiRDB file system area of that size or larger.
3. HiRDB file system areas for RDAREAs must be created at the server machine where the single server is defined.
4. For the following RDAREAs, create HiRDB file system areas on the server machine where the single server is defined:
 - System RDAREAs
 - Data dictionary LOB RDAREAs
 - Registry RDAREAs
 - Registry LOB RDAREAs
5. For the following RDAREAs, create HiRDB file system areas on the server machine where the single server is defined:
 - User RDAREAs
 - User LOB RDAREAs
6. If the system switchover facility is to be used, the HiRDB file system areas for user RDAREAs should be allocated as character special files.

(2) Design for improved performance

1. You should create separate HiRDB file system areas for the following types of RDAREAs:
 - System RDAREAs
 - Data dictionary LOB RDAREAs
 - User RDAREAs
 - User LOB RDAREAs

- Registry RDAREAs
 - Registry LOB RDAREAs
2. You should create HiRDB file system areas for system files on hard disks separate from the ones used for HiRDB file system areas for RDAREAs. In this way, you can distribute input/output operations when collecting a synchronization point dump, thereby reducing the amount of time required to collect the synchronization point dump.
 3. If you are not using the prefetch facility, the sequential read speed is faster with regular files than with character special files.
 4. For random one-page reads, processing speed is faster with character special files than with regular files.
 5. For write processing, the speed is faster with character special files than with regular files.
 6. Regular files have a hierarchical structure; therefore, as files become larger, the hierarchical levels increase. When you access such files with many hierarchical levels, the number of input/output operations increases, adversely affecting the access efficiency.
 7. The HiRDB file system areas should be allocated as shown in the following table, so that input/output time can be reduced.

Table 8–1: Allocation of HiRDB file system areas to improve performance

Type of HiRDB file system areas	File to be allocated
HiRDB file system area for system RDAREAs	Character special file
HiRDB file system area for dictionary LOB RDAREAs	
HiRDB file system area for user LOB RDAREAs	
HiRDB file system area for user RDAREAs that store frequently updated tables or tables primarily subject to retrieval of small amounts of data	
HiRDB file system area for user RDAREAs that store tables with a large amount of data that are primarily subject to retrieval of all entries or retrieval of large amounts of data by key using the cluster key (but with data that is rarely updated)	<ul style="list-style-type: none"> • Regular file (when not using the prefetch facility) • Character special file (when using the prefetch facility)

8.2.2 Designing HiRDB file system areas for system files

This section discusses the design considerations for HiRDB file system areas in which system files are to be created.

(1) Design for improved reliability

1. For update processing, character special files are more reliable than regular files. Regular files may become inaccessible after a system shutdown. Thus, character special files should be used for HiRDB file system areas for system log files, synchronization point dump files, and status files.
2. Create at least two HiRDB file system areas for system files. If there is only one HiRDB file system area for system files, HiRDB cannot continue operating in the event of a hard disk failure at the disk containing the system files.
3. Create HiRDB file system areas for system files on separate hard disks. In this way, in the event of a hard disk error, you can restart HiRDB using the other hard disk.
4. Calculate the size of the required HiRDB file system area and create a HiRDB file system area of that size or larger.

(2) Design for improved performance

You should create HiRDB file system areas for system files on hard disks separate from the ones used for HiRDB file system areas for RDAREAs. In this way, you can distribute input/output operations when collecting a synchronization point dump, thereby reducing the amount of time required to collect the synchronization point dump.

8.2.3 Designing HiRDB file system areas for work table files

This section discusses the design considerations for HiRDB file system areas in which work table files are to be created.

(1) Design considerations

1. Work table files can be allocated to regular files. However, if the system switchover facility is used, disk space can be saved by allocating work table files to character special files because the files can then be shared.
2. The amount of space required for a HiRDB file system area for work table files must be greater than the total size of the work table files to be created in the area. If you specify the `-a` option with the `pdfmkfs` command, the HiRDB file system area can be automatically extended. We recommend specifying the `-a` option, since the HiRDB file system area will be extended automatically when the total size of the work table files reaches the size of the HiRDB file system area.[#]

For details about the sizes of work table files, see *Chapter 18. Determining Work Table File Size*.

#

To reduce the amount of disk space that the HiRDB file system area for work table files occupies when HiRDB is re-started, before restarting HiRDB, execute the `pdfmkfs` command and then re-initialize the HiRDB file system area for work table files.

(2) Checking the peak capacity

The peak capacity of a HiRDB file system area for work table files can be obtained by entering the following command:

```
pdfstatfs -d name-of-HiRDB-file-system-area-for-work-tables
```

`-d`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be displayed. The peak capacity display that is output is this value. The maximum utilization value is cleared by entering the following `pdfstatfs` command:

```
pdfstatfs -c name-of-HiRDB-file-system-area-for-work-table
```

`-c`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be cleared to 0.

8.2.4 Designing HiRDB file system areas for utilities

This section discusses the design considerations for HiRDB file system areas in which utility files (backup files, unload data files, and unload log files) are created. Use the HiRDB file system areas for utilities to create the following files:

- Backup files
- Unload data files
- Unload log files
- Differential backup management files

(1) Design considerations

1. If you are creating a HiRDB file system area for backup files, allocate it to character special files.
2. The amount of space required for a HiRDB file system area for backup files must be greater than the total size of the RDAREAs that will be backed up. For details about the sizes of RDAREAs, see *Chapter 16. Determining RDAREA Size*.
3. If you use the system switchover facility, create unload log files on a shared disk (character special files).

4. If you are creating a HiRDB file system area for unload log files, specify the following options in the `pdfmkfs` command:

- `-k` option: Specify `UTL` (HiRDB file system area for utilities) as the usage.
- `-n` option: For the size of the HiRDB file system area, specify the value obtained from the following formula:

$$(total\ record\ count\ for\ system\ log\ file\ to\ be\ unloaded)^{\#1} \times system\ log\ file\ record\ size)^{\#2} \times number\ of\ unload\ log\ files\ to\ be\ created \times 1.2 \div 1,048,576$$
- `-l` option: For the maximum number of files, specify the number of unload log files to be created.
- `-e` option: For the maximum number of extensions, specify the number of unload log files to be created times 24.

#1

If you are using the system log file automatic extension facility, calculate this number using the value specified as the upper limit for expansion in the `pd_log_auto_expand_size` operand.

#2

This is the approximate value of the system log file.

(2) Checking the peak capacity

You can obtain the peak capacity of a HiRDB file system area for utilities by entering the following command:

```
pdfstatfs -d name-of-HiRDB-file-system-area-for-utilities
```

`-d`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be displayed. The peak capacity display that is output is this value. You can clear the maximum utilization value by entering the following `pdfstatfs` command:

```
pdfstatfs -c name-of-HiRDB-file-system-area-for-utilities
```

`-c`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be cleared to 0.

8.2.5 Designing HiRDB file system areas for list RDAREAs

This section discusses the design considerations for HiRDB file system areas in which list RDAREAs are to be created.

(1) Design considerations

1. A list is used to store temporary intermediate results of search processing; therefore, it does not need to be as reliable as other RDAREAs. You can create HiRDB file system areas for list RDAREAs in regular files.
2. If you use the system switchover facility, you can save disk space by allocating HiRDB file system areas for list RDAREAs to character special files because the files can then be shared.

(2) Design for improved performance

1. If you are creating HiRDB file system areas for list RDAREAs on RAID, you should use character special files to reduce processing time. If you are creating them on any other disk, you should use regular files to reduce processing time.
2. You should create HiRDB file system areas for list RDAREAs on a hard disk separate from the ones used for the following HiRDB file system areas. In this way, you can distribute input/output operations when searching a list, thereby reducing the processing time.
 - HiRDB file system areas for user RDAREAs
 - HiRDB file system areas for user LOB RDAREAs
 - HiRDB file system areas for work table files

8.2.6 Maximum sizes of HiRDB file system areas

The following table lists the maximum sizes of the HiRDB file system areas.

Table 8–2: Maximum sizes of HiRDB file system areas

HiRDB type	Conditions		Maximum size of HiRDB file system area (MB)
HP-UX	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file	1,048,575
		Character special file	
Solaris	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file	1,048,575
		Character special file	
AIX	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file(JFS)	65,411
		Regular file(JFS2)	1,048,575
		Character special file	
Linux	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file	1,048,575
		Character special file	

8.3 Designing system files

This section describes the design considerations for system files.

8.3.1 Designing system log files

The following explains some of the design considerations for system log files.

(1) Design considerations

1. System log files are not required for a utility special unit.
2. Specify the same record length and number of records for all system log files.
3. The number of system log files that can be created is 2 to 200 groups (we recommend creating at least six groups).
4. For one server, creation of system log files must meet the following condition:

$$100 \text{ (unit: megabytes)} \times (200 - \text{number of system log file groups assigned to server}) \geq \text{total system log file capacity allocated to the server} \times 3$$

Multiplying by three is necessary to restart a HiRDB system that terminated abnormally due to insufficient space for the system log files. Depending on usage and whether there is enough space for system log files, you might need to take action by creating system log files of three times the capacity allocated to the server, and then adding them to the server.

5. To reduce the number of unload operations, it is advisable to create many large system log files.
6. A file that is involved in many input/output operations (such as a log unload file) should not be created on the same disk that contains HiRDB directory.
7. The amount of space required for one system log file must satisfy the condition shown following.

$$\text{Size of one system log file (bytes)} \geq \uparrow(a + 368) \div c \uparrow \times c \times b \times d$$

- a*: Value of `pd_log_max_data_size` operand
b: Value of `pd_log_sdinterval` operand
c: Value of `pd_log_rec_leng` operand
d: Value of `pd_spd_assurance_count` operand

8. The total size of the system log files (if duplexed, only their size in one of the systems) must meet the following two conditions.

Condition 1

The size must be at least the value calculated according to the formula in 17.1.1(1) *How to obtain the total size of system log files*.

Condition 2

After the start of the large object transaction, system log files cannot be overwritten until the large object transaction finishes. Moreover, the current generation of system log files and system log files corresponding to the number of guaranteed-valid generations of the synchronization point dump file cannot be overwritten either. For this reason, make sure that there is at least enough system log file capacity to accommodate this amount of data. Use the following equation to estimate the required capacity.

$$\text{System log file total size (bytes)} \geq 3 \times a \times (b + 1)$$

a:

Size of system log information that may be output at the corresponding server while executing the database updating transaction with the longest execution time.

For details about the formula for estimating the size of system log information, see 17.1 *Determining the size of system log files*.

b: Value of `pd_spd_assurance_count` operand

Number of guaranteed-valid generations for synchronization point dump files.

(a) Effects on operations of the number of generations of system log files

If the total size of the system log files is unchanged, the size of each generation will depend on how many generations of system log files are being maintained. The following table describes the effect on operations of the number of generations of system log files. The total size of the system log files is unchanged.

Table 8–3: Effects on operations of the number of generations of system log files

Comparison item	System log file configuration	
	Small number of generations	Large number of generations
Size of each generation of system log files	Becomes larger.	Becomes smaller.
Swap interval	Because the size of each generation of system log files becomes larger, the swap interval becomes longer.	Because the size of each generation of system log files becomes smaller, the swap interval becomes shorter.
Unload frequency	Because the swap interval becomes longer, the unload frequency becomes lower.	Because the swap interval becomes shorter, the unload frequency becomes higher.
Effects on the system log size when something such as a disk failure makes several generations of system log files unusable	<ul style="list-style-type: none"> Because the size of each generation of system log files becomes larger, the log volume used for database recovery in the event of a disk failure increases, and the time required for database recovery increases. If the decrease in the system log volume is large, the effects of the decrease in system log volume will have increasing effects on HiRDB operations. 	<ul style="list-style-type: none"> Because the size of each generation of system log files becomes smaller, the log volume used for database recovery in the event of a disk failure decreases, and the time required for database recovery decreases. If the decrease in the system log volume is small, the effects of the decrease in system log volume will have decreasing effects on HiRDB operations.

In normal operations, the lower the number of generations of system log files, the more advantageous the swapping interval and the unload frequency will become. However, if there is a failure, the effects on operations will be reduced with a larger number of log file generations.

(2) Design for improved reliability

(a) System log file duplexing

When system log file duplexing is used, HiRDB acquires the same system log information in both versions. In the event of an error on one of the versions, the system log can be read from the other version, thereby improving system reliability. When dual system log files are used, they must be used under the management of HiRDB rather than using a mirror disk. When using dual system log files, create the files for each system on a separate hard disk.

To use dual system log files, specify the following operands in the server definition:

- `pd_log_dual=Y`
- `-b` option in the `pdlogadpf` operand (to specify the name of the B version of system log file)

(b) Single operation of system log files

Single operation of system log files is employed when dual system log files are used.

In the event of an error in a system log file, processing can continue using the normal version of the system log file without having to terminate the HiRDB unit abnormally even if neither system has a usable system log file. This is called single operation of system log files. To perform single operation of system log files, specify `pd_log_singleoperation=Y` in the server definition.

As opposed to single operation of system log files, continuing processing using both versions of system log files (normal processing mode) is called double operation of system log files.

(c) Automatic opening of system log files

If there is no overwrite-enabled system log file at the time of a HiRDB restart, but a reserved file is available, then HiRDB continues processing by opening the reserved file and placing it in overwrite-enabled status. This is called automatic opening of system log files.

To perform automatic opening of system log files, specify `pd_log_rerun_reserved_file_open=Y` in the server definition.

(3) Features to improve availability

(a) Facility for monitoring the free space remaining for system log files

When system log files need to be swapped, if no swappable system log files exist, HiRDB terminates abnormally. To prevent this, HiRDB has a *facility for monitoring the free space remaining for system log files*. This facility operates when the percentage of available free area for the system log files reaches a warning value. Select one of the following two levels.

Level 1:

When the available space in the system log file falls below the warning level, the `KFPS01162-W` warning message is output.

Level 2:

When the available space in the system log file falls below the warning level, scheduling of new transactions is suppressed, and all transactions inside the server are forcibly terminated. At this time, message `KFPS01160-E` is output. This controls the output volume of the system logs.

If level 2 is selected, all transactions on the server are terminated forcibly when there is insufficient free space in the system log files. Therefore, the design of system log files requires more accuracy.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

(b) System log file automatic extension facility

The HiRDB system (or unit) terminates abnormally when it runs out of free space for system log files. To prevent this, HiRDB provides a facility to automatically expand the space for system log files (the system log file automatic extension facility). By using this facility, you can reduce the frequency of abnormal termination of the HiRDB system (or unit) due to lack of free space for system log files.

For details about the system log file automatic extension facility, see the *HiRDB Version 9 System Operation Guide*.

(c) Skipped effective synchronization point dump monitoring facility

If a UAP continues to update the database in an infinite loop, it cannot enable the synchronization point, and the number of system log files that cannot be overwritten increases. If the point is reached where none of the system log files can be overwritten, HiRDB terminates abnormally.

If HiRDB is forcibly terminated or terminates abnormally when the number of system log files that cannot be overwritten reaches one-half or more of all system log files, a shortage of system log files occurs during rollback processing when HiRDB restarts. In this case, HiRDB cannot be restarted unless new system log files are added. Any such restart processing will take longer than usual.

To prevent this, HiRDB has established a skipped effective synchronization point dump monitoring facility.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

(4) Facility for parallel output of system logs (AIX and Linux versions only)

When the system log file is duplexed, using the facility for parallel output of system logs allows output processing of logs to the two systems to be executed in parallel. As a result, less time is required for log output. To use the facility for parallel output of system logs, you need an aio library (for AIX, the Asynchronous I/O Subsystem; for Linux, `libaio`). For details about the Asynchronous I/O Subsystem or `libaio`, see the documentation for the OS you are using.

Note that when you use the facility for parallel output of system logs in the Linux version, it is assumed that the platform is one of the following.

- Red Hat Enterprise Linux ES 4(AMD64 & Intel EM64T) or later

- Red Hat Enterprise Linux AS 4(AMD64 & Intel EM64T) or later

(a) Recommended usage

To shorten log output time, we recommend that the A and B systems be placed on separate devices.

(b) Environment settings (setting system definitions)

In the server definition, specify `pd_log_dual_write_method=parallel`.

- Dual system log files are not used (the value of the `pd_log_dual` operand is not Y).
- The system log files are not placed in character special files.

(c) Environment settings (setting the aio library)

Install the aio library on the server machine and configure the required settings. See the OS documentation for details about how to install and set up the aio library.

If the aio library installation and setting are not performed correctly using the environment settings described in (b), HiRDB cannot start.

■ Using AIX 5L

Asynchronous I/O Subsystem must be enabled prior to starting HiRDB. Otherwise, HiRDB will not start. On AIX 5L V5.2 and later, Asynchronous I/O Subsystem comes in a legacy version and a POSIX version. Since HiRDB uses the legacy version, enable the legacy Asynchronous I/O Subsystem.

After you have installed Asynchronous I/O Subsystem, set the following parameters.

Parameter	Recommended value
STATE to be configured at system restart	available
STATE of FastPath	enable [#]

[#]: This is the default, so no change is needed. If the setting is changed, performance might decline.

Other parameters do not require tuning.

(d) Notes

1. If Asynchronous I/O Subsystem is not enabled on HiRDB for AIX 5L, HiRDB cannot start and terminates abnormally. Either enable the Asynchronous I/O Subsystem, omit the `pd_log_dual_write_method` operand, or specify `serial`, and then restart HiRDB. If the Asynchronous I/O Subsystem parameter is set to `STATE to be configured at system restart = available`, HiRDB starts automatically.
2. The facility for parallel output of system logs is not applicable to system log files placed in regular files. If you add system log files, place them in character special files.
3. The facility for parallel output of system logs is applied only when both primary and secondary current files are placed in character special files and system log information can be output to those current files (they are not in closed, reserved, or error status). Parallel output of system logs does not take place, regardless of the system definition, if the current file satisfies either of the following conditions:
 - The primary or secondary current file is placed in a regular file.
 - The primary or secondary current file is in a status such that no log information can be output to it.
4. If the server that uses the facility for parallel output of system logs is running on multiple server machines, as when the system switchover facility is used, startup of the standby unit or system switchover might fail if the aio library is not enabled on any of the server machines. Therefore, enable the aio library on all server machines.

(5) Record length of a system log file (when the HiRDB system is already operating)

If the record length is not 1,024, we recommend that you change it to 1,024 to improve system log storage efficiency.

For details about how to change the system log file record length, see the *HiRDB Version 9 System Operation Guide*.

(6) Defining the system log files

The `pdlogadfg` and `pdlogadpf` operands are used to define the correspondence between file groups and the created system log files.

8.3.2 Designing synchronization point dump files

This section describes the design considerations for synchronization point dump files.

(1) Design considerations

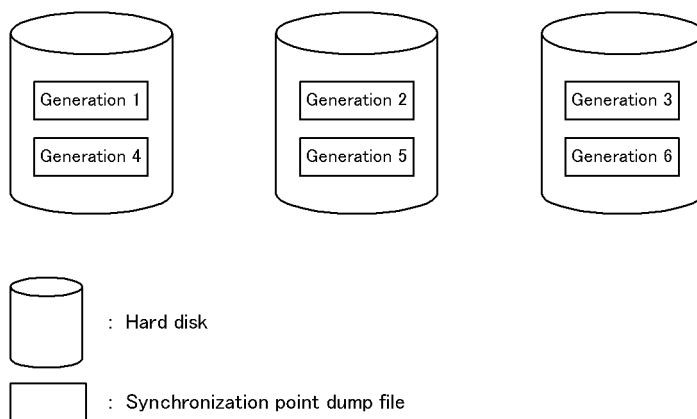
1. Synchronization point dump files are not required for a utility special unit.
2. Between 2 and 60 groups (if `ONL` is specified, 2 and 30 groups) of synchronization point dump files can be created per server.
3. HiRDB uses synchronization point dump files in the order specified in the `pdlogadfg -d spd` operand.
4. You should create at least four synchronization point dump files.
5. If a shortage of space occurs in a synchronization point dump file, HiRDB cannot be restarted. For this reason, the size of a synchronization point dump file should be set to be greater than the value specified for the maximum number of concurrent connections (`pd_max_users`) in the system common definition. For details about the calculation of synchronization point dump file size, see *17.2 Determining the size of synchronization point dump files*.

(2) Design for improved reliability

(a) Example of file organization

As a safeguard against the possibility of hard disk failures, you should create the synchronization point dump files on separate hard disks. If this is not possible, you should create adjacent generations of files on separate hard disks, as shown in the example in the following figure.

Figure 8–3: Example of creating adjacent generations on separate hard disks (HiRDB single server configuration)



(b) Duplexing of the synchronization point dump file

When the synchronization point dump file is duplexed, HiRDB collects the same synchronization point dump on both system A and system B. This increases system reliability, because when a collected synchronization point dump is read and there is an abnormality in the file, the synchronization point dump can still be read from the other file. Duplexing also enables the number of guaranteed-valid generations to be set to one generation, yet reliability is not compromised and the number of synchronization point dump files in overwrite disabled status is reduced.

Specify the following operands in the server definition to enable duplexing of synchronization point dump files:

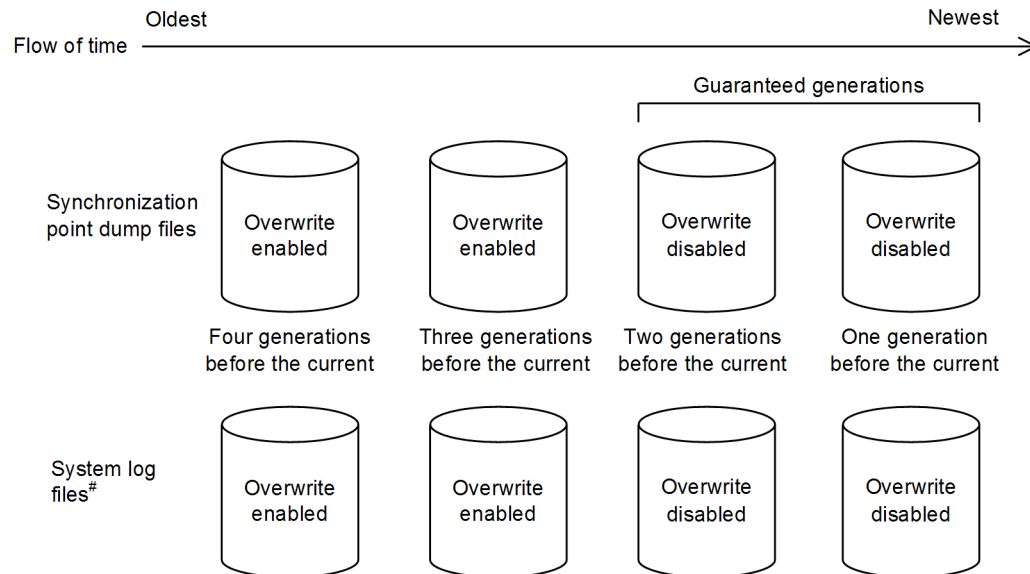
- `pd_spd_dual = Y`

- `-b` option in the `pdlogadpf` operand (specifies the system log file name on system B)

(c) Number of guaranteed-valid generations for synchronization point dump files

Each synchronization point dump acquired by HiRDB is stored in a separate synchronization point dump file. HiRDB uses the generation concept to manage synchronization point dump files. The HiRDB administrator specifies the number of generations of synchronization point dump files, and the corresponding system log files, that are to be placed in overwrite-disabled status. This concept is called the number of guaranteed-valid generations for synchronization point dump files, and it is illustrated in the following figure.

Figure 8–4: Number of guaranteed-valid generations for synchronization point dump files



#: These are the system log files that correspond to the synchronization point dump files.

Explanation

If there are two guaranteed-valid generations, the synchronization point dump files up to the second generation, and the system log files relevant to those synchronization point dump files, are in overwrite disabled status. The synchronization point dump files prior to the third generation, and the system log files relevant to those synchronization point dump files, are in overwrite enabled status.

The required number of synchronization point dump files is the number of guaranteed-valid generations + 1. Specify the number of guaranteed-valid generations for synchronization point dump files in the `pd_spd_assurance_count` operand in the server definition.

If synchronization point dump files are to be duplexed, it is recommended that only one guaranteed-valid generation be required. If duplexing is not to be used, two generations are recommended.

(d) Reduced mode operation for synchronization point dump files

If the number of synchronization point dump files available for use is reduced to the number of guaranteed-valid generations + 1 because of errors in synchronization point dump files, processing can continue with a minimum of two files. This is called the reduced mode operation for synchronization point dump files.

To perform reduced mode operation for synchronization point dump files, specify the `pd_spd_reduced_mode` operand in the server definition.

(e) Automatic opening of synchronization point dump files

When the number of synchronization point dump files available for use is reduced to the number of guaranteed-valid generations + 1 because of errors in synchronization point dump files, processing can continue by opening a reserved file and placing it in overwrite-enabled status (assuming that such a reserved file is available). This is called automatic opening of synchronization point dump files.

To perform automatic opening of synchronization point dump files, specify the `pd_spd_reserved_file_auto_open=Y` in the server definition.

(3) Defining the synchronization point dump files

The `pdlogadfg` and `pdlogadpf` operands are used to define the correspondence between file groups and the created synchronization point dump files.

If only the `pdlogadfg` operand is specified, synchronization point dump files can be added during HiRDB operation.

8.3.3 Designing status files

This section describes the design considerations for status files.

(1) Design considerations

1. Create the primary and secondary files on separate disks in order to avoid errors on both files.
2. To prevent abnormal termination of HiRDB as a result of a shortage of status file space, create several spare files whose size is greater than the estimated value. When a status file becomes full, file swapping occurs in order to use a spare file. If the size of the spare file is the same as the full status file, a space shortage also occurs on the spare file, resulting in abnormal termination of HiRDB. For example, if you create six sets of status files, we recommend that you make the file size for two of the sets larger than for the other sets.
3. Server status files are not required for a utility special unit. Create only unit status files.
4. Specify the same record length and number of records for both versions (A and B) of the status files.
5. You can create 1-7 sets of unit status files.
6. You can create 1-7 sets of server status files.

(2) Design for improved reliability

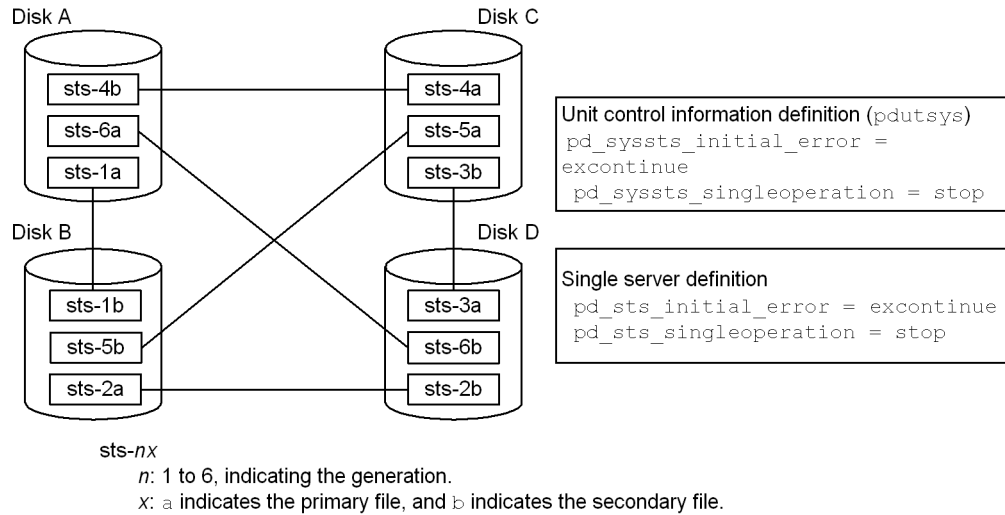
1. Provide at least three sets of status files (*dual files* \times 3 = 6 files) and place them in such a manner that corruption of all status files by disk errors is unlikely.
2. To prevent abnormal termination of HiRDB resulting from a space shortage, we recommend that you set the size of a status file to at least 1.2 times the estimated value.
3. A status file contains information that will be needed in order to restore the system status during HiRDB restart processing. If an error occurs in such a file and no spare file is available, the system status cannot be restored. Therefore, make sure that spare files are always available as a safeguard in the event of errors on the current files.

(a) Recommended configuration

In order to provide a safety margin until a disk becomes operational after it has been recovered from a disk failure, we recommend that you provide six sets of status files on four sets of disks (*dual files* \times 6 = 12 files) and place them as shown in the following figure. If an error occurs on the normal system during single operation, HiRDB cannot be restarted; therefore, we recommend that you do not apply single operation to status files (specify `stop` in `pd_syssts_singleoperation` and `pd_sts_singleoperation`).

The following figure shows an example of placing six sets of status files on four sets of disks.

Figure 8–5: Example of placing six sets of status files on four sets of disks

**Explanation:**

With this arrangement, if an error occurs on a disk, and then another error occurs on another disk, the remaining two disks still contain intact primary and secondary files, and HiRDB can continue operation using the status files on the error-free disks as the current files. For example, if an error occurs on disk A and then an error occurs on disk B, HiRDB continues operation using as the current files the primary and secondary status files on disks C and D (sts-3a and sts-3b). In this status, if another error occurs on one of the current files, HiRDB terminates abnormally; however, because one of the current files is normal, HiRDB can be restarted after one of the disks is recovered from its error.

(3) Defining the status files

The `pd_syssts_file_name_1` to `pd_syssts_file_name_7` and the `pd_sts_file_name_1` to `pd_sts_file_name_7` operands are used to define the correspondence between the status files created by the `pdutssinit` command and the logical files.

The `pd_syssts_file_name_1` to `pd_syssts_file_name_7` operands are for unit status files, and the `pd_sts_file_name_1` to `pd_sts_file_name_7` operands are for server status files.

If the names of imaginary logical files or status files are defined in the `pd_syssts_file_name2` to `pd_syssts_file_name7` operands or in the `pd_sts_file_name2` to `pd_sts_file_name7` operands, status files can be added during HiRDB operation. In such a case, the following operands must be specified.

Unit status files

```
pd_syssts_initial_error
pd_syssts_last_active_file
```

Server status files

```
pd_sts_initial_error
pd_sts_last_active_file
```

(4) Single operation of status files

If an error occurs on one of the current files while there is no available spare file, continuing operation using only the normal file (either the primary or secondary file) is called *status-file single operation*. When status files are placed in the single operation mode, the `KFPS01044-I` message is displayed.

If an error occurs on the current file in the single operation mode, HiRDB can no longer be restarted. Therefore, status-file single operation is not recommended. Increase the number of status file sets to avoid a situation where no spare file is available.

As opposed to status-file single operation, continuing operation using both status files (normal processing mode) is called *status-file double operation*.

(a) Advantages and disadvantages of status-file single operation

Advantages

Processing can continue even if an error occurs on one of the current files while no spare file is available. This reduces the adverse consequences of HiRDB shutdown resulting from a status file error.

Disadvantages

If an error occurs on the normal file during single operation or HiRDB terminates abnormally while the status file is being updated, the contents of the current file are lost, disabling HiRDB restart.

(b) Specification method

To use unit status-file single operation, specify `pd_syssts_singleoperation = continue` in the unit control information definition file. To use server status-file single operation, specify `pd_sts_singleoperation = continue` in the server definition. Make sure that `pd_syssts_singleoperation` and `pd_sts_singleoperation` have the same value.

- Relationship with other operands

The combination of the `pd_syssts_singleoperation` and `pd_syssts_initial_error` operand values or the `pd_sts_singleoperation` and `pd_sts_initial_error` operand values determines the HiRDB operation that is to take place if an error is detected in a status file during HiRDB startup. Therefore, determine the values of these operands together. For details about the HiRDB operation that is to take place if an error is detected in a status file during HiRDB startup, see the description of the `pd_syssts_initial_error` or `pd_sts_initial_error` operand in the manual *HiRDB Version 9 System Definition*.

(c) Usage guidelines

The following are guidelines on when to use status-file single operation.

- Do not use single operation if the primary goal is to avoid HiRDB being unable to restart.
- Use single operation if the primary goal is to avoid disabling the ability of HiRDB to go online.
- Do not use single operation if you have set HiRDB to restart automatically, such as at the time of a system switchover.

(d) Notes about using single operation

The following table outlines HiRDB operations and HiRDB administrator actions that depend on whether single operation is used. For details about how to handle status file errors, see the *HiRDB Version 9 System Operation Guide*.

Table 8–4: HiRDB operation and HiRDB administrator's action that depend on whether single operation is used

Condition		Status-file single operation (<code>pd_syssts_singleoperation</code> or <code>pd_sts_singleoperation</code> operand value)	
		Used (continue specified)	Not used (omitted or stop specified)
There are spare files	Error occurred in the current file	HiRDB operation: Swaps status files. HiRDB administrator's action: Handle the error in the applicable status file.	
	Error occurred on both current files simultaneously	HiRDB operation: Terminates abnormally. HiRDB cannot be restarted. HiRDB administrator's action: See <i>Handling of status file errors</i> in the <i>HiRDB Version 9 System Operation Guide</i> .	
There is no spare file	Error occurred in one of the current files	HiRDB operation: Resumes processing using single operation.	HiRDB operation: Terminates abnormally.

Condition		Status-file single operation (pd_syssts_singleoperation or pd_sts_singleoperation operand value)	
		Used (continue specified)	Not used (omitted or stop specified)
		HiRDB administrator's action: Create spare files immediately and return HiRDB to the double operation mode.	HiRDB administrator's action: Create spare files, and then restart HiRDB.
	Error occurred on both current files simultaneously	HiRDB operation: Terminates abnormally. HiRDB cannot be restarted. HiRDB administrator's action: See <i>Handling of status file errors</i> in the <i>HiRDB Version 9 System Operation Guide</i> .	
	Error occurred in the normal file during single operation	HiRDB operation: Terminates abnormally. HiRDB cannot be restarted. HiRDB administrator's action: See <i>Handling of status file errors</i> in the <i>HiRDB Version 9 System Operation Guide</i> .	--

Legend:

--: Not applicable

(5) Notes on status file errors (important)

- If errors occur on both current files simultaneously, HiRDB terminates abnormally and HiRDB can no longer be restarted. A possible measure for avoiding this situation is to use multiple physical disks (mirroring).
- If the current file (existing during termination) is deleted or initialized by the `pdstsinit` command prior to HiRDB startup, HiRDB can no longer be restarted.

8.4 Placing RDAREAs

This section discusses considerations concerning placement of the following types of RDAREAs:

- System RDAREAs
- Data dictionary LOB RDAREAs
- User RDAREAs
- User LOB RDAREAs
- List RDAREAs

8.4.1 Placing system RDAREAs

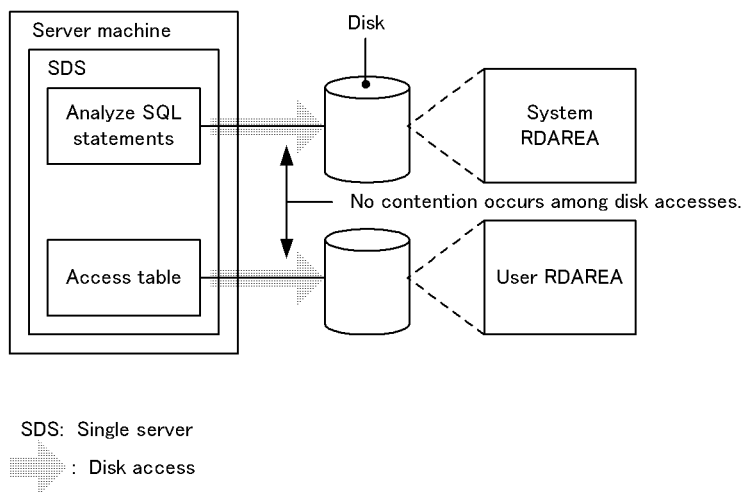
System RDAREAs should be placed taking into account the placement of user RDAREAs. Points to be considered when a system RDAREA is placed are discussed as follows.

A system RDAREA should not be placed on the same disk with user RDAREAs.

Among the system RDAREAs, data dictionary RDAREAs and data directory RDAREAs are accessed frequently by HiRDB for SQL statement analysis and other operations. If they are placed on the same disk as user RDAREAs, contention may occur between an access request for the purpose of SQL statement analysis and a table access request, in which case one of the requests will be placed on hold until the other request has been processed.

The following figure shows an example of system RDAREA placement that does not generate disk access contention.

Figure 8–6: Example of system RDAREA placement (HiRDB single server configuration)

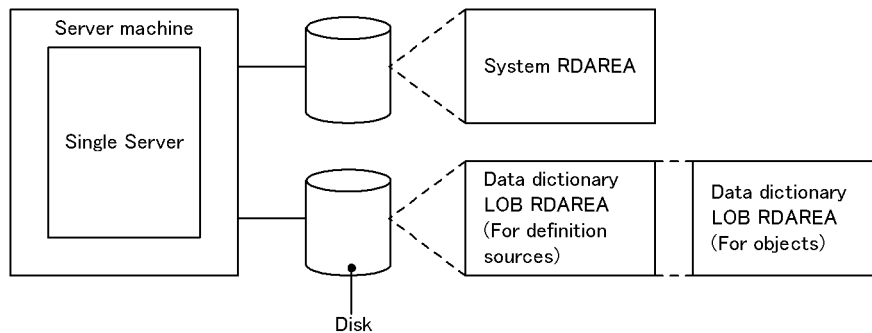


8.4.2 Placing data dictionary LOB RDAREAs

To avoid contention among disk accesses, a data dictionary LOB RDAREA should not be placed on the same disk as any other RDAREA.

The following figure shows an example of data dictionary LOB RDAREA placement.

Figure 8–7: Example of data dictionary LOB RDAREA placement (HiRDB single server configuration)

**Relationship with data dictionary RDAREAs**

A dictionary table used to manage stored procedures or stored functions can be placed in a separate data dictionary RDAREA from other dictionary tables.

8.4.3 Placing user RDAREAs

(1) Relationship with system log files

A user RDAREA should not be placed on the same disk as a system log file. When this rule is observed, input/output operations on HiRDB files that constitute system log files and on user RDAREAs can be distributed to multiple disks when a synchronization point dump is collected, thereby reducing the amount of time required for synchronization point dump processing.

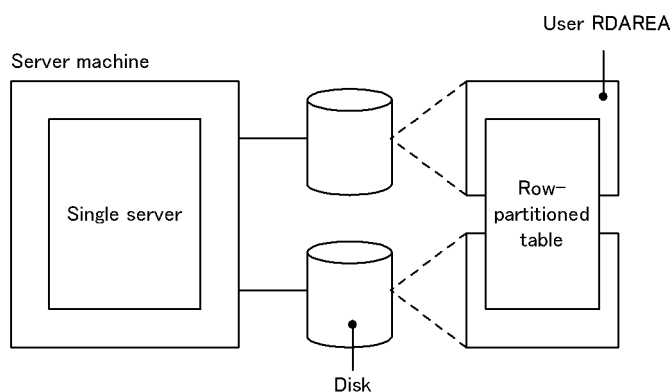
(2) Relationship with system RDAREAs

A user RDAREA should not be placed on the same disk as a system RDAREA.

(3) Row-partitioned tables

If you have partitioned a table by row, place the RDAREAs storing the row-partitioned table on separate disks. The following figure shows an example of user RDAREA placement.

Figure 8–8: Example of user RDAREA placement (HiRDB single server configuration)

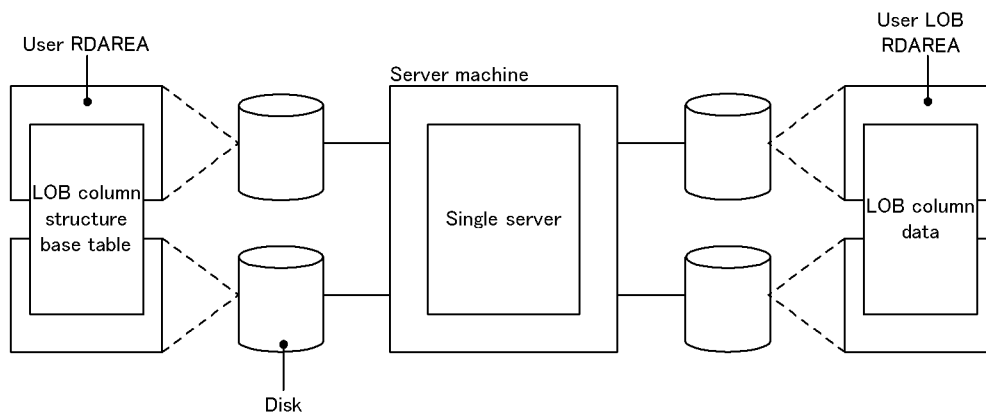


8.4.4 Placing user LOB RDAREAs

To avoid contention among disk accesses, a user LOB RDAREA should not be placed on the same disk as any other RDAREA.

If you have partitioned a table by row, place the RDAREAs storing the row-partitioned table on separate disks. The following figure shows an example of user LOB RDAREA placement.

Figure 8–9: Example of user LOB RDAREA placement (HiRDB single server configuration)



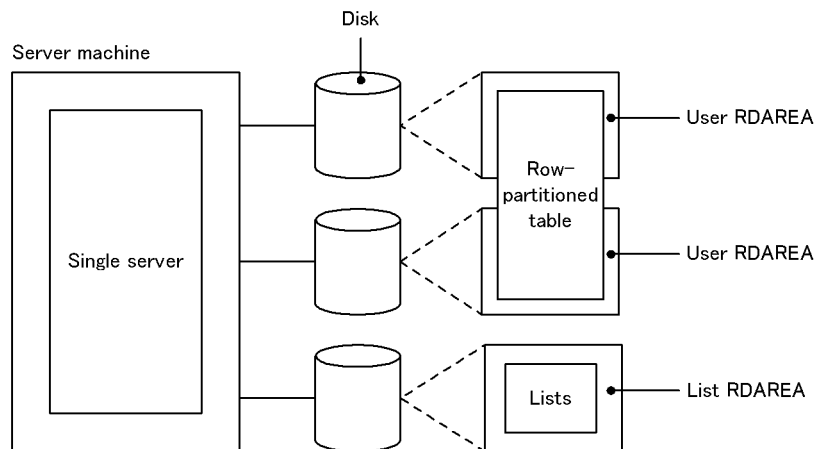
8.4.5 Placing list RDAREAs

To avoid contention among disk accesses, you should place list RDAREAs on a separate disk from any other RDAREAs.

Creating one list RDAREA lets you create lists for the tables that are stored in all user RDAREAs.

The following figure shows an example of list RDAREA placement.

Figure 8–10: Example of list RDAREA placement (HiRDB single server configuration)



9

Designing a HiRDB parallel server configuration

This chapter describes the design considerations for a HiRDB parallel server configuration, its HiRDB file system areas, and its system files, and provides notes on the placement of RDAREAs.

9.1 System design for a HiRDB parallel server configuration

This section describes the system design considerations and the system configuration for a HiRDB parallel server configuration.

9.1.1 System design

(1) Server configuration

The basic configuration of a HiRDB parallel server configuration consists of a front-end server, dictionary server, and back-end server on the same server machine.

If the CPU workload of the server machine is low, multiple servers may be placed on one server machine. In such a case, more shared memory is required. If there is not enough shared memory, unit startup fails; for this reason, sufficient memory must be allocated.

The following table shows the ranges for the number of servers that can be installed.

Table 9–1: Number of permitted servers

Item	Number of permitted servers
Number of system managers	1
Number of front-end servers	1 to 1,024
Number of dictionary servers	1
Number of back-end servers	1 to 16,382
Number of servers per unit	1 to 34

(2) Placement of system manager

The server machine on which the system manager is defined should be at a location that is easily accessible by the HiRDB administrator for the following reasons:

- The HiRDB administrator uses operation commands to operate HiRDB, and most operation commands must be entered from the server machine on which the system manager is defined.
- When HiRDB system definition files are shared, they should be placed on the server machine on which the system manager is defined. For details about how to share HiRDB system definition files, see *4.2.3 Sharing HiRDB system definition files (HiRDB parallel server configuration)*.

(3) Placement of floating server

When a complicated retrieval such as join processing is executed, it is better for HiRDB to use a back-end server that does not have a database in order to improve performance. If the server machine has sufficient space and complicated retrieval processing is to be performed, installation of a floating server should be considered. When a floating server is installed, a HiRDB file system area for work table files must be created. The name of this HiRDB file system area is specified in the `pdwork` operand of the back-end server definition.

(4) Using multiple front-end servers

If the CPU workload for SQL processing is too high to be processed on the front-end server, multiple front-end servers can be set up. This is called multiple front-end servers; for details, see *9.1.3 Setting up multiple front-end servers*.

(5) Memory used by a HiRDB parallel server configuration

This subsection describes the memory used by a HiRDB parallel server configuration.

A HiRDB parallel server configuration uses the following memory.

- Shared memory
- Process private memory

(a) Storage requirements

The storage space required by the HiRDB parallel server configuration must be estimated for each server machine. For details about how to estimate the storage requirements, see *15.2 Estimating the memory size required for a HiRDB parallel server configuration*.

(b) Page fixing of shared memory

With HiRDB, the following types of shared memory can be fixed in actual memory.

- Shared memory for unit controllers
- Shared memory for global buffers
- Shared memory used by dynamically changed global buffers
- Shared memory for in-memory data buffers

Fixing shared memory in actual memory reduces the number of page I/Os, resulting in more stable performance.

Prerequisites

The following table shows the prerequisites for page fixing of shared memory for each OS.

OS	Prerequisites
HP-UX	None
Solaris	None
AIX	Must be 64-bit mode
Linux	None

Operating environment settings

AIX requires you to set operating system parameters. For details, see *20.3(1) Specifying parameters unique to AIX*.

Page fixing methods

This subsection describes shared memory page fixing methods for each type of shared memory.

- Shared memory for unit controllers
Specify *fixed* in the `pd_shmpool_attribute` operand of the system common definition or unit control information definition.
- Shared memory for global buffers
Specify *fixed* in the `pd_dbbuff_attribute` operand of the system common definition or unit control information definition.
- Shared memory used by dynamically changed global buffers
Specify *fixed* in the `pd_dbbuff_attribute` operand of the system common definition or unit control information definition. This fixes shared memory used by global buffers dynamically changed by the `pdbufmod` command in actual memory.
- Shared memory for in-memory data buffers
Specify *fixed* in the `pdmemdb` command `-p` option.

! Important note

When contiguous areas cannot be secured in actual memory, shared memory pages cannot be fixed. HiRDB operation when page fixing fails is as follows.

OS	HiRDB operation when page locking fails			
	Unit controller shared memory	Global buffer shared memory	Dynamically changed global buffer shared memory	In-memory data buffer shared memory
HP-UX	Y	N	N	N
Solaris	Y	N	N	N
AIX	Y [#]	Y [#]	Y [#]	Y [#]
Linux	Y	N	N	N

Legend

Y: Shared memory is secured without fixing pages, and processing continues.

N: HiRDB or the command terminates abnormally.

#

In AIX, system calls terminate normally, even when page fixing fails. This means that you cannot tell from HiRDB that page fixing failed. Use the following procedure to check whether pages were fixed.

1. While HiRDB is running, execute the `pdls -d mem` command to check the identifier of the following shared memory segment.

- For shared memory for unit controllers, the shared memory segment with `MANAGER` displayed under `SHM-OWNER`.
- For other types of shared memory, the shared memory segment with a character string consisting of the unit name in parentheses or the HiRDB server name displayed under `SHM-OWNER`.

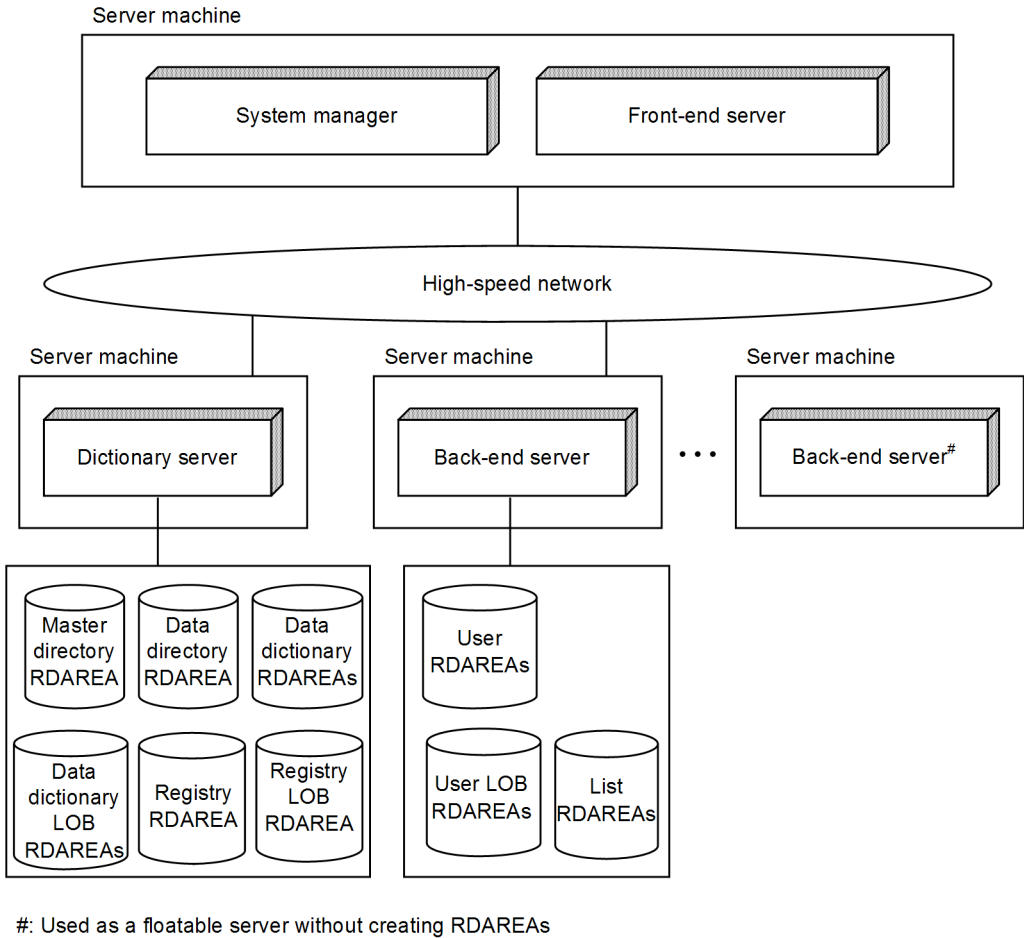
2. Execute the OS's `ipcs -s` command, and then check the `SID` value of the shared memory that has the identifier of the shared memory segment you checked in step 1.

3. Execute the OS's `svmon` command on the `SID` value you checked in step 2, and then check whether the number of actual memory pages of the shared memory in question matches the number of fixed pages.

9.1.2 System configuration of a HiRDB parallel server configuration

The following figure shows an example of a system configuration for a HiRDB parallel server configuration. The system configuration of a HiRDB parallel server configuration is defined in the HiRDB system definition. For a definition example of the HiRDB system configuration shown in the following figure, see the manual *HiRDB Version 9 System Definition*.

Figure 9–1: System configuration for a HiRDB parallel server configuration



9.1.3 Setting up multiple front-end servers

A HiRDB parallel server configuration uses multiple back-end servers to process multiple SQL processing in parallel. The front-end server is responsible for such operations as executing SQL analysis and SQL optimization, sending instructions to the back-end servers, and editing retrieval results. In a system with heavy traffic, the front-end server's workload may be very high, resulting in adverse effects on processing performance. In such a case, multiple front-end servers can be installed in order to distribute the workload. This is called multiple front-end servers.

Advantages

Throughput bottlenecks are resolved for each server machine where a front-end server is running, and its scalability is improved.

Criteria

The CPU workload in SQL processing is too high to be processed by one server machine.

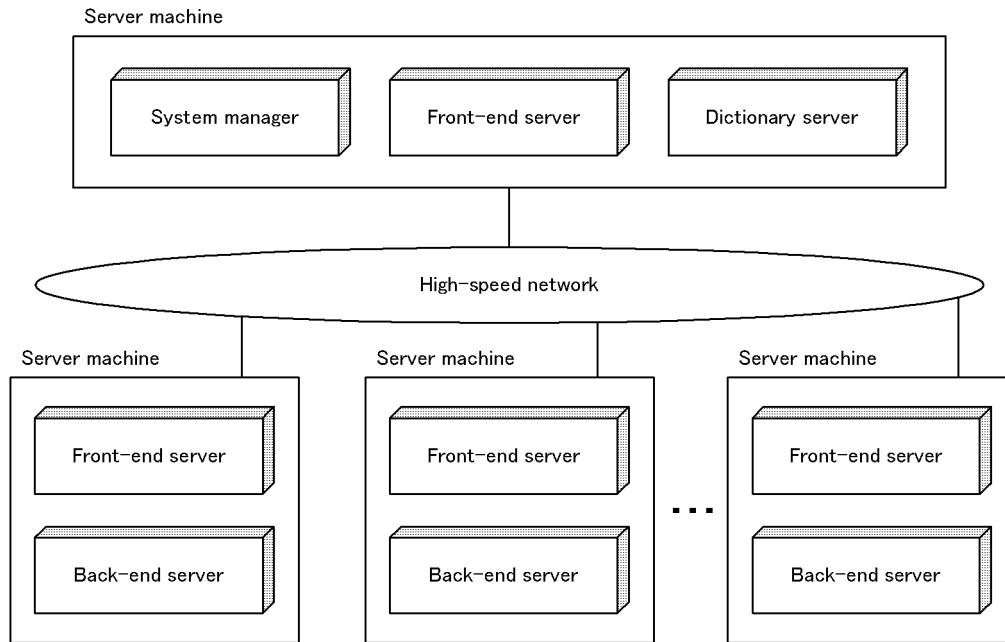
Rules

A maximum of 1,024 front-end servers can be installed.

Relationship to server machine

Multiple front-end servers cannot be set up in a single unit. It is also possible not to set up a front-end server in a specific unit. The following figure shows an example of a configuration of multiple front-end servers.

Figure 9–2: System configuration for a HiRDB parallel server configuration with multiple front-end servers



(1) Selecting a front-end server to be connected

When there are multiple front-end servers, the front-end server that is to be connected to a UAP is determined as follows.

- Client user
The client user can specify the front-end server to be connected to a UAP using the `PDFESHOST` operand in the client environment definition.
- HiRDB
HiRDB automatically determines the front-end server to be connected to a UAP.
If no particular front-end server is specified in the client environment definition, HiRDB selects an appropriate front-end server for connection of the UAP.

(2) Environment setup

Because there are multiple front-end servers executing, no special specifications are usually required.

However, appropriate values must be specified in the following operands (for details about the standard values for these operands, see the manual *HiRDB Version 9 System Definition*):

- `pd_max_dic_process`
- `pd_max_bes_process`

(3) HiRDB administrator operation

The operating procedures are the same, but there are differences in the operands that are specified in the HiRDB system definition and in the environment setup procedures, such as the number of system files to be created.

(4) Sorting by insertion or update time in a configuration of multiple front-end servers

If a table contains a timestamp-type column for which the `DEFAULT` clause with `CURRENT_TIMESTAMP` as the default value was specified during table definition, you must take precautions when you sort the table by row insertion or update time in a configuration of multiple front-end servers.

In the case of multiple front-end servers, the front-end server connected to the UAP acquires the current timestamp and sets that value as the default value for the timestamp column. Note that the system time may not match between the units containing the front-end servers. If the system time does not match, the sort order based on the timestamp column and the sort order based on the actual row insertion or update time will not match.

To match the sort order, specify the `DEFAULT` clause with `CURRENT_TIMESTAMP USING BES` as the default value for the timestamp column during table definition. If you specify `USING BES`, the back-end server that manages the RDAREA storing the row to be inserted or updated is used to acquire the current timestamp, and then that value is inserted in the row or the row is updated by that value. As a result, the sort order based on the timestamp-type column matches the sort order based on the actual row insertion or update time for each unit that contains the back-end server managing the RDAREA that stores the row.

The following table shows the server that acquires the current timestamp depending on whether `USING BES` is specified.

Table 9–2: Server that acquires the current timestamp depending on whether `USING BES` is specified

USING BES	Server that acquires the current timestamp
Not specified	Front-end server that connected to the UAP
Specified	Back-end server that manages the RDAREA containing the row to be inserted or updated

Notes

- If a table is partitioned and the table storage RDAREA is managed by multiple back-end servers located on different units, the sort order based on the value of the timestamp-type column may not match the sort order based on the actual row insertion or update time.
- In the case of a shared table, you cannot insert the default value in a timestamp-type column or update it by the default value unless the table is locked.
- If the database load utility (`pdload`) is used to store data in a table, the time the utility was started by the activated unit is set as the timestamp value.

(5) Client user operation

To enable a client user to select the front-end server for connection, the desired front-end server must be specified in the client environment definition. The client environment definition varies depending on whether the high-speed connection facility or the FES host direct connection facility is used. The following table shows the client environment definitions that must be specified. For details about the client environment definitions, see the *HiRDB Version 9 UAP Development Guide*.

Table 9–3: Client environment definitions required for multiple front-end servers

Client environment definition operand	Not specifying a front-end server to be connected	Specifying a front-end server to be connected	
		FES host direct connection	High-speed connection
PDHOST	M	M	M
PDFESHOST	--	M	M
PDNAMEPORT	M	M	M
PDSERVICEPORT	--	--	M
PDSERVICEGRP	--	M	M
PDSRVTYPE	--	--	--

M: Must be specified.

--: Need not be specified.

(a) Guidelines for determining the front-end server to be specified

- You should specify the front-end server on the server machine where the back-end server that manages the RDAREA to be accessed is also located.
- You should select an appropriate front-end server according to its processing purposes. For example, separate front-end servers may be used for general information retrieval processing, batch UAP processing, and UAP processing under OLTP.

(b) HiRDB server connection time

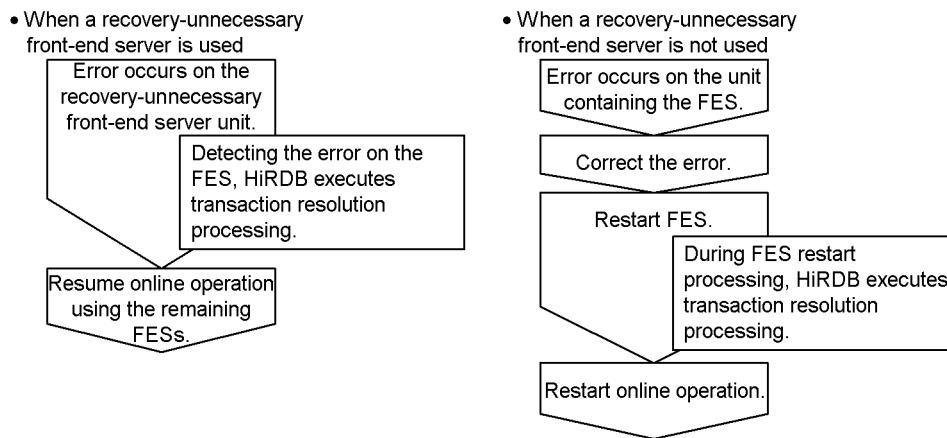
The HiRDB server connection time increases from 1 to 3 as follows (with 1 being the shortest):

1. High-speed connection facility
2. FES host direct connection facility
3. No front-end server specified for connection

9.1.4 Recovery-unnecessary front-end server

If the unit containing the front-end server terminates abnormally due to an error, the transaction that was being executed from that front-end server may be placed in uncompleted status. Because uncompleted transactions lock the database, some database referencing or updating may be limited. To resolve an uncompleted transaction, normally the front-end server must be recovered from the error and then restarted. If the abnormally terminated front-end server is a recovery-unnecessary front-end server, HiRDB automatically resolves the uncompleted transaction. This enables you to use another front-end or back-end server to restart database update processing. A unit that contains a recovery-unnecessary front-end server is called a *recovery-unnecessary front-end server unit*. The following figure shows operations based on whether a recovery-unnecessary front-end server is used.

Figure 9–3: Operation based on whether a recovery-unnecessary front-end server is used



FES: Front-end server

Note that HiRDB Non Recover FES is required in order to use recovery-unnecessary front-end servers.

Advantages

You can continue online operation using the remaining front-end servers without having to restart the erroneous front-end server.

Criteria

We recommend that you use recovery-unnecessary front-end servers in a system that requires non-stop operation 24 hours a day.

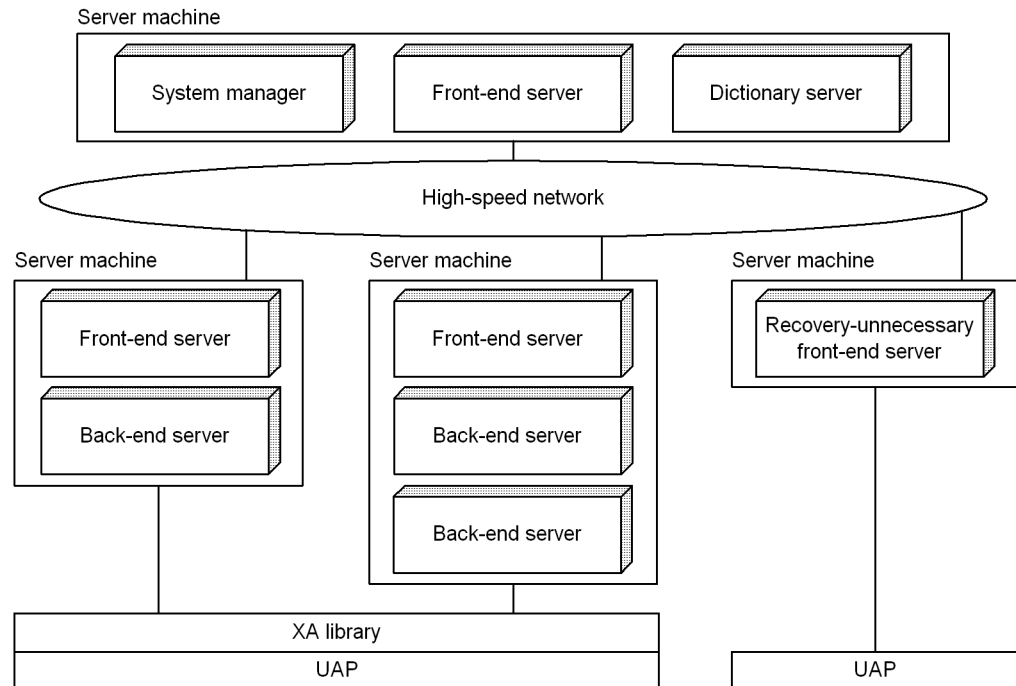
Relationship with other front-end servers

- Place a recovery-unnecessary front-end server on an independent unit.

- A recovery-unnecessary front-end server cannot support a UAP that uses the X/Open XA interface for connection. Specify `PDFESHOST` and `PDSERVICEGRP` in the client environment definition and connect such a UAP to a non-recovery-unnecessary front-end server.
- You can execute the `pdrplstart` and `pdrplstop` commands even when the recovery-unnecessary front-end server and recovery-unnecessary front-end server unit are inactive.

The following figure shows an example of a system configuration using a recovery-unnecessary front-end server.

Figure 9-4: Example of a system configuration using a recovery-unnecessary front-end server



- A recovery-unnecessary front-end server cannot execute import processing using the two-phase commitment method for the synchronization point processing method (enabled when `fxa_sqle` is specified in the import system definition `commitment_method` operand) of the target Datareplicator. To use the two-phase commitment method for the synchronization point processing method of the target Datareplicator, you need to place one or more front-end servers other than the recovery-unnecessary front-end server at the target HiRDB. You also need to set the client environment variables `PDFESHOST` and `PDSERVICEGRP` at the target Datareplicator to connect to a front-end server other than a recovery-unnecessary front-end server.

Relation with other facilities

- Recovery-unnecessary front-end servers cannot use the system switchover facility. To use the system switchover facility with the system, you must specify `nouse` in the `pd_ha_unit` operand in the unit control information definitions of the recovery-unnecessary front-end server unit.

(1) Setup method

To use a recovery-unnecessary front-end server, specify `stls` in the `-k` option of the `pdstart` operand.

(2) Notes

1. If the recovery-unnecessary front-end server does not start during HiRDB startup, HiRDB continues startup processing excluding the corresponding unit, regardless of the value specified in the `pd_start_level` operand. If all front-end servers in use are recovery-unnecessary front-end servers, HiRDB system startup cannot be completed unless at least one front-end server starts successfully.
2. Recovery-unnecessary front-end servers are independently subject to reduced activation. HiRDB ignores the name of a recovery-unnecessary front-end server in the `pd_start_skip_unit` operand, if specified.

3. If a recovery-unnecessary front-end server terminates abnormally, the status information for the front-end server and unit is `STOP (A)`. Unlike the normal `STOP (A)`, this status allows the `pdstop` command to perform normal termination or planned termination on HiRDB's system manager and other units. If a recovery-unnecessary front-end server is terminated forcibly, the status information for the front-end server and unit is `STOP (F)`. However, this status allows the `pdstop` command to perform normal termination or planned termination on HiRDB's system manager and other units.
4. A recovery-unnecessary front-end server always starts the unit normally except in the following case:
 - The unit was terminated by a method other than normal termination, and `stls` was not specified in the `-k` option of the `pdstart` operand during the previous session.
5. If the status information for a recovery-unnecessary front-end server is `STOP (A)`, HiRDB stops accepting SQL requests from a UAP that has established connection with that recovery-unnecessary front-end server. In this case, the `KFPS01820-E` message displays `c800` as the process termination status of the recovery-unnecessary front-end server. The server process termination status of a back-end server, dictionary server, or other server that contains data on which an operation was attempted in SQL, might also be displayed in the `KFPS01820-E` message as `c900`. If the `KFPS01820-E` message is displayed, use `pdstop -z` to terminate the unit containing the front-end server whose process termination status is `c800`, eliminate the cause of the status `STOP (A)`, and then restart.
6. When a recovery-unnecessary front-end server unit is running but its status information is `STOP (A)` because of a problem such as a network error, once the server unit has recovered from the error and the system manager is again able to communicate with the unit, the system manager will forcibly stop the unit automatically and then restart it. The system manager will do this in the following circumstances:
 - When the unit monitoring process that monitors the unit's operating status confirms that the recovery-unnecessary front-end server unit whose status information changed to `STOP (A)` is running, and outputs a `KFPS05288-I` message.
 - When a unit that has a system manager checks the operating status of all units when it restarts, and confirms that the recovery-unnecessary front-end server unit whose status information changed to `STOP (A)` is running.

During automatic restart after a forced stop, check whether a `KFPS05110-I` message was output by the recovery-unnecessary front-end server unit. If it was, start processing of the recovery-unnecessary front-end server unit terminated normally. If the `KFPS05110-I` message was not output within the time specified in the `pd_system_complete_wait_time` operand of the system common definition after the circumstance that triggered the automatic forcible stop and restart, start processing did not terminate normally. Take the following corrective action.

(1) Use the `pdls -d ust` command to check the operating status of units that have system managers and units that have a dictionary server.

If those units are not operating, start them with the `pdstart` command.

If they are operating, or if they have started but the `KFPS05110-I` message has not been output, take the corrective action described in (2).

(2) Check the operating status of the recovery-unnecessary front-end server unit using the `pdls -d ust` command, and take the corrective action described below that corresponds to the execution result.

Operating status of recovery-unnecessary front-end server unit (pdls -d command execution result)	Corrective action
STOP (stopped)	Execute the <code>pdstart -q</code> command, and then restart the recovery-unnecessary front-end server unit.
PAUSE (restart of the process server process is paused)	<ol style="list-style-type: none"> 1. Check the <code>KFPS00715-E</code> message and any messages previously output to the syslogfile. Remove the cause of the error and then execute the <code>pdrpause</code> command. 2. Execute the <code>pdstart -q</code> command, and then restart the recovery-unnecessary front-end server unit.
STARTING	<ol style="list-style-type: none"> 1. Execute the <code>pdstop -z</code> command, and then forcibly terminate the recovery-unnecessary front-end server unit. 2. Execute the <code>pdstart -q</code> command, and then restart the recovery-unnecessary front-end server unit.
ONLINE (operating)	
STOPPING	

Note that when a unit stops before the system manager is able to communicate with that unit, the system manager does not forcibly stop and restart the unit.

7. If a transaction is processed at another server branched out from the recovery-unnecessary front-end server, its completion is synchronized with the target server when the transaction is committed. If the target back-end server or dictionary server cannot execute transaction processing at the time of synchronization (because system switchover is underway, the server is stopped, the server is not ready for startup, or the server is not ready for termination), processing might be queued in a state in which the first transaction status is set to `READY` or `COMMIT`. If this happens, check the server to determine the cause of the queuing of transaction processing and take appropriate action so that the transaction resolution processing can be resumed.
8. You may not be able to use the `pdcmr`, `pdbrk`, or `pdfgt` command to forcefully terminate a transaction for which processing was performed by connection to a front-end server that uses the recovery-unnecessary FES facility, regardless of whether the transaction is in first or second status. In such a case, see *Forcing determination of uncompleted transactions* in the *HiRDB Version 9 System Operation Guide* for details about how to automatically resolve an uncompleted transaction.

9.2 Designing HiRDB file system areas

When a HiRDB system is constructed, areas for HiRDB-specific files (HiRDB files) must be created. This section discusses the design considerations for creating HiRDB file system areas.

Separate HiRDB file system areas should be created for the types of items listed below, so that contention between input/output operations on files with different purposes or access characteristics can be avoided. If regular files are used, write performance can be improved by specifying the purpose explicitly, and an appropriate device can be allocated according to the purpose. Separate file system areas should be created for:

- RDAREAs
- Shared RDAREAs
- System files
- Work table files
- Utilities
- RDAREAs for lists (list RDAREAs)

9.2.1 Designing HiRDB file system areas for RDAREAs

This section discusses the design considerations for HiRDB file system areas in which RDAREAs are to be created.

(1) Design for improved reliability

1. For purposes of update processing, character special files are more reliable than regular files. Regular files may become inaccessible if the OS terminates abnormally. Thus, character special files should be used for HiRDB file system areas for user RDAREAs that satisfy the following characteristics:
 - User RDAREAs for storing tables that are used primarily for update processing.
 - User RDAREAs for storing particularly important data.
2. Calculate the size of the required HiRDB file system area, and create a HiRDB file system area of that size or larger.
3. HiRDB file system areas for RDAREAs must be created on a server machine where the following servers are defined:
 - Dictionary server
 - Back-end server
4. HiRDB file system areas for the following RDAREAs can be created only on a server machine where a dictionary server is defined:
 - System RDAREAs
 - Data dictionary LOB RDAREAs
 - Registry RDAREA
 - Registry LOB RDAREA
5. HiRDB file system areas for the following RDAREAs can be created only on a server machine where back-end servers are defined:
 - User RDAREAs
 - User LOB RDAREAs
6. If the system switchover facility is to be used, the HiRDB file system areas for user RDAREAs should be allocated as character special files.

(2) Design for improved performance

1. You should create separate HiRDB file system areas for the following types of RDAREAs:

- System RDAREAs
 - Data dictionary LOB RDAREAs
 - User RDAREAs
 - User LOB RDAREAs
 - Registry RDAREAs
 - Registry LOB RDAREAs
2. You should create HiRDB file system areas for system files on separate hard disks separately from the ones used for HiRDB file system areas for RDAREAs. In this way, you can distribute input/output operations when collecting a synchronization point dump, thereby reducing the amount of time required to collect the synchronization point dump.
 3. If you are not using the prefetch facility, the sequential read speed is faster with regular files than with character special files.
 4. For random one-page reads, processing speed is faster with character special files than with regular files.
 5. For write processing, the speed is faster with character special files than with regular files.
 6. Regular files have a hierarchical structure; therefore, as files become larger, the hierarchical levels increase. When you access such files with many hierarchical levels, the number of input/output operations increases, adversely affecting the access efficiency.
 7. The HiRDB file system areas should be allocated as shown in the following table, so that input/output time can be reduced.

Table 9–4: Allocation of HiRDB file system areas to improve performance

Type of HiRDB file system area	File to be allocated
HiRDB file system area for system RDAREAs	Character special file
HiRDB file system area for dictionary LOB RDAREAs	
HiRDB file system area for user LOB RDAREAs	
HiRDB file system area for user RDAREAs that store frequently updated tables or tables primarily subject to retrieval of small amounts of data	
HiRDB file system area for user RDAREAs that store tables with a large amount of data that are primarily subject to retrieval of all entries or retrieval of large amounts of data by key using the cluster key (but with data that is rarely updated)	<ul style="list-style-type: none"> • Regular file (when not using the prefetch facility) • Character special file (when using the prefetch facility)

9.2.2 Designing HiRDB file system areas for system files

This section discusses the design considerations for HiRDB file system areas in which system files are to be created.

(1) Design for improved reliability

1. For update processing, character special files are more reliable than regular files. Regular files may become inaccessible after a system shutdown. Thus, character special files should be used for HiRDB file system areas for system files.
2. Create at least two HiRDB file system areas for system files. If there is only one HiRDB file system area for system files, HiRDB cannot continue operating in the event of a hard disk failure at the disk containing the system files.
3. Create HiRDB file system areas for system files on separate hard disks. In this way, in the event of a hard disk error, you can restart HiRDB using the other hard disk.
4. Calculate the size of the required HiRDB file system area, and create a HiRDB file system area of that size or larger.

(2) Design for improved performance

You should create HiRDB file system areas for system files on separate hard disks separately from the ones used for HiRDB file system areas for RDAREAs. In this way, you can distribute input/output operations when collecting a synchronization point dump, thereby reducing the amount of time required to collect the synchronization point dump.

9.2.3 Designing HiRDB file system areas for work table files

This section discusses the design considerations for HiRDB file system areas in which work table files are to be created.

(1) Design for improved reliability

1. Work table files can be allocated to regular files. However, if the system switchover facility is used, disk space can be saved by allocating work table files to character special files because the files can then be shared.
2. The amount of space required for a HiRDB file system area for work table files must be greater than the total size of the work table files to be created in the area. If you specify the `-a` option with the `pdfmkfs` command, the HiRDB file system area can be automatically extended. We recommend specifying the `-a` option, since this allows the HiRDB file system area to be extended automatically when the total size of the work table files reaches the size of the HiRDB file system area.[#]

For details about the sizes of work table files, see *Chapter 18. Determining Work Table File Size*.

3. A HiRDB file system area for work table files must be created at the server machines where the following servers are defined:

- Dictionary server
- Back-end server

[#]

To reduce the amount of disk space that is occupied by the HiRDB file system area for work table files when HiRDB restarts, before you restart HiRDB, execute the `pdfmkfs` command and then re-initialize the HiRDB file system area for work table files.

(2) How to check the peak capacity

You can use the following command to obtain the peak capacity of a HiRDB file system area for work table files:

```
pdfstatfs -d name-of-HiRDB-file-system-area-for-work-tables
```

`-d`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be displayed. The peak capacity display that is output is this value. The maximum utilization value is cleared by entering the following `pdfstatfs` command:

```
pdfstatfs -c name-of-HiRDB-file-system-area-for-work-tables
```

`-c`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be cleared to 0.

9.2.4 Designing HiRDB file system areas for utilities

This section discusses the design considerations for HiRDB file system areas in which utility files (backup files, unload data files, and unload log files) are created. Use the HiRDB file system areas for utilities to create the following files:

- Backup files
- Unload data files
- Unload log files
- Differential backup management files

(1) Design considerations

1. If you are creating a HiRDB file system area for backup files, allocate it to character special files.
2. The amount of space required for a HiRDB file system area for backup files must be greater than the total size of the RDAREAs that will be backed up. For details about the sizes of RDAREAs, see *Chapter 16. Determining RDAREA Size*.
3. Create a HiRDB file system area for differential backup management files on the server machine where the system manager is located.
4. If you use the system switchover facility, create unload log files on a shared disk (character special files).
5. If you are creating a HiRDB file system area for unload log files, specify the following options in the `pdfmkfs` command:
 - `-k` option: Specify `UTL` (HiRDB file system area for utilities) as the usage.
 - `-n` option: For the size of the HiRDB file system area, specify the value obtained from the following formula:

$$(total\ record\ count\ for\ system\ log\ file\ to\ be\ unloaded)^{\#1} \times (system\ log\ file\ record\ size)^{\#2} \times number\ of\ unload\ log\ files\ to\ be\ created$$

$$zueng005.tif\ 1.2 \div 1,048,576$$
 - `-l` option: For the maximum number of files, specify the number of unload log files to be created.
 - `-e` option: For the maximum number of extensions, specify the number of unload log files to be created times 24.

#1

If you are using the system log file automatic extension facility, calculate this number using the value specified as the upper limit for expansion in the `pd_log_auto_expand_size` operand.

#2

This is the approximate value of the system log file.

(2) How to check the peak capacity

You can use the following command to obtain the peak capacity of a HiRDB file system area for utilities:

```
pdfstatfs -d name-of-HiRDB-file-system-area-for-utilities
```

`-d`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be displayed. The peak capacity display that is output is this value. You can clear the maximum utilization value by entering the following `pdfstatfs` command:

```
pdfstatfs -c name-of-HiRDB-file-system-area-for-utilities
```

`-c`

Specifies that the maximum utilization value for the allocated HiRDB file system area is to be cleared to 0.

9.2.5 Designing HiRDB file system areas for list RDAREAs

This section discusses the design considerations for HiRDB file system areas in which list RDAREAs are to be created.

(1) Design considerations

1. A list is used to store temporary intermediate results of search processing. Therefore, it does not need to be as reliable as other RDAREAs. You can create HiRDB file system areas for list RDAREAs in regular files.
2. If you use the system switchover facility, you can save disk space by allocating HiRDB file system areas for list RDAREAs to character special files because the files can then be shared.
3. Create a HiRDB file system area for list RDAREAs on the same back-end server that contains the base table.

(2) Design for improved performance

1. If you are creating HiRDB file system areas for list RDAREAs on RAID, you should use character special files to reduce processing time. If you are creating them on any other disk, you should use regular files to reduce processing time.
2. You should create HiRDB file system areas for list RDAREAs on a separate hard disk separately from the ones used for the following HiRDB file system areas. In this way, you can distribute input/output operations when searching a list, thereby reducing the processing time.
 - HiRDB file system areas for user RDAREAs
 - HiRDB file system areas for user LOB RDAREAs
 - HiRDB file system areas for work table files

9.2.6 Maximum sizes of HiRDB file system areas

The following table lists the maximum sizes of the HiRDB file system areas.

Table 9–5: Maximum sizes of HiRDB file system areas

HiRDB type	Conditions		Maximum size of HiRDB file system area (MB)
HP-UX	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file	1,048,575
		Character special file	
Solaris	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file	1,048,575
		Character special file	
AIX	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file (JFS)	65,411
		Regular file (JFS2)	1,048,575
		Character special file	
Linux	Large file not used	Regular file	2,047
		Character special file	
	Large file used	Regular file	1,048,575
		Character special file	

9.3 Designing system files

This section discusses design considerations for various system files.

9.3.1 Designing system log files

The following explains some of the design considerations for system log files.

(1) Design considerations

1. System log files are required for each server, except for the system manager.
2. Specify the same record length and number of records for all system log files on the same server.
3. The number of system log files that can be created for each server is 2 to 200 groups (we recommend creating at least six groups).
4. For one server, creation of system log files must meet the following condition:

$$100 \text{ (unit: megabytes)} \times (200 - \text{number of system log file groups assigned to server}) \geq \text{total system log file capacity allocated to the server} \times 3$$

Multiplying by three is necessary to restart a HiRDB system that has terminated abnormally due to insufficient space for the system log files. Depending on usage and whether there is enough space for system log files, you might need to take action by creating system log files of three times the capacity allocated to the server, and then adding them to the server.

5. To reduce the number of unload operations, it is advisable to create many large system log files.
6. If the system switching facility is used, a file that is involved in many input/output operations (such as a log unload file) should not be created on the same disk that contains \$PDDIR%PDDIR%.
7. The amount of space required for one system log file must satisfy the condition shown following:

$$\text{Size of one system log file (bytes)} \geq \uparrow (a + 368) \div c \uparrow \text{ zueng005.tif } c \text{ zueng005.tif } b \text{ zueng005.tif } d$$

- a*: Value of `pd_log_max_data_size` operand
- b*: Value of `pd_log_sdinterval` operand
- c*: Value of `pd_log_rec_leng` operand
- d*: Value of `pd_spd_assurance_count` operand

8. The total size of the system log files (if duplexed, only their size in one of the systems) must meet the following two conditions.

Condition 1

The size must be at least the value calculated according to the formula in 17.1.1(1) *How to obtain the total size of system log files*.

Condition 2

After the start of the large object transaction, system log files cannot be overwritten until the large object transaction finishes. Moreover, the current generation of system log files and system log files corresponding to the number of guaranteed-valid generations of the synchronization point dump file cannot be overwritten either. For this reason, make sure that there is at least enough system log file capacity to accommodate this amount of data. Use the following equation to estimate the required capacity.

$$\text{System log file total size (bytes)} \geq 3 \text{ zueng005.tif } a \text{ zueng005.tif } (b + 1)$$

a:

Size of system log information that may be output at the corresponding server while executing the database updating transaction with the longest execution time.

For details about the formula for estimating the size of system log information, see 17.1 *Determining the size of system log files*.

b: Value of `pd_spd_assurance_count` operand

Number of guaranteed-valid generations for synchronization point dump files.

(a) Effects on operations of the number of generations of system log files

If the total size of the system log files is unchanged, the size of each generation will depend on how many generations of system log files are being maintained. The following table describes the effect on operations of the number of generations of system log files. The total size of the system log files is unchanged.

Table 9–6: Effects on operations of the number of generations of system log files

Comparison item	System log file configuration	
	Small number of generations	Large number of generations
Size of each generation of system log files	Becomes larger.	Becomes smaller.
Swap interval	Because the size of each generation of system log files becomes larger, the swap interval becomes longer.	Because the size of each generation of system log files becomes smaller, the swap interval becomes shorter.
Unload frequency	Because the swap interval becomes longer, the unload frequency becomes lower.	Because the swap interval becomes shorter, the unload frequency becomes higher.
Effects on the system log size when something such as a disk failure makes several generations of system log files unusable	<ul style="list-style-type: none"> Because the size of each generation of system log files becomes larger, the log volume used for database recovery in the event of a disk failure increases, and the time required for database recovery increases. If the decrease in the system log volume is large, the effects of the decrease in system log volume will have increasing effects on HiRDB operations. 	<ul style="list-style-type: none"> Because the size of each generation of system log files becomes smaller, the log volume used for database recovery in the event of a disk failure decreases, and the time required for database recovery decreases. If the decrease in the system log volume is small, the effects of the decrease in system log volume will have decreasing effects on HiRDB operations.

In normal operations, the lower the number of generations of system log files, the more advantageous the swapping interval and the unload frequency will become. However, if there is a failure, the effects on operations will be reduced with a larger number of log file generations.

(2) Design for improved reliability

(a) System log file duplexing

When system log file duplexing is used, HiRDB acquires the same system log information in both versions. In the event of an error on one of the versions, the system log can be read from the other version, thereby improving system reliability. When dual system log files are used, they must be used under the management of HiRDB rather than using a mirror disk. When using dual system log files, create the files for each system on a separate hard disk.

To use dual system log files, specify the following operands in the server definition:

- `pd_log_dual=Y`
- `-b` option in the `pdlogadpf` operand (to specify the name of the B version of system log file)

(b) Single operation of system log files

Single operation of system log files is employed when dual system log files are used.

In the event of an error in a system log file, processing can continue using the normal version of the system log file without having to terminate the HiRDB unit abnormally even if neither system has a usable system log file. This is called single operation of system log files. To perform single operation of system log files, specify `pd_log_singleoperation=Y` in the server definition.

As opposed to single operation of system log files, continuing processing using both versions of system log files (normal processing mode) is called double operation of system log files

(c) Automatic opening of system log files

If there is no overwrite-enabled system log file at the time of a HiRDB restart, but a reserved file is available, then HiRDB continues processing by opening the reserved file and placing it in overwrite-enabled status. This is called automatic opening of system log files.

To perform automatic opening of system log files, specify `pd_log_rerun_reserved_file_open=Y` in the server definition.

(3) Features to improve availability**(a) Facility for monitoring the free space remaining for system log files**

When system log files need to be swapped, if no swappable target system log files exist, HiRDB (the unit) terminates abnormally. To prevent this, HiRDB has a *facility for monitoring the free space remaining for system log files*. This facility operates when the percentage of available free area for the system log files reaches a warning value. Select one of the following two levels:

Level 1:

When the available space in the system log file falls below the warning level, the KFPS01162-W warning message is output.

Level 2:

When the available space in the system log file falls below the warning level, scheduling of new transactions is suppressed, and all transactions inside the server are forcibly terminated. At this time, message KFPS01160-E is output. This controls the output volume of the system logs.

If level 2 is selected, all transactions on the server are terminated forcibly when there is insufficient free space in the system log files. Therefore, the design of system log files requires more accuracy.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

(b) System log file automatic extension facility

The HiRDB system (or unit) terminates abnormally when it runs out of free space for system log files. To prevent this, HiRDB provides a facility to automatically expand the space for system log files (the system log file automatic extension facility). By using this facility, you can reduce the frequency of abnormal termination of the HiRDB system (or unit) due to lack of free space for system log files.

For details about the system log file automatic extension facility, see the *HiRDB Version 9 System Operation Guide*.

(c) Skipped effective synchronization point dump monitoring facility

If a UAP continues to update the database in an infinite loop, it cannot enable the synchronization point, and the number of system log files that cannot be overwritten increases. If the point is reached where none of the system log files can be overwritten, HiRDB terminates abnormally.

If HiRDB is forcibly terminated or terminates abnormally when the number of system log files that cannot be overwritten reaches one-half or more of all system log files, a shortage of system log files occurs during rollback processing when HiRDB restarts. In this case, HiRDB cannot be restarted unless new system log files are added. Any such restart processing will take longer than usual.

To prevent this, HiRDB has established a skipped effective synchronization point dump monitoring facility.

For details about this facility, see the *HiRDB Version 9 System Operation Guide*.

(4) Facility for parallel output of system logs (AIX and Linux versions only)

When the system log file is duplexed, using the facility for parallel output of system logs allows log output to the two systems to be executed in parallel, so less time is required. To use the facility for parallel output of system logs, you will need an aio library (for AIX, the Asynchronous I/O Subsystem; for Linux, libaio). For details about the Asynchronous I/O Subsystem or libaio, see the OS documentation.

Note that when you use the facility for parallel output of system logs on the Linux version, it is assumed that the platform you are using is one of the following:

- Red Hat Enterprise Linux ES 4(AMD64 & Intel EM64T) or later

- Red Hat Enterprise Linux AS 4(AMD64 & Intel EM64T) or later

(a) Recommended usage

Although you can define for each server whether the facility for parallel output of system logs is to be used, we recommend that you apply this facility to all servers. We also recommend that you place the primary and secondary files on separate devices in order to further reduce the time required for output of log information.

(b) Environment settings (setting system definitions)

In the server definition, specify `pd_log_dual_write_method=parallel`. In the following cases, the facility for parallel output of system logs is not applied, regardless of the specified value:

- Dual system log files are not used (the value of the `pd_log_dual` operand is not `Y`).
- The system log files are not placed in character special files.

(c) Environment settings (setting the aio library)

Install the aio library on all server machines that will use the facility for parallel output of system logs, and perform the required settings. See the OS documentation for details about how to install and set up the aio library.

If the aio library installation and settings are not performed correctly using the environment settings described in (b), HiRDB cannot start.

■ Using AIX 5L

Asynchronous I/O Subsystem must be enabled prior to starting HiRDB. Otherwise, HiRDB will not start. On AIX 5L V5.2 and later, Asynchronous I/O Subsystem comes in a legacy version and a POSIX version. HiRDB uses the legacy version, so enable the legacy Asynchronous I/O Subsystem.

Once you have installed Asynchronous I/O Subsystem, set the following parameters.

Parameter	Recommended value
STATE to be configured at system restart	available
STATE of FastPath	enable [#]

[#]: This is the default, so no change is needed. If the setting is changed, performance might decline.

Other parameters do not require tuning.

(d) Notes

1. If Asynchronous I/O Subsystem is not enabled on HiRDB for AIX 5L, HiRDB cannot start and will terminate abnormally. Either enable the Asynchronous I/O Subsystem, omit the `pd_log_dual_write_method` operand, or specify `serial`, and then restart HiRDB. If the Asynchronous I/O Subsystem parameter is set to `STATE to be configured at system restart = available`, HiRDB starts automatically.
2. The facility for parallel output of system logs is not applicable to system log files placed in regular files. If you add system log files, place them in character special files.
3. The facility for parallel output of system logs is applied only when both primary and secondary current files are placed in character special files and system log information can be output to those current files (they are not in closed, reserved, or error status). Parallel output of system logs does not take place, regardless of the system definition, if the current file satisfies either of the following conditions:
 - The primary or secondary current file is placed in a regular file.
 - The primary or secondary current file is in a status such that no log information can be output to it.
4. If the server that uses the facility for parallel output of system logs is running on multiple server machines, as when the system switchover facility is used, startup of the standby unit or system switchover might fail if the aio library is not enabled on any of the server machines. Enable the aio library on all server machines.

(5) Record length of a system log file (when the HiRDB system is already operating)

If the record length is not 1,024, we recommend that you change it to 1,024 to improve system log storage efficiency.

For details about how to change the system log file record size, see the *HiRDB Version 9 System Operation Guide*.

(6) Defining the system log files

The `pdlogadfg` and `pdlogadpf` operands are used to define the correspondence between file groups and the created system log files.

9.3.2 Designing synchronization point dump files

This section describes the design considerations for synchronization point dump files.

(1) Design considerations

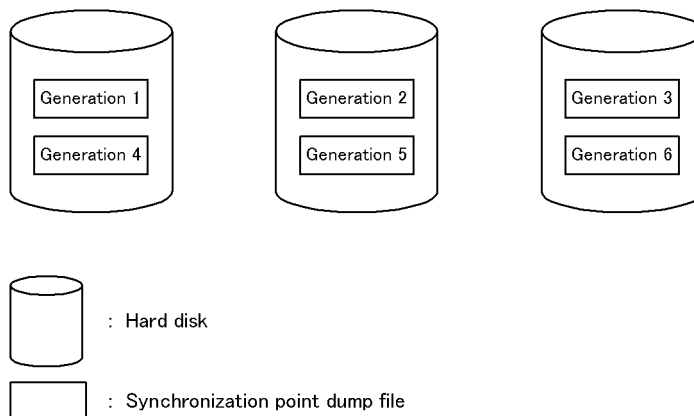
1. Synchronization point dump files are required for each server, except for the system manager.
2. You can create 2-60 groups of synchronization point dump files per server (if `ONL` is specified, 2-30 groups per server).
3. You should create at least four synchronization point dump files per server.
4. HiRDB uses synchronization point dump files in the order specified in the `pdlogadfg -d spd` operand.
5. If a shortage of space occurs in a synchronization point dump file, HiRDB cannot be restarted. For this reason, the size of a synchronization point dump file should be set to be greater than the value specified for the maximum number of concurrent connections (`pd_max_users`) in the system common definition. For details about the calculation of synchronization point dump file size, see *17.2 Determining the size of synchronization point dump files*.

(2) Design for improved reliability

(a) Example of file organization

As a safeguard against the possibility of hard disk failures, the synchronization point dump files should be created on separate hard disks. If this is not possible, adjacent generations of files should be created on separate hard disks, as shown in the example in the following figure.

Figure 9-5: Example of creating adjacent generations on separate hard disks (HiRDB parallel server configuration)



(b) Duplexing of the synchronization point dump file

When the synchronization point dump file is duplexed, HiRDB collects the same synchronization point dump on both system A and system B. This increases system reliability, because when a collected synchronization point dump is read and there is an abnormality in the file, the synchronization point dump can still be read from the other file. Duplexing also enables the number of guaranteed-valid generations to be set to one generation, yet reliability is not compromised and the number of synchronization point dump files in overwrite disabled status is reduced.

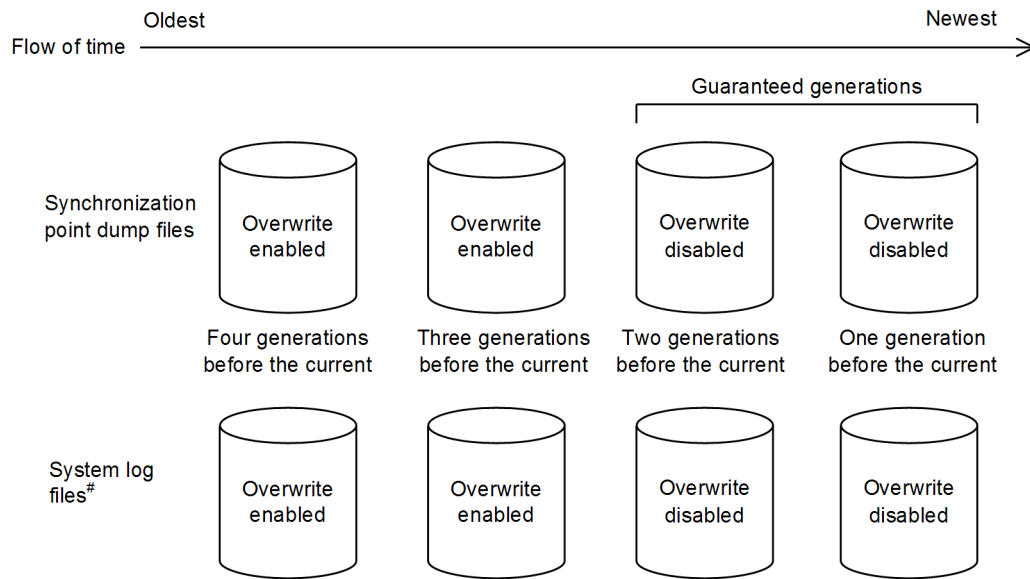
Specify the following operands in the server definition to enable duplexing of synchronization point dump files:

- `pd_spd_dual = Y`
- `-b` option in the `pdlogadpf` operand (specifies the system log file name on system B)

(c) Number of guaranteed-valid generations for synchronization point dump files

Each synchronization point dump acquired by HiRDB is stored in a separate synchronization point dump file. HiRDB uses the generation concept to manage synchronization point dump files. The HiRDB administrator specifies the number of generations of synchronization point dump files, and the corresponding system log files, that are to be placed in overwrite-disabled status. This concept is called the number of guaranteed-valid generations for synchronization point dump files, and it is illustrated in the following figure.

Figure 9–6: Number of guaranteed-valid generations for synchronization point dump file (HiRDB parallel server configuration)



#: These are the system log files that correspond to the synchronization point dump files.

Explanation

If there are two guaranteed-valid generations, the synchronization point dump files up to the second generation, and the system log files relevant to those synchronization point dump files, are in overwrite disabled status. The synchronization point dump files prior to the third generation, and the system log files relevant to those synchronization point dump files, are in overwrite enabled status.

The required number of synchronization point dump files is the number of guaranteed-valid generations + 1. Specify the number of guaranteed-valid generations for synchronization point dump files in the `pd_spd_assurance_count` operand in the server definition.

If synchronization point dump files are to be duplexed, it is recommended that only one guaranteed-valid generation be required. If duplexing is not to be used, two generations are recommended.

(d) Reduced mode operation for synchronization point dump files

If the number of synchronization point dump files available for use is reduced to the number of guaranteed-valid generations + 1 because of errors in synchronization point dump files, processing can continue with a minimum of two files. This is called the reduced mode operation for synchronization point dump files.

To perform reduced mode operation for synchronization point dump files, specify the `pd_spd_reduced_mode` operand in the server definition.

(e) Automatic opening of synchronization point dump files

When the number of synchronization point dump files available for use is reduced to the number of guaranteed-valid generations + 1 because of errors in synchronization point dump files, processing can continue by opening a reserved file and placing it in overwrite-enabled status (assuming that such a reserved file is available). This is called automatic opening of synchronization point dump files.

To perform automatic opening of synchronization point dump files, specify the `pd_spd_reserved_file_auto_open=Y` in the server definition.

(3) Defining the synchronization point dump files

The `pdlogadfg` and `pdlogadpf` operands are used to define the correspondence between file groups and the created synchronization point dump files.

If only the `pdlogadfg` operand is specified, synchronization point dump files can be added during HiRDB operation.

9.3.3 Designing status files

This section describes the design considerations for status files.

(1) Design considerations

1. Create the primary and secondary files on separate disks in order to avoid errors on both files.
2. To prevent abnormal termination of HiRDB as a result of a shortage of status file space, create several spare files whose size is greater than the estimated value. When a status file becomes full, file swapping occurs in order to use a spare file. If the size of the spare file is the same as the full status file, a space shortage also occurs on the spare file, resulting abnormal termination of HiRDB. For example, if you create six sets of status files, we recommend that you make the file size for two of the sets larger than the other sets.
3. Unit status files are required for each server machine.
4. Server status files are required for all servers except for the system manager.
5. Make sure that the primary and secondary files have the same record length and the same number of records.
6. You can create 1-7 sets of unit status files per unit.
7. You can create 1-7 sets of server status files per server.

(2) Design for improved reliability

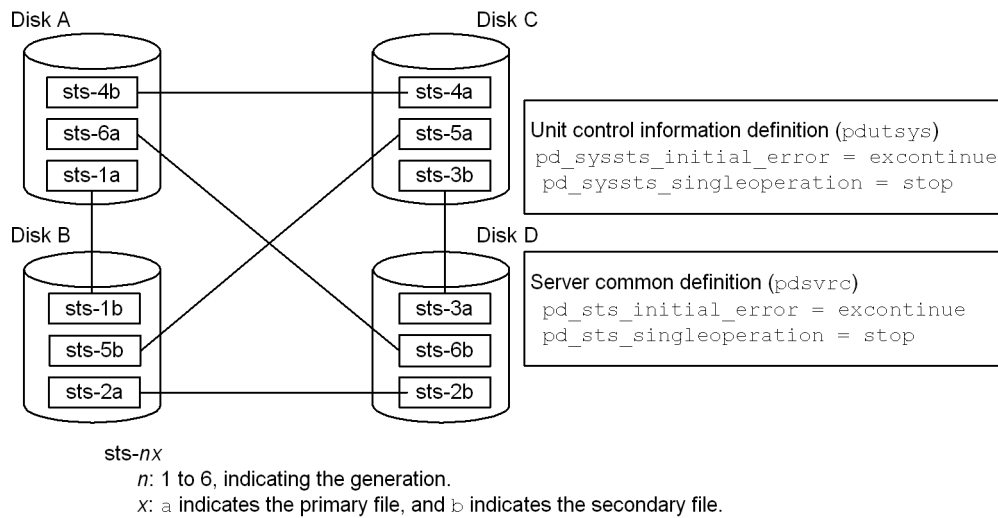
1. Provide at least three sets of status files (*dual files* `zueng005.tif` 3 = 6 files) and place them in such a manner that corruption of all status files by disk errors is unlikely.
2. To prevent abnormal termination of HiRDB resulting from a space shortage, we recommend that you set the size of a status file to at least 1.2 times the estimated value.
3. A status file contains information that will be needed in order to restore the system status during HiRDB restart processing. If an error occurs in such a file and no spare file is available, the system status cannot be restored. Therefore, make sure that spare files are always available as a safeguard in the event of errors on the current files.

(a) Recommended configuration

In order to provide a safety margin until a disk becomes operational after it has been recovered from a disk failure, we recommend that you provide six sets of status files on four sets of disks (*dual files* `zueng005.tif` 6 = 12 files) and place them as shown in the following figure. If an error occurs on the normal system during single operation, HiRDB cannot be restarted; therefore, we recommend that you do not apply single operation to status files (specify `stop` in `pd_syssts_singleoperation` and `pd_sts_singleoperation`).

The following figure shows an example of placing six sets of status files on four sets of disks.

Figure 9–7: Example of placing six sets of status files on four sets of disks

**Explanation:**

With this arrangement, if an error occurs on a disk and then another error occurs on another disk, the remaining two disks still contain intact primary and secondary files, and HiRDB can continue operation using the status files on the error-free disks as the current files. For example, if an error occurs on disk A and then an error occurs on disk B, HiRDB continues operation using as the current files the primary and secondary status files on disks C and D (sts-3a and sts-3b). In this status, if another error occurs on one of the current files, HiRDB terminates abnormally; however, because one of the current files is normal, HiRDB can be restarted after one of the disks is recovered from its error.

(3) Defining the status files

The `pd_syssts_file_name_1` to `pd_syssts_file_name_7` and the `pd_sts_file_name_1` to `pd_sts_file_name_7` operands are used to define the correspondence between the status files created by the `pdtsinit` command and the logical files.

The `pd_syssts_file_name_1` to `pd_syssts_file_name_7` operands are for unit status files, and the `pd_sts_file_name_1` to `pd_sts_file_name_7` operands are for server status files.

If the names of imaginary logical files or status files are defined in the `pd_syssts_file_name_2` to `pd_syssts_file_name_7` operands or in the `pd_sts_file_name_2` to `pd_sts_file_name_7` operands, status files can be added during HiRDB operation. In this case, the following operands must be specified.

Unit status files

```
pd_syssts_initial_error
pd_syssts_last_active_file
```

Server status files

```
pd_sts_initial_error
pd_sts_last_active_file
```

(4) Single operation of status files

If an error occurs on one of the current files while there is no available spare file, continuing operation using only the normal file (either the primary or secondary file) is called *status-file single operation*. When status files are placed in the single operation mode, the `KFPS01044-I` message is displayed.

If an error occurs on the current file in the single operation mode, HiRDB can no longer be restarted. Therefore, status-file single operation is not recommended. Increase the number of status file sets to avoid a situation where no spare file is available.

As opposed to status-file single operation, continuing operation using both status files (normal processing mode) is called *status-file double operation*.

(a) Advantages and disadvantages of status-file single operation

Advantages

Processing can continue even if an error occurs on one of the current files while no spare file is available. This reduces the adverse consequences of HiRDB shutdown resulting from a status file error.

Disadvantages

If an error occurs on the normal file during single operation or HiRDB terminates abnormally while the status file is updated, the contents of the current file are lost, disabling HiRDB restart.

(b) Specification method

To use unit status-file single operation, specify `pd_syssts_singleoperation = continue` in the unit control information definition file. To use server status-file single operation, specify `pd_sts_singleoperation = continue` in the server definition. Make sure that `pd_syssts_singleoperation` and `pd_sts_singleoperation` have the same value.

- Relationship with other operands

The combination of the `pd_syssts_singleoperation` and `pd_syssts_initial_error` operand values or the `pd_sts_singleoperation` and `pd_sts_initial_error` operand values determines the HiRDB operation that is to take place if an error is detected in a status file during HiRDB startup. Therefore, determine the values of these operands together. For details about the HiRDB operation that is to take place if an error is detected in a status file during HiRDB startup, see the description of the `pd_syssts_initial_error` or `pd_sts_initial_error` operand in the manual *HiRDB Version 9 System Definition*.

(c) Usage guidelines

The following are guidelines on when to use status-file single operation.

- Do not use single operation if the primary goal is to avoid HiRDB being unable to restart.
- Use single operation if the primary goal is to avoid disabling the ability of HiRDB to go online.
- Do not use single operation if you have set HiRDB to restart automatically, such as at the time of a system switchover.

(d) Notes about using single operation

The following table outlines HiRDB operations and HiRDB administrator actions that depend on whether single operation is used. For details about how to handle status file errors, see the *HiRDB Version 9 System Operation Guide*.

Table 9–7: HiRDB operation and HiRDB administrator's action that depend on whether single operation is used

Condition		Status-file single operation (<code>pd_syssts_singleoperation</code> or <code>pd_sts_singleoperation</code> operand value)	
		Used (continue specified)	Not used (omitted or stop specified)
There are spare files	Error occurred in the current file	HiRDB operation: Swaps status files. HiRDB administrator's action: Handle the error in the applicable status file.	
	Error occurred on both current files simultaneously	HiRDB operation: Terminates abnormally. HiRDB cannot be restarted. HiRDB administrator's action: See <i>Handling of status file errors</i> in the <i>HiRDB Version 9 System Operation Guide</i> .	
There is no spare file	Error occurred in one of the current files	HiRDB operation: Resumes processing using single operation.	HiRDB operation: Terminates abnormally.

Condition		Status-file single operation (pd_syssts_singleoperation or pd_sts_singleoperation operand value)	
		Used (continue specified)	Not used (omitted or stop specified)
		HiRDB administrator's action: Create spare files immediately and return HiRDB to the double operation mode.	HiRDB administrator's action: Create spare files, and then restart HiRDB.
	Error occurred on both current files simultaneously	HiRDB operation: Terminates abnormally. HiRDB cannot be restarted. HiRDB administrator's action: See <i>Handling of status file errors</i> in the <i>HiRDB Version 9 System Operation Guide</i> .	
	Error occurred in the normal file during single operation	HiRDB operation: Terminates abnormally. HiRDB cannot be restarted. HiRDB administrator's action: See <i>Handling of status file errors</i> in the <i>HiRDB Version 9 System Operation Guide</i> .	--

Legend:

--: Not applicable

(5) Notes on status file errors (important)

- If errors occur on both current files simultaneously, HiRDB terminates abnormally and HiRDB can no longer be restarted. A possible measure for avoiding this situation is to use multiple physical disks (mirroring).
- If the current file (existing during termination) is deleted or initialized by the `pdstsinit` command prior to HiRDB startup, HiRDB can no longer be restarted.

9.4 Placing RDAREAs

This section discusses considerations concerning placement of the following types of RDAREAs:

- System RDAREAs
- Data dictionary LOB RDAREAs
- User RDAREAs
- User LOB RDAREAs
- List RDAREAs

9.4.1 Placing system RDAREAs

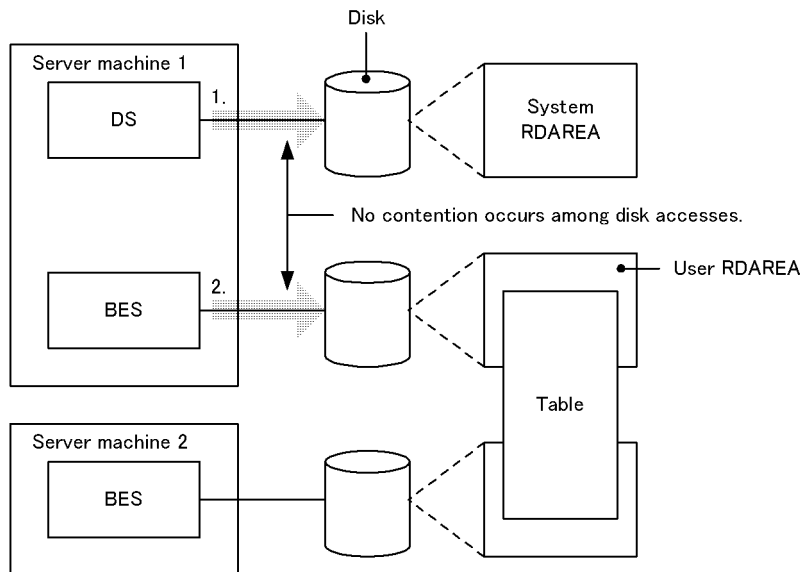
System RDAREAs should be placed taking into account the placement of user RDAREAs. Points to be considered when a system RDAREA is placed are discussed below.

- **Place system RDAREAs on the dictionary server.**
- **If both dictionary server and back-end server are located on the same server machine, place system RDAREA areas on a separate disk from that for user RDAREAs.**

Among the system RDAREAs, data dictionary RDAREAs and data directory RDAREAs are accessed frequently by HiRDB for SQL statement analysis. If they are placed on the same disk as user RDAREAs, contention may occur between an access request for the purpose of SQL statement analysis and a table access request, in which case one of the requests is placed on hold until the other request has been processed.

The following figure shows an example of system RDAREA placement that does not generate disk access contention.

Figure 9–8: Example of system RDAREA placement (HiRDB parallel server configuration)



DS: Dictionary server

BES: Back-end server

→ : Disk access

Explanation:

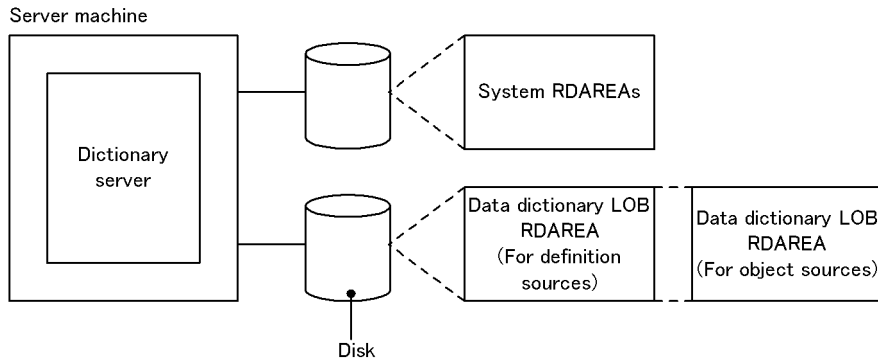
1. Access for the purpose of analyzing SQL statements, etc.
2. Access to the table

9.4.2 Placing data dictionary LOB RDAREAs

To avoid contention among disk accesses, a data dictionary LOB RDAREA should not be placed on the same disk as any other RDAREA.

The following figure shows an example of data dictionary LOB RDAREA placement.

Figure 9–9: Example of data dictionary LOB RDAREA placement (HiRDB parallel server configuration)



Relationship with data dictionary RDAREAs

A dictionary table used to manage stored procedures or stored functions can be placed in a separate data dictionary RDAREA from other dictionary tables.

9.4.3 Placing user RDAREAs

(1) Relationship with system log files

A user RDAREA should not be placed on the same disk as a system log file. When this rule is observed, input/output operations on HiRDB files that constitute system log files and on user RDAREAs can be distributed to multiple disks when a synchronization point dump is collected, thereby reducing the amount of time required for synchronization point dump processing.

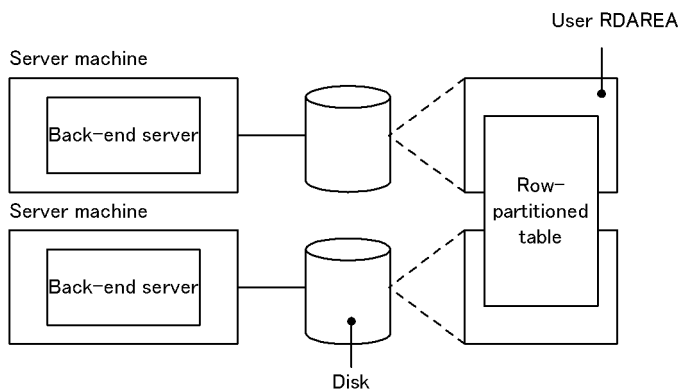
(2) Relationship with system RDAREAs

A user RDAREA should not be placed on the same disk as a system RDAREA.

(3) Row-partitioned tables

If you have partitioned a table by row, place the RDAREAs storing the row-partitioned table on separate back-end servers and on separate disks. The following figure shows an example of user RDAREA placement.

Figure 9–10: Example of user RDAREA placement (HiRDB parallel server configuration)



(4) Placement of a floating server

If you perform complicated query processing on tables, such as join and sort processing involving multiple back-end servers, carefully determine the placement of user RDAREAs.

If you place user RDAREAs on all back-end servers, some back-end servers' workloads become high because they not only access user RDAREAs but also execute complicated query processing. This results in reduction of overall system throughput.

If you have a sufficient number of server machines, define a back-end server that has no user RDAREA placed on it (floating server). In this way, complicated query processing is handled by the floating server, thereby reducing each back-end server's workload.

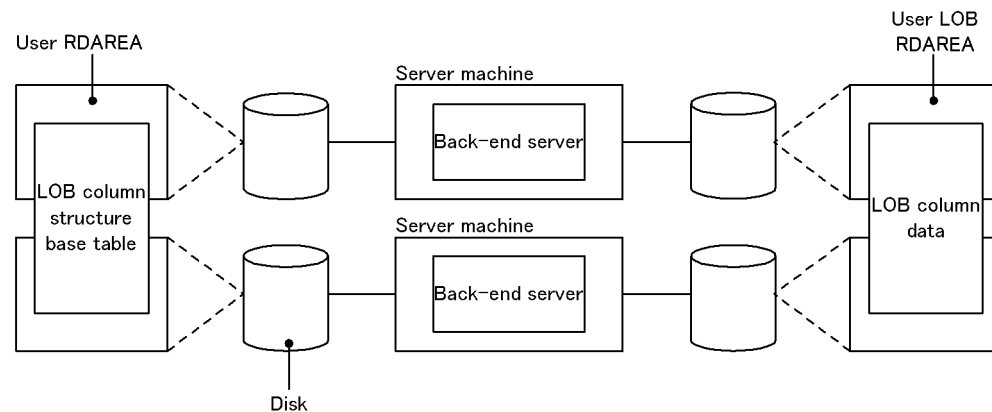
9.4.4 Placing user LOB RDAREAs

To avoid contention among disk accesses, a user LOB RDAREA should not be placed on the same disk as any other RDAREA.

In the case of a table containing a LOB column in a HiRDB parallel server configuration, the user LOB RDAREAs containing the LOB data and the user RDAREAs containing the LOB column structure base table must be placed on the same back-end server.

The following figure shows an example of user LOB RDAREA placement.

Figure 9–11: Example of user LOB RDAREA and user RDAREA placement (HiRDB parallel server configuration)



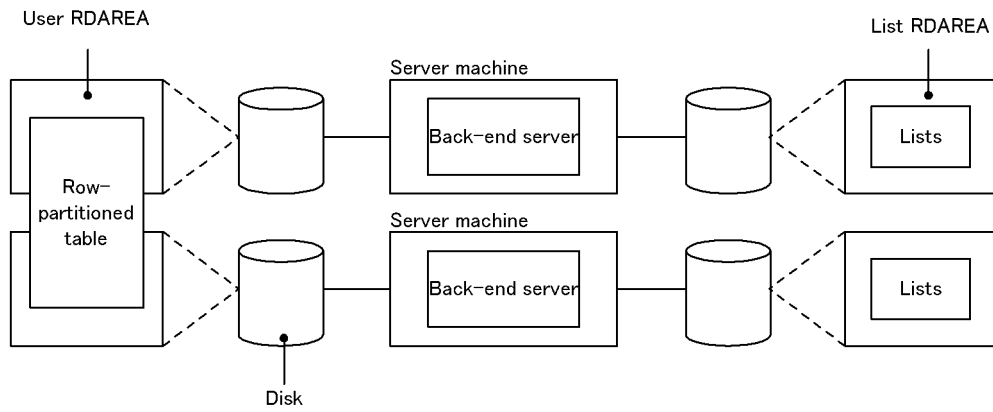
9.4.5 Placing list RDAREAs

Place list RDAREAs on the back-end server that contains its base table.

Creating one list RDAREA lets you create lists for all the tables that are stored in that back-end server.

To avoid contention among disk accesses, you should place list RDAREAs on a separate disk from any other RDAREAs. The following figure shows an example of list RDAREA placement.

Figure 9–12: Example of list RDAREA placement (HiRDB parallel server configuration)



9.5 Considerations that apply to building a system with many units or servers

This section provides information that you need to take into account if you build and operate a system with many units or servers. In general, you should read this section if you are building and operating a system that has 10 or more units, or 10 or more servers.

Here, the term *servers* refers to front-end servers, dictionary servers, and back-end servers. Read this section if you have ten or more of such servers.

9.5.1 Considerations that apply to configuring systems

(1) Setting system definitions

When you configure a system with many units or servers, you must reduce the communications load by locking the ports that HiRDB uses. To do this, specify the system definition operands shown in the table below.

Table 9–8: Operands that must be specified to reduce the communications load

No.	Operand	Description
1	<code>pd_name_fixed_port_lookup=Y</code>	Specify <code>Y</code> for this operand and use the unit's own shared memory information so that it communicates with other units. Also specify the operands shown in No. 2.
2	<ul style="list-style-type: none"> • <code>pd_mlg_port</code> or the <code>pdunit -m</code> option • <code>pd_alv_port</code> or the <code>pdunit -a</code> option • <code>pd_trn_port</code> or the <code>pdunit -t</code> option • <code>pd_scd_port</code> or the <code>pdunit -s</code> option • <code>pd_name_port</code> or the <code>pdunit -p</code> option 	--
3	<code>pd_ipc_conn_nblock_time</code>	<p>When HiRDB is performing communication between servers and there is a server that is not running, the next operation is pended by the amount of time specified in this operand. For that reason, specifying too high a value for this operand might cause performance to decline.</p> <p>In systems without a high network load, specify 2 (seconds) for this operand.</p> <p>In systems with a high network load, do one of the following.</p> <ul style="list-style-type: none"> • Determine the value for this operand according to the network environment. • Consider modifying or upgrading your network environment.
4	<code>pd_bes_connection_hold=Y</code>	No particular specification.
5	<code>pd_bes_conn_hold_trn_interval</code>	If the UAP connection time (from <code>SQL CONNECT</code> to <code>DISCONNECT</code>) is short, specify 0 for this operand.

Legend:

--: Not applicable

(2) Setting the high-speed connection facility

Use the high-speed connection facility to reduce the communication load. For details about the high-speed connection facility, see the *HiRDB Version 9 UAP Development Guide*.

(3) Setting the standby-less system switchover (effects distributed) facility

The maximum number of units that can be defined for a single HA group is 32. So if the number of units is 33 or more, define multiple HA groups. For details about the standby-less system switchover (effects distributed) facility, see the *HiRDB Version 9 System Operation Guide*.

(4) Setting the number of open files

When you have high numbers of units or servers, you must also set an upper limit for the number of open files. Determine a value for the operating system parameter (for HP-UX, `maxfiles_lim`; for Solaris, `rlim_fd_max`; for AIX, `nofiles_hard`; and for Linux, `hard nofile`) that sets the physical limit value for the number of files that can be opened or locked by a single process. For details about the operating system parameters, see *Chapter 20. Specifying OS Parameters*.

(5) Settings to prevent Listen queue shortages

The TCP/IP Listen queue used during communications might run short. To prevent this, make sure that the Listen queue specification is not too small. For details about how to specify the Listen queue, see *20.6 Listen queue specified values*.

(6) Settings to prevent a shortage of ports

The number of ports used during communications might run short. To prevent this, you need to take steps to keep a port shortage from developing. For details about how to configure these settings, see *22.4.4 Ways to avoid a shortage of ports*.

9.5.2 Considerations for system operation

When a transaction or command (including utilities) is executed, one or more of the following phenomena might occur, causing communication errors. This might result in a transaction or command error, or a unit abnormality might be detected by host-to-host monitoring (the `KFPS05289-E` message is output).

- There are not enough ports for the remote shell to use
When a HiRDB command is executed, internally a remote shell is executed. For this reason, when the command is being executed on a large number of units or servers, there might not be enough ports for the remote shell to use.
- There are not enough ports for the system to use
When a system has a large number of units or servers, the number of HiRDB server-to-server communication connections increases, and there might not be enough ports for the system to use.
- There is not enough network area
When a system has a large number of units or servers, the number of HiRDB server-to-server communication connections increases, and there might not be enough network area.

Should such phenomena occur, reduce the communications load using the following procedures as applicable.

(1) Specify a unit name when you execute a command

When you execute any of the following commands, specify a unit name. This makes the command execute for individual units, so fewer ports are used.

- `pdaudbegin`, `pdaudend`, `pdaudrm`, `pdaudswap`
- `pdcancel`
- `pdcat`
- `pdrisechk`
- `pdstscls`, `pdstsin`, `pdstsope`, `pdstsr`, `pdstsswap`
- `pdstbegin`, `pdstend`
- `pdcm`, `pdrbk`, `pdfgt`

- `pdls` (a server name can be specified)

However, ports can temporarily run short even when these commands are executed for individual units, if other units generate processing requests while processing of the command is extended. Should this happen, wait a moment and then re-execute the command.

(2) Specify a server name when you execute a command

When you execute any of the following commands, specify a server name. This makes the command execute for individual servers, so fewer ports are used.

- `pdchprc`
- `pdclttc`
- `pdjarsync`
- `pdlogadpf`, `pdlogchg`, `pdlogcls`, `pdloginit`, `pdlogls`, `pdlogopen`, `pdlogrm`, `pdlogswap`, `pdlogsync`, `pdlogunld`, `pdlogatul`
- `pdobils`
- `pdpfresh`

However, ports can temporarily run short even when these commands are executed for individual servers, if other servers generate processing requests while processing of the command is extended. Should this happen, wait a moment and then re-execute the command.

(3) Reduce the communications load when you execute the `pddbstd` command

When you execute the `pddbstd` command, execute status analysis in RDAREA units.

(4) Reduce the communications load when you execute the `pdload` command

When you execute a `pdload` command, create input files for each partition storage condition to reduce the communications load when the `pdload` command is executed in RDAREA units.

You can also reduce the communications load by placing the multiple input files that have been created not in one place (on the same machine), but on a server machine that has a table storage RDAREA.

(5) Reduce the communications load when you execute the `pdrorg` command

When you reorganize a table, unload a table or reload a table using the `pdrorg` command, execute the command in RDAREA or server units to reduce the communications load.

When you re-create indexes, reorganize indexes, or batch create indexes using the `pdrorg` command, execute the command in index or server units to reduce the communications load.

(6) Reduce the communications load when you execute the `pdcopy` or `pdrstr` command

When you execute the `pdcopy` or `pdrstr` command, reduce the communications load as follows.

- Specify a single server name in the `-s` option, and then execute the command.
- If you use the `-r` option to specify multiple RDAREAs, specify only RDAREAs on the same back-end server when you execute the command.

You can also reduce the communications load by placing backup files on the server machine that processes commands.

(7) Reduce the communications load when you execute SQL statements

Since accessing data on multiple back-end servers through a single transaction generates data communication between servers, you need to reduce data communication routes between servers as much as possible. Data on multiple back-end servers is accessed when any one of the following conditions is met.

- An SQL statement that specifies row-partitioned tables in multiple back-end servers is called.
- An SQL statement that specifies two or more table in a FROM clause is called.
- An SQL statement that specifies a subquery is called.
- An SQL statement that specifies a set operation is called.
- An SQL statement that updates shared table data is called.
- When multiple SQL statements are called within a single transaction, tables defined in different back-end servers are specified in the FROM clauses of the respective SQLs.

When any one of the above conditions is met, you can reduce the data communication routes between servers by taking the following actions.

- Reduce the number of partitions in row-partitioned tables.
- Specify conditions for partition keys in the SQL search conditions.
- For table joining, align the partitions of the tables in question and make the joining key the partition key.
- Store the data of the various tables that the transaction accesses on the same back-end server.

(8) Reduce the communications load when you use floating servers

The communications load increases when floating servers are used, so avoid floating servers whenever possible. Note that floating servers are used when any one of the following conditions is met:

- An SQL statement that specifies two or more tables in a FROM clause (except when a nested loop join is used as a joining method) is called.
- An SQL statement that specifies a subquery is called.
- An SQL statement that specifies a set operation is called.
- An SQL statement that specifies an ORDER BY clause is called (except when the sort order of a column contained in ORDER BY can be guaranteed by searching an index, even without sort processing for the ORDER BY clause).
- An SQL statement that specifies a GROUP BY clause is called.
- An SQL statement that specifies DISTINCT is called.
- An SQL statement that specifies a derived table in a view table, WITH clause, or FROM clause is called (except when an internal derived table is not created by the SQL that specifies a view table or WITH clause).

For details about conditions that create internal derived tables, see the manual *HiRDB Version 9 SQL Reference*.

- An SQL statement that specifies a FOR READ ONLY clause is called.

When any one of the above conditions is met, you can reduce the number of floating servers used by taking the following actions.

- Specify FLTS_ONLY_DATA_BES in the SQL optimization option.
- Specify SORT_DATA_BES in the SQL optimization option.

For details about SQL optimization options, see the *HiRDB Version 9 UAP Development Guide*.

(9) Reduce the communications load when you use shared tables

When shared tables are updated through multiple front-end servers, the communications load increases. For this reason, when you update multiple shared tables, connect the HiRDB client to the same front-end server whenever possible.

(10) Notes on simultaneous execution of utilities

When a system has many back-end servers, utilities might terminate abnormally when many of them are executed simultaneously. Should this occur, take a corrective action such as reducing the number of utilities that execute simultaneously.

(11) Corrective action to take when the connected user data file is not output

When the KFPS05120-W message is output, the connected user data file might not be output under \$PDDIR/spool/cnctusrinf. In these cases, execute the `pdls -d act`, `pdls -d prc`, and `pdls -d trn` commands for units individually.

For details about the connected user data file, see the *HiRDB Version 9 System Operation Guide*.

(12) Limitations

- When there are 65 or more units, the facility for monitoring MIB performance information cannot be used.
- When the limit for the number of ports that the remote shell can use is exceeded, the following commands cannot be executed.
 - `pdconfchk`
 - `pdls -d rpc`

9.5.3 Corrective actions to take in response to errors that occur when commands are executed

The following table shows corrective actions to take in response to errors that occur when commands are executed.

Table 9–9: Corrective actions to take in response to errors that occur when commands are executed

Executed command	Type of error and its corrective action
<code>pdchgconf</code>	The <code>pdchgconf</code> command cannot be executed when the number of ports that the remote shell can use exceeds its limit. If this occurs, terminate HiRDB normally using the <code>pdstop</code> command, and then modify the configuration.
<code>pdtrndec</code>	The <code>pdtrndec</code> command cannot be executed when the number of ports that the remote shell can use exceeds its limit. If this occurs, manually resolve uncompleted transactions. For details about how to manually resolve uncompleted transactions, see <i>Actions when there is an undetermined transaction</i> in the <i>HiRDB Version 9 System Operation Guide</i> .
<code>pdprgcopy</code> or <code>pdprgrenew</code>	The <code>pdprgcopy</code> and <code>pdprgrenew</code> commands cannot be executed when the number of ports that the remote shell can use exceeds its limit. If this occurs, install updates to replace HiRDB on the server machines on which it is installed.
<code>pdstart</code>	The <code>pdstart</code> command cannot be executed when the number of ports that the remote shell can use exceeds its limit. If this occurs, execute the <code>pdstart -q</code> command to start HiRDB on each of the server machines that comprise a HiRDB parallel server configuration.
<code>pdorend</code>	When you reflect online reorganization with the <code>pdorend</code> command, the communications load can become large if there are many servers, and an error can result if the maximum wait time for reflection processing specified in the <code>-w</code> option is exceeded. If this happens, take the following corrective actions. <ul style="list-style-type: none"> • Specify a larger value in the <code>-w</code> option. • Make sure the number of online job transactions is low while the <code>pdorend</code> command is executing.

10

Designing a Multi-HiRDB

This chapter describes the system design considerations for a multi-HiRDB.

10.1 System design for a multi-HiRDB

This section describes only those design procedures for a multi-HiRDB that differ from an ordinary HiRDB.

10.1.1 Installing a multi-HiRDB

This section describes the points to be noted when installing a multi-HiRDB.

(1) Registering the HiRDB administrator

A different HiRDB administrator must be registered for each HiRDB. For details about registering a HiRDB administrator, see *2.1.2 Registering the HiRDB administrator*.

(2) Creating the HiRDB directories

A different HiRDB directory must be created for each HiRDB. For details about how to create a HiRDB directory, see *2.3.1 Creating the HiRDB directory*.

10.1.2 Setting the environment for a multi-HiRDB

(1) Setting environment variables

Each HiRDB administrator separately defined with a multi-HiRDB server uses the `PDDIR` environment variable to identify his/her own HiRDB. Specify the HiRDB directory in the `PDDIR` environment variable for each HiRDB administrator.

If you specify `$PDDIR/bin` in the `PATH` environment variable for each HiRDB, only the previously specified HiRDB operation commands in `PATH` become available. To operate each HiRDB individually, you should provide a window for each HiRDB and define the environment variable for each window.

For details about the environment variables, see *2.3.4 Setting environment variables*.

(2) Specifying HiRDB system definitions

Create a HiRDB system definition for each HiRDB. Specify the following information appropriately to each HiRDB in the HiRDB system definition:

- HiRDB identifier (`pd_system_id` operand in the system common definition)
- HiRDB port number (`pd_name_port` operand in the system common definition)
- Unit identifier (`pd_unit_id` operand in the unit control information definition)

(3) Specifying client environment definitions

Use the `PDNAMEPORT` operand of the client environment definition to specify a HiRDB to be accessed from a client. Specify the port number of a HiRDB to be accessed in the `PDNAMEPORT` operand. For details about the client environment definition, see the *HiRDB Version 9 UAP Development Guide*.

(4) Installing utility special units

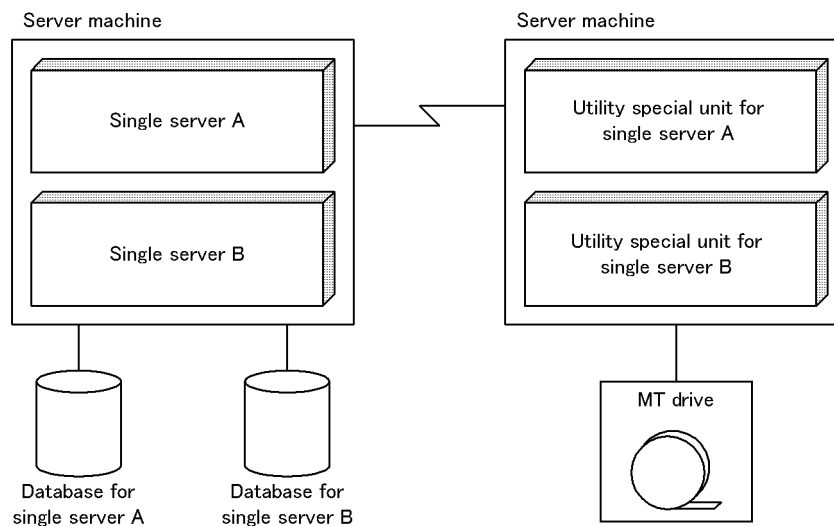
When the utility special unit facility is to be provided for a multi-HiRDB, one of the following system configurations can be selected:

1. System configuration in which one utility special unit is installed for each HiRDB.
2. System configuration in which one utility special unit is shared among multiple HiRDBs.

Configuration 1 should be used if different applications are to be executed by the individual HiRDBs. Configuration 2 should be used when the system switching facility is to be used in a mutual system switching configuration.

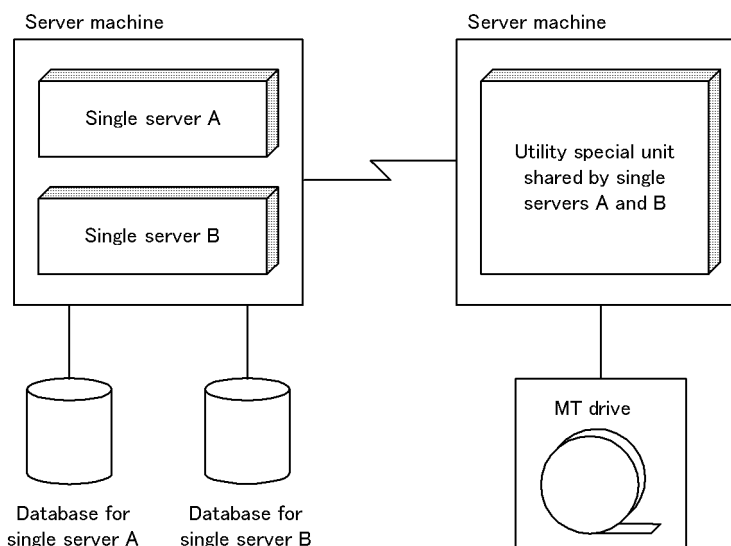
Figure 10-1 Example of utility special unit installation for a multi-HiRDB system: One utility special unit provided for each HiRDB system and Figure 10-2 Example of utility special unit installation for a multi-HiRDB system: One utility special unit shared among multiple HiRDB systems show examples of utility special unit installation for a multi-HiRDB.

Figure 10–1: Example of utility special unit installation for a multi-HiRDB system: One utility special unit provided for each HiRDB system



Note: One utility special unit is installed for each HiRDB.

Figure 10–2: Example of utility special unit installation for a multi-HiRDB system: One utility special unit shared among multiple HiRDB system



(5) Library sharing

All HiRDBs constituting the multi-HiRDB can share a portion of a library in the installation directory (`/opt/HiRDB_S` or `/opt/HiRDB_P`). You can save memory by using the `pdmemsv -s` command to share a portion of the library.

To share a portion of a library, the following conditions must be satisfied:

- All HiRDBs constituting the multi-HiRDB must be the same version.
- All HiRDBs constituting the multi-HiRDB use the same character encoding.
- All HiRDBs constituting the multi-HiRDB use the same load type (the POSIX library must be used with all HiRDBs).

If you share a library, place the installation directory and the HiRDB directory on separate volumes. If you do this and a hard disk failure affects the installation directory, you can use the `pdmemsv -d` command to cancel library sharing and enable HiRDB operation. The following shows the procedure for sharing a library:

```
# pdsetup /USERS/DB1          1
# pdsetup -d /USERS/DB1       2
# PDDIR="/USERS/DB1 "        3
# export PDDIR
# $PDDIR/bin/pdmemsv -s       4
# pdsetup /USERS/DB1          5
```

Explanation:

This procedure must be executed by the superuser.

1. Sets up the HiRDB environment in a directory other than the installation directory.
2. Releases the HiRDB registration from the OS. When the KFPS0036-Q message is displayed, enter `n` as the response.
3. Defines the `PDDIR` environment variable.
4. Sharing the library.
5. Registers the HiRDB into the OS again.

Note

When reinstalling HiRDB, use the `pdmemsv -d` command to release library sharing. After reinstalling, re-execute the `pdmemsv -s` command to share libraries.

When you execute the `pdsetup -d` command, if you respond with `y` to message KFPS00036-Q and cancel registration with the OS, re-execute the `pdmemsv` command using the procedure described above.

10.2 Notes about upgrading

This section describes the points to be noted when upgrading HiRDB under a multi-HiRDB environment.

(1) Upgrading all HiRDBs at the same time

The following shows an overview of the procedure for upgrading all HiRDBs at the same time in a multi-HiRDB environment. For details about how to upgrade, see *1.4 Upgrading HiRDB*.

To upgrade all HiRDBs:

1. Delete all of the existing versions of HiRDB from the OS using the `pdsetup -d` command. Respond with `y` to the KFPS00036-Q message (if you respond with `n`, the original environment is kept unchanged, so HiRDB is not replaced by the new version even if the new version of HiRDB is installed and set up).
2. Install the new version(s) of HiRDB.
3. Register all of the new version of HiRDB in the OS using the `pdsetup` command.

(2) Upgrading only some of the HiRDBs

The following shows an overview of the procedure for upgrading only some of the HiRDBs in a multi-HiRDB environment. For details about how to upgrade, see *1.4 Upgrading HiRDB*.

To upgrade only some of the HiRDBs:

1. Delete the HiRDBs being upgraded from the OS using the `pdsetup -d` command. Respond with `y` to the KFPS00036-Q message (if you respond with `n`, the original environment is kept unchanged, so HiRDB is not replaced by the new version even if the new version of HiRDB is installed and set up).
2. Install the new version of HiRDB.
3. Register the upgraded HiRDBs in the OS using the `pdsetup` command.

(3) Using JP1/Software Distribution

If you are using JP1/Software Distribution to distribute HiRDB online, HiRDB cannot be registered automatically in the OS (execution of the `pdsetup` command) after HiRDB is distributed.

11

Designing Global Buffers and Local Buffers

This chapter describes global buffer and local buffer design.

11.1 Allocating global buffers

Global buffers refers to a group of buffers allocated in shared memory and used to read and write data stored in the RDAREAs on a disk. A buffer that is designed to hold data that has been updated but not yet written to a database is called an *update buffer*. A buffer that is designed for referencing data or that holds data that has already been written to a database is called a *reference buffer*.

Global buffers must be allocated for RDAREAs that store data and indexes. There are three types of global buffers:

- Index global buffers
- Data global buffers
- LOB global buffers

Addition, modification, or deletion of global buffers while HiRDB is operating is called *dynamic updating of global buffers*. The `pdbufmod` command is used for dynamic updating. For details about dynamic updating of global buffers, see the *HiRDB Version 9 System Operation Guide*.

This section describes the various methods of allocating global buffers.

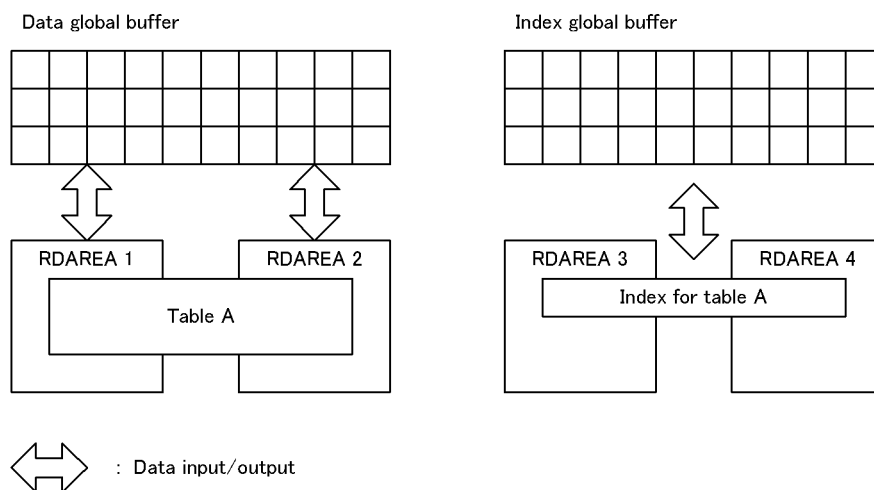
11.1.1 Allocating index global buffers

A dedicated global buffer should be allocated to an index that is accessed frequently, especially an index for which a cluster key or `UNIQUE` is specified. This enables the index to be made resident, thereby reducing the number of input/output operations required to access the index.

A dedicated global buffer that is allocated to an index is managed independently of the global buffer for the user RDAREAs that contain the table rows. This means that index pages and data pages are not shared within a global buffer. If the same global buffer is allocated to more than one index or table, information for one index may be swapped out of the global buffer in the event a large amount of data for another table is placed in it temporarily.

The following figure shows an overview of a dedicated global buffer for an index.

Figure 11–1: Overview of global buffer for an index



11.1.2 Allocating data global buffers

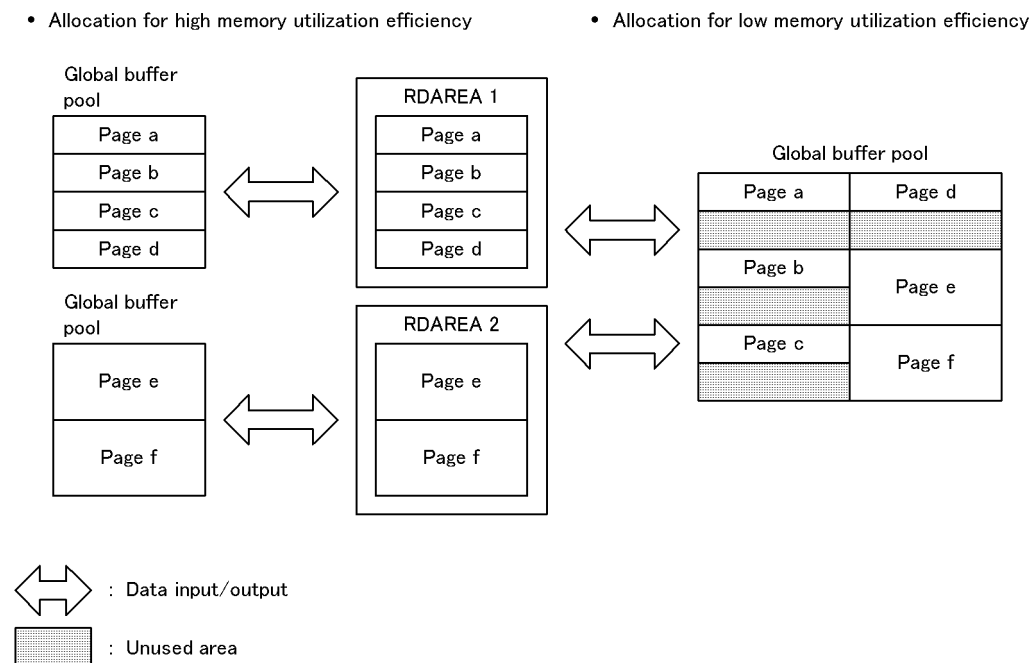
(1) Multiple RDAREAs with different page lengths

If there are multiple RDAREAs with different page lengths, all RDAREAs with the same or almost the same page length should be allocated to a single global buffer, so that memory utilization efficiency can be improved.

If multiple RDAREAs with very different page lengths are assigned to the same global buffer, the global buffer is allocated as appropriate for the RDAREA with the largest page length. When data pages are input/output in this global buffer for an RDAREA with a small page length, some of the global buffer sectors will remain unused, thereby adversely affecting the memory utilization efficiency.

The following figure shows an example of global buffer allocation.

Figure 11–2: Example of data global buffer allocation



For a HiRDB parallel server configuration, global buffers are maintained for each server as appropriate for that server's RDAREA with the largest page length. For example, if the largest RDAREA page length on back-end server 1 is 4,096 bytes and the largest RDAREA page length on back-end server 2 is 8,192 bytes, the global buffer sizes that could be allocated would be 4,096 for back-end server 1 and 8,192 for back-end server 2.

(2) Allocating multiple RDAREAs to one global buffer

If a single HiRDB file system area contains a HiRDB file that consists of multiple RDAREAs, all those RDAREAs should be allocated to the same global buffer.

(3) Multiple RDAREAs with different UAP access methods

If multiple RDAREAs have the same page length, but their UAP access methods are different, each RDAREA should be allocated to a different global buffer. Examples of such RDAREAs include RDAREAs with different usage, RDAREAs with frequent sequential processing and infrequent update processing, and RDAREAs subject to frequent addition or update processing.

(4) Addition of RDAREAs expected

The database structure modification utility (`pdmod`) can be used to add the following types of RDAREAs:

- User RDAREAs
- User LOB RDAREAs
- Data dictionary LOB RDAREAs
- Data dictionary RDAREAs for storing a dictionary table for management of stored procedures
- List RDAREAs

For an RDAREA containing a table that uses flexible hash partitioning, ALTER TABLE can be used to add RDAREAs.

Before an added RDAREA can be used, a global buffer must be allocated to it. Thus, if it is expected that RDAREAs will need to be added in the future, global buffers for which the `-o` option is specified in the `pdbuffer` operand must be provided in the system common definition, taking into account the largest likely page length for RDAREAs that may be added later.

If no global buffers have been allocated in advance, global buffer allocation must be redefined in order to add an RDAREA and make it usable; the `pdbuffer` operand in the system common definition is used for this purpose.

(5) Notes about allocating global buffer pools to list RDAREAs

When allocating global buffer pools to list RDAREAs, note the following, as well as the design considerations for global buffer pool allocation to user RDAREAs:

1. If you create many lists while sharing the global buffers for list RDAREAs and for tables and indexes, tables or indexes may be swept out of the global buffers. Therefore, if possible allocate a global buffer that is dedicated to list RDAREAs without sharing it.
2. For the number of global buffer sectors for list RDAREAs, specify a value that is at least the number of concurrently accessible lists times 1.5.
3. If you want to share the global buffers for list RDAREAs and for tables and indexes, share those RDAREAs with page lengths that are the same or close to each other.
4. If you specify the prefetch facility for the global buffer for list RDAREAs, executing the following SQL statements reads a page with a size of one segment at one time:
 - If you use the `SELECT` statement to search a table via lists, the system reads a list page in batch mode.
 - If you use the `ASSIGN LIST` statement to create a list, the system reads a list page of the list specified in the `FROM` clause in batch mode.

11.1.3 Allocating LOB global buffers

If any of the following conditions is applicable, a global buffer must be allocated for the LOB RDAREA, so that the number of input/output operations on data stored in the LOB RDAREA can be reduced:

- Plug-in index is stored
- Buffering effects can be expected because there is not much data
- LOB data that is accessed frequently is stored

A LOB global buffer should be allocated to a single RDAREA in order to avoid buffering interference between RDAREAs. LOB global buffers can be allocated to the following types of LOB RDAREAs:

- Data dictionary LOB RDAREAs
- User LOB RDAREAs
- Registry LOB RDAREAs

11.1.4 Global buffer allocation procedures

(1) Index global buffer

Use the *authorization-identifier.index-identifier* format to specify the index to which the index global buffer is allocated in the `pdbuffer` operand's `-i` option in the system common definition.

For a cluster key, HiRDB determines the index identifier. Therefore, after you have defined the table that specifies the cluster key, search the `INDEX_NAME` column of the `SQL_INDEXES` table (a dictionary table) to confirm the index identifier. The cluster key's index identifier is displayed as follows.

```
(CLUSTER table-number)
```

For details about dictionary table retrieval and the `SQL_INDEXES` table, see the *HiRDB Version 9 UAP Development Guide*.

When an index global buffer is allocated to the defined index, terminate HiRDB normally, and then specify the `pdbuffer` operand to allocate the index global buffer. If you omit this task, the defined index will use the data global buffer allocated to the index storage RDAREA.

■ Approaches to calculating the global buffer sector count

The general rule is to make the global buffer sector count greater than the total page count of the index (the value calculated as the number of pages stored in the index). From that count, reduce the global buffer sector count according to the importance of the index.

The number of index pages in use can be checked with the database condition analysis utility (`pddbstat`).

(2) Data global buffer

The name of the RDAREA for which a global buffer is to be allocated is specified in the `-r` option of the `pdbuffer` operand in the system common definition.

(3) LOB global buffer

To allocate a LOB global buffer, use the following procedure:

1. Specify in the `-r` option of the `pdbuffer` operand in the system common definition the name of the LOB RDAREA for which a global buffer is to be allocated.
2. Specify in the `-b` option of the `pdbuffer` operand in the system common definition the name of the LOB RDAREA for which a global buffer is to be allocated.

(4) Example of global buffer definition

Organization of RDAREAs

The following shows the organization of the RDAREAs:

Type of RDAREA	RDAREA name
Master directory RDAREA	RDMAST
Data directory RDAREA	RDDIR
Data dictionary RDAREA	RDDIC
User RDAREAs	USER01, USER02, USER03
User LOB RDAREA	ULOB03
Data dictionary LOB RDAREAs	DICLOB01, DICLOB02
List RDAREA	LIST01

Definition example

The following shows an example of global buffer definition:

```

pdbuffer -a DGB1 -n 1000 -r RDMAST,RDDIR,RDDIC 1
pdbuffer -a DGB2 -n 1000 -r USER01,USER02
pdbuffer -a DGB3 -n 1000 -r USER03
pdbuffer -a DGB4 -n 1000 -r ULOB03
pdbuffer -a DGB5 -n 1000 -r DICLOB01
pdbuffer -a DGB6 -n 1000 -r DICLOB02
pdbuffer -a DGB7 -n 1000 -r LIST01

pdbuffer -a LGB1 -n 1000 -b ULOB03 2
pdbuffer -a LGB2 -n 1000 -b DICLOB01
pdbuffer -a LGB3 -n 1000 -b DICLOB02

pdbuffer -a IGB1 -n 1000 -i USER1.INDX01 3
pdbuffer -a IGB2 -n 1000 -i USER1.INDX02

```

Explanation:

1. This is a definition of data global buffer. It uses the `-r` option to specify all RDAREAs to be created.
2. This is a definition of LOB global buffer. The RDAREAs specified with the `-b` option must also be specified with the `-r` option.
3. This is a definition of index global buffer. It uses the `-i` option to specify the index authorization identifier and index identifier.

The following provides a brief explanation of the `pdbuffer` operand's options that are used in this example:

- `-a`: Specifies the name of the global buffer.
- `-n`: Specifies the number of global buffer sectors.
- `-r`: Specifies the RDAREAs to be allocated to the data global buffer.
- `-b`: Specifies the LOB RDAREAs to be allocated to the LOB global buffer.
- `-i`: Specifies the indexes to which the index global buffer is to be allocated.

11.2 Setting the number of global buffer sectors

(1) Maximum value for shared memory considered

The number of global buffer sectors must be set so that the maximum value for the shared memory is not exceeded. If the required number of global buffer sectors is greater than the maximum value for the shared memory that is being used, the OS's `sam` command must be used to reset the maximum value per shared memory segment, and then the number of buffer sectors that fit in one shared memory segment must be set. There is a limit to the amount of shared memory in a server machine that can be allocated.

The amount of shared memory that can be allocated in one shared memory segment is determined by the OS. If a defined global buffer is too large to allocate in one shared memory segment, multiple shared memory segments are allocated, resulting in increased overhead for shared memory accesses.

(2) Buffer hit rate considered

Global buffers are allocated in shared memory and made resident in the memory. If more global buffer sectors are set than are actually needed, the amount of memory space being used for shared memory increases, which can have adverse effects on the system memory. Overhead for global buffer retrieval processing also increases. For these reasons, global buffers must be set so that the required minimum input/output performance is achieved.

To achieve the minimum input/output performance, global buffers should be set so that the overall hit rate for global buffers (update buffers hit rate + reference buffers hit rate) and the hit rate for reference buffers become high. This can be done by the following methods:

- Increase the number of global buffer sectors.
- Allocate each RDAREA or index to a separate global buffer sector.

To improve the performance even more after setting the number of buffer sectors on the basis of the aforementioned considerations, the `pdbufls` command or the statistics analysis utility (`pdstedit`) can be used after the HiRDB system has been started.

If the `pdbufls` command is used, the number of buffer sectors must be set so that the overall global buffers hit rate becomes high.

If the statistics analysis utility is used, the update buffers hit rate and reference buffers hit rate should be checked, and the number of buffer sectors should be set so that the overall global buffers hit rate becomes high.

For details about the `pdbufls` command and the statistics analysis utility (`pdstedit`), see the manual *HiRDB Version 6 Command Reference*.

(3) Setting procedure

The number of buffer sectors is set in the `-n` option of the `pdbuffer` operand in the system common definition.

11.3 Specifying the prefetch facility

The prefetch facility inputs multiple pages to a global buffer or local buffer in the batch mode.

(1) Effects of the prefetch facility

When a large amount of data is to be retrieved using a character special file, the prefetch facility can reduce the input/output time. This facility is especially effective for retrieving data without using an index or for using an index to search a table that contains many data items in ascending order.

(2) Criteria

The prefetch facility can be used to input multiple pages in the batch mode for the following SQL statements and utility:

- For the `SELECT`, `UPDATE`, and `DELETE` statements without using an index, multiple data pages can be input in the batch mode.
- For the `SELECT`, `UPDATE`, and `DELETE` statements (excepting the `=` and `IN` conditions) for a search in ascending order using an index, multiple index leaf pages can be input in the batch mode.
- For the `SELECT`, `UPDATE`, and `DELETE` statements (excepting the `=` and `IN` conditions) for a search in ascending order using a cluster key, multiple index leaf pages and data pages can be input in batch mode.
- For the database reorganization utility's (`pdroorg`) unload processing without using a local buffer, multiple index leaf pages and data pages can be input in the batch mode.

(3) Specification

(a) Global buffers

To use the prefetch facility, specify 1 or a greater value in the `pdbuffer` operand's `-m` option in the system common definition. Specify the number of pages to be read in batch mode in the `pdbuffer` operand's `-p` option.

(b) Local buffers

To use the prefetch facility, use the `-p` option of the `pdlbuffer` operand to specify the number of pages for batch input.

(4) Considerations

- When the prefetch facility is used, a buffer dedicated to batch input is obtained separately from the global buffers or local buffers. This results in an increase in the amount of shared memory required for global buffers. For details about the formula for calculating the amount of shared memory used by global buffers, see *Chapter 15. Storage Requirements for HiRDB*.
- Whether the prefetch facility is operating effectively can be determined by checking the prefetch hit rate with the statistics analysis utility (`pdstedit`) or with the `pdbuf1s` command.

11.4 Specifying the asynchronous READ facility

When multiple pages are batch input to a global buffer using the prefetch facility, the pages are pre-read as batch input in a synchronous process from the database processing server process to the batch input buffer. With the asynchronous READ facility, when the prefetch facility is used two batch input buffers are prepared, and while database processing uses one of the buffers, the asynchronous READ process pre-reads asynchronously into the other buffer. By executing the database processing concurrently with the pre-read input, processing time is reduced. For a HiRDB parallel server configuration, thread switchover processing reduces the input/output wait time.

The asynchronous READ facility cannot be used with local buffers. It is also not applicable to RDAREAs for which the SCHEDULE attribute is set. It operates using the prefetch facility.

(1) Effectiveness of the asynchronous READ facility

Although it is the same as the prefetch facility, compared to use of the prefetch facility alone, the asynchronous READ facility is effective for such processing as high processing-load joins. The asynchronous READ facility is particularly effective when used with character special files, for which the processing time is high. Conversely, if you are using a normal file or Hitachi disk array system disk, neither of which requires long I/O times, the facility might not be very effective.

(2) Specification

You must declare use of the prefetch facility by specifying 1 or greater in the `-m` option of the `pdbuffer` operand.

Use the `pd_max_ard_process` operand to specify the number of asynchronous READ processes. If 0 is specified, or if the operand is omitted, the asynchronous READ facility will not operate.

(3) Considerations

When the prefetch facility is used, two dedicated batch input buffers are used in addition to the global buffers. This increases the global buffer shared memory. For details about the formula for calculating the shared memory used by the global buffers, see *Chapter 15. Storage Requirements for HiRDB*.

11.5 Specifying deferred write processing

Deferred write processing is a type of processing in which data is written to disk only when the number of updated pages reaches a specified value, instead of data being written each time a `COMMIT` statement is issued. The point when the number of updated pages reaches the specified value (as determined by HiRDB) is called the deferred write trigger. HiRDB determines the number of updated pages to be written to disk on the basis of the updated output page rate for deferred write trigger that is specified with the `-w` option of the `pdbuffer` operand in the system common definition. Deferred write processing cannot be performed for the following RDAREAs:

- Data dictionary LOB RDAREAs
- User LOB RDAREAs
- Registry LOB RDAREAs
- List RDAREAs

(1) Effect of deferred write processing

Overloading caused by input/output processing can be reduced because data is not written to disk each time a `COMMIT` statement is issued.

(2) Specification

Either specify `sync` or `nothing` in the `pd_dbsync_point` operand. In addition, specify the updated output page rate for the deferred write trigger in the `-w` option of the `pdbuffer` operand.

(3) Considerations

1. If a table or index in an RDAREA allocated to a global buffer is updated frequently, a low value should be set as the updated output page rate for deferred write trigger.
2. If a global buffer is updated frequently but the same data is rarely updated, a high value should be set as the updated output page rate for deferred write trigger.
3. After the HiRDB system has been started, the `pdbuf1s` command can be used to improve performance even further. In other words, each global buffer's update request hit rate, which is an edit item, should be checked and set as follows:
 - If the update request buffer hit rate is high, set the updated output page rate for deferred write trigger to a low value.
 - If the update request buffer hit rate is low, set the updated output page rate for deferred write trigger to a high value.

(4) Notes

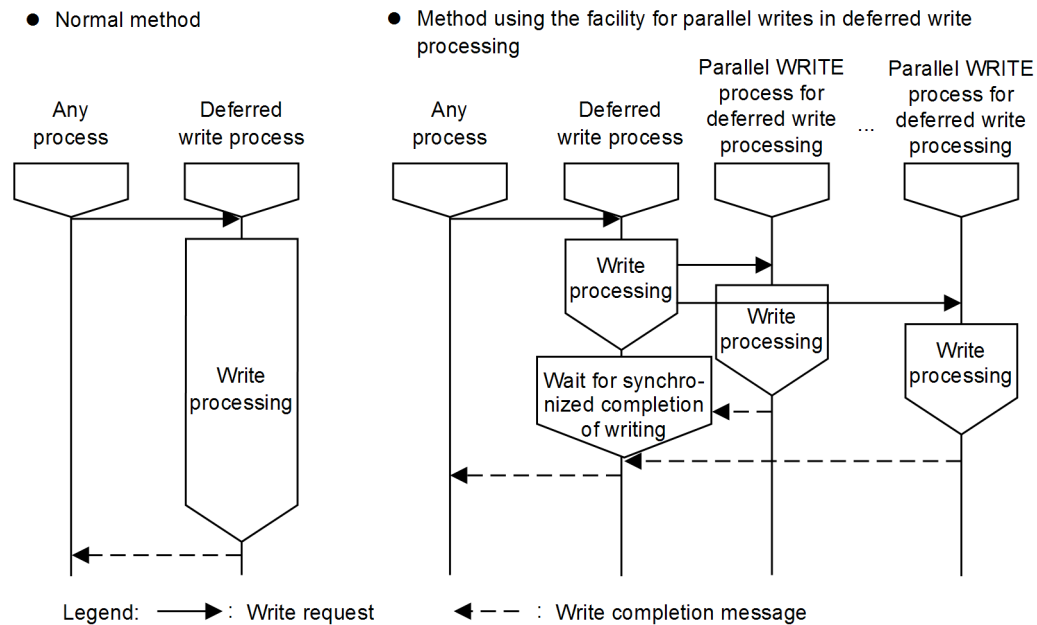
If the updated output page rate for deferred write trigger is set to a higher value than is necessary, disk write operations occur more frequently during deferred write processing. This may cause a concurrently executing transaction to be placed in input/output wait status, with adverse effects on the response time.

On the other hand, if the updated output page rate for deferred write trigger is set too low, the number of pages to be written to the database increases, which may cause a concurrently executing transaction to be placed in input/output wait status, with adverse effects on the response time.

11.6 Specifying the facility for parallel writes in deferred write processing

The facility for parallel writes in deferred write processing executes multiple processes of deferred write processing in parallel. The following figure provides an overview of two cases: the first uses the facility for parallel writes, and the second does not use the facility.

Figure 11–3: Overview of the facility for parallel writes in deferred write processing



(1) Effects of the facility for parallel writes in deferred write processing

For deferred write processing, the time required to write to disk is reduced because write processing is executed by multiple parallel WRITE processes.

(2) Specification

Specify in the `pd_dfw_awt_process` operand the number of parallel WRITE processes for deferred write processing, which performs write processing. Also specify the deferred write trigger request ratio in the `pd_dbbuff_rate_updpage` operand. If the `pd_dfw_awt_process` operand is omitted, the facility for parallel writes in deferred write processing is disabled.

(3) Considerations

If you specify the facility for parallel writes in deferred write processing, the number of parallel WRITE processes for deferred write processing increases. As a result, the CPU usage rate also increases

11.7 Setting the commit-time reflection processing

Commit-time reflection processing is a type of processing that involves writing the pages updated in a global buffer to disk whenever a `COMMIT` statement is issued.

(1) Effects of commit-time reflection processing

The contents of the database are guaranteed upon completion of a transaction because the updated database contents are written to disk when the `COMMIT` statement is issued. Therefore, there is no need to recover the database from a synchronization point during full recovery processing, thereby reducing the time required for full recovery processing.

(2) Specification

Specify `commit` in the `pd_dbsync_point` operand.

LOB RDAREAs are not affected by this operand. Directories are reflected at the point that the `COMMIT` statement is issued. How data is processed depends on whether a LOB global buffer was allocated. If a LOB global buffer was not allocated, data is reflected immediately upon issuance of the update request. If a LOB global buffer was allocated, data is reflected at the point that the `COMMIT` statement is issued. However, data is also reflected whenever the global buffer becomes full.

(3) Considerations

If the information obtained by the `pdbufls` command indicates that there are many output operations to disk and that the update request hit rate is low, the number of global buffer sectors should be set to a large value.

11.8 Global buffer LRU management

A global buffer LRU management method appropriate to the type of application (online or batch) can be selected.

11.8.1 LRU management methods

There are two LRU management methods:

- Independent LRUs for management of reference buffers and update buffers.
- Batch LRU management of global buffers.

(1) Independent LRUs for management of reference buffers and update buffers

With this method, reference buffers and update buffers are managed by independent LRUs.

If a shortage occurs in the global buffers, the least recently accessed reference buffer from among the global buffers is removed from the memory.

(a) Criteria

In the following case, it is preferable for the reference buffer and the update buffer to be managed in separate LRUs:

- There is a relatively small amount of update processing compared to retrieval processing, and the update buffer hit rate is high (the number of reference or update operations per transaction is relatively small, such as in the case of online applications).

(b) Specification

SEPARATE is specified in `pd_dbbuff_lru_option` in the system common definition.

(c) Notes

- If a large amount of update processing occurs, the reference buffer hit rate drops, slowing down retrieval processing.
- If either of the following applies, MIX is unconditionally assumed for the `pd_dbbuff_lru_option` operand. For this reason, the reference and update buffers cannot be managed by independent LRUs.
 - `commit` is specified in the `pd_dbsync_point` operand
 - `N` is specified in the `pd_dbbuff_binary_data_lru` operand

(2) Batch LRU management of global buffers

With this method, the global buffers are managed collectively by one LRU.

If a shortage occurs in the global buffers, the least recently accessed buffer from among the global buffers is removed from the memory.

(a) Criteria

It is beneficial to employ batch LRU management of global buffers in the following case:

- There is more update processing than retrieval processing, or a large amount of data is retrieved or updated sporadically (both retrieval processing and update processing involving a large amount of data occur, such as when online applications and batch applications co-exist).

(b) Specification

MIX is specified in `pd_dbbuff_lru_option` in the system common definition.

The updated output page rate for deferred write trigger is specified in the `-w` option of the `pdbuffer` operand in the system common definition.

(c) Notes

- If the update buffer hit rate is high, update buffers may be removed temporarily from memory due to retrieval of a large amount of data. In such a case, file read operations may occur as an extension of update processing, slowing down the processing.
- If `pd_dbsync_point=sync` is specified or omitted, file write operations may occur as an extension of retrieval processing, slowing down the retrieval processing.

11.8.2 LRU management suppression settings for a UAP

In an OLTP environment, the data recently cached in the global buffer may be removed from memory due to a UAP that searches and updates a large amount of data, resulting in a temporary reduction in OLTP performance. If it is possible to identify the UAP that searches and updates a large amount of data, this reduction in OLTP performance can be prevented by suppressing LRU management for the UAP.

Reference note

LRU management can only be suppressed for accesses from UAPs. Accesses from commands and utilities are managed by LRU. However, the following commands suppress LRU management.

- The database condition analysis utility (`pddbst`)
LRU management is suppressed if either of the following conditions is met:
 - The `-s` option is not specified
 - The `-s` option and `-b` option are specified
 - The free page release utility (`pdreclaim`)
LRU management is suppressed if the value of the `globalbuffer_lru` statement is no.
-

(1) Criteria

We recommend that you apply LRU management suppression when you execute a UAP that searches and updates a large amount of data using the global buffer.

(2) Effects

A page accessed by a UAP for which LRU management has been suppressed is cached in the global buffer as the oldest page accessed, regardless of the access frequency. This means that the pages accessed by this UAP are removed from memory before pages accessed by any UAP to which LRU management is being applied, thereby retaining the latter in memory. However, when a UAP that suppresses LRU management and a UAP that does not suppress LRU management both access the same page, LRU management is used for that page.

(3) Specification

Specify `NO` in the `PDDDBUFLRU` operand in the client environment definition.

(4) Notes

1. If a buffer shortage occurs, the pages accessed by a UAP for which LRU management has been suppressed are removed from memory regardless of the access frequency. Therefore, the response performance for such a UAP may decrease as the input/output count increases due to reduction in the buffer hit rate.
2. A UAP allocates 1-4 buffer sectors simultaneously. Therefore, even if LRU management suppression has been set, 1-4 cached pages may be removed from the global buffer per UAP.
3. If LRU management is suppressed for a UAP that executes update processing, more log information is output than when LRU management is not suppressed, because more write operations occur on the database and log output occurs frequently. To avoid space shortages, you should take the following steps:
 - Re-evaluate the sizes of the system log files.
 - If the UAP can be executed in the no-log mode, specify `NO` in `PDDDBLOG` in the client environment definition.

Use the formula shown below to determine the amount of log information when LRU management is suppressed. If you specify 1024 in the `pd_log_rec_leng` operand, you can minimize the amount of log information that is output when LRU management is suppressed.

$\text{update get count}^{\#} \times \text{value of the pd_log_rec_leng operand}$

[#]: To determine the update-get-count, check the value of `DIDUC` in the UAP statistical report or in the UAP statistical information.

4. When a rollback occurs, LRU management might be used even if the UAP suppresses LRU management. This differs depending on the timing of the rollback, as shown below.

Rollback timing		Is LRU management used?
Timing is defined by SQL	There is a rollback to the commit-time immediately prior to execution of a specified control SQL <code>ROLLBACK</code> statement.	LRU management is suppressed.
Rollback is executed automatically by HiRDB	Processing cannot continue at the time SQL is executed, and an implicit rollback to the immediately prior commit-time is executed by HiRDB.	LRU management is suppressed.
	The UAP terminates abnormally and HiRDB rolls back to the immediately prior commit-time.	LRU management is used.
	Rollback is executed by a delayed rerun.	LRU management is used.

11.8.3 Setting suppression of LRU management of binary data accessed by UAPs

When you execute a UAP that accesses a great deal of large-sized binary data, caching the binary data in the global buffer pushes out of memory whatever was most recently cached in the global buffer, so performance might decline temporarily. If the binary data is not accessed frequently, performance declines can be avoided at this time by suppressing LRU management of the branch row page of the binary data.

Note that this setting is valid for `BINARY` type binary data. It is not valid for `BLOB` type binary data.

Reference note

LRU management can only be suppressed for accesses from UAPs. Accesses from commands and utilities are managed by LRU. However, the following commands suppress LRU management.

- Commands provided by plug-ins
- `pddbst`
LRU management is suppressed if either of the following conditions is met:
 - The `-s` option is not specified
 - The `-s` option and `-b` option are specified
- `pdreclaim`
LRU management is suppressed if the value of the `globalbuffer_lru` statement is `no`.

Note also that for the following commands, LRU management cannot be suppressed, but you can avoid pushing from memory the most recently cached data in the global buffer.

- `pdload`, `pdrorg`, `pdrbal`
Pushing of base row data out of the global buffer can be avoided by specifying the `-n` option and using a local buffer.
 - `pdpgbfon`
When the `-b` option is omitted, the branch row page of the binary data is not accessed.
-

(1) Application criteria

We recommend using suppression of LRU management of binary data accessed by UAPs when both of the following conditions are met.

- There are tables that include large binary data such as `BINARY` type, abstract data types that include `BINARY` type attributes, or `XML` type.
- Access of binary data is rare.

! Important note

Do not apply this setting if you access binary data frequently.

(2) Effect of application

The branch row that stores binary data is cached in the global buffer as the least recently used page regardless of access frequency. For this reason, the branch row page is pushed from memory before the base row page, meaning that data in the base row page is not pushed from memory.

(3) Specification method

Specify `N` for the system common definition operand `pd_dbbuff_binary_data_lru`.

(4) Notes

1. When LRU management is suppressed, branch row pages of binary data that UAPs access are pushed from memory when the buffer runs short, regardless of access frequency. For this reason, UAPs that access binary data branch row pages might experience degraded response performance as well due to the increase in the number of I/Os that results from a lower buffer hit rate.
2. UAPs secure one to four sectors of buffer simultaneously. For this reason, one to four of the pages cached in the global buffer might be pushed out of the cache for each UAP, even when suppression of LRU management has been specified.
3. When LRU management is suppressed, there are numerous writes to databases and log output events when UAPs are executed that update binary data branch row pages. For this reason, the amount of log output will be greater than if LRU management is not suppressed. Do the following to make sure that capacity does not run short.

- Re-calculate the size of the system log file.
- If you are executing in no-log mode, specify `NO` for client environment definition `PDDBLOG`.

The log size when LRU management is suppressed can be determined using the following formula. If `1024` is specified as the `pd_log_rec_leng` operand, the log output size when LRU management is suppressed will be minimized.

number of update GETs of UAPs that access binary data[#] × pd_log_rec_leng operand value

#

The number of update GETs can be confirmed using the `DIDUC` value in the UAP statistical report or the `DIDUC` value in the statistical information pertaining to UAPs.

4. When a rollback occurs, LRU management might be used even if the UAP suppresses LRU management. This differs depending on the timing of the rollback, as shown below.

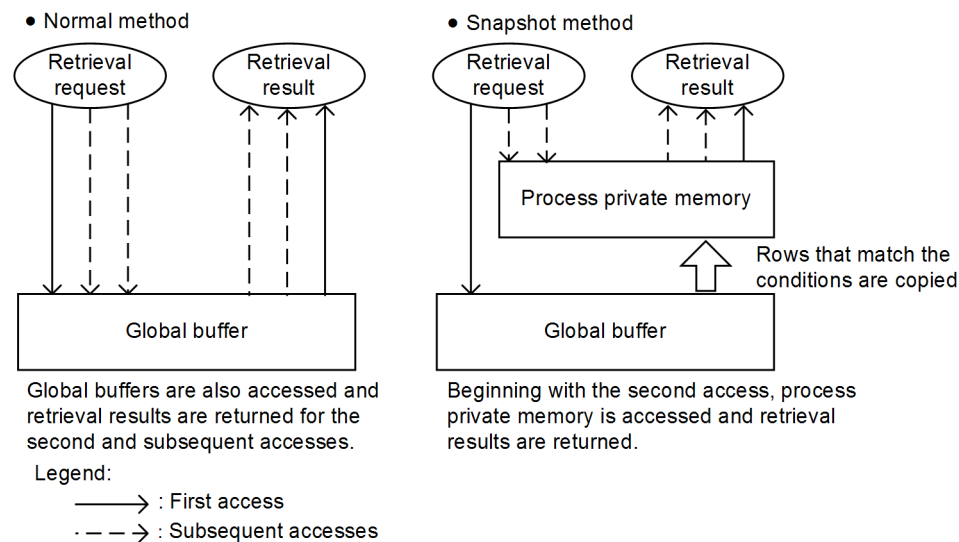
Rollback timing		Is LRU management used?
Timing is defined by SQL	There is a rollback to the commit-time immediately prior to execution of a specified control SQL <code>ROLLBACK</code> statement.	LRU management is suppressed.
Rollback is executed automatically by HiRDB	Processing cannot continue at the time SQL is executed, and an implicit rollback to the immediately prior commit-time is executed by HiRDB.	LRU management is suppressed.
	The UAP terminates abnormally and HiRDB rolls back to the immediately prior commit-time.	LRU management is used.
	Rollback is executed by a delayed rerun.	LRU management is used.

5. When this facility is applied, `MIX` is unconditionally assumed for the `pd_dbbuff_lru_option` operand. For this reason, the reference and update buffers cannot be managed by independent LRUs.

11.9 Page access using the snapshot method

When retrieval is performed but facilities designed to improve performance (such as the rapid grouping facility) cannot be applied, the global buffers are accessed several times to retrieve rows that satisfy the retrieval conditions. With the snapshot method, all rows in the buffer that match the retrieval conditions are copied into the process private memory the first time they are accessed, and when the same pages are accessed subsequently the retrieval result is returned by referencing the process private memory. The following figure provides an overview of the snapshot method.

Figure 11–4: Overview of the snapshot method



(1) Effectiveness of accesses using the snapshot method

The row that first matches the search conditions is copied into process private memory, which allows the retrieval time for the second and subsequent accesses to be shortened. The number of times that the global buffers are accessed is also reduced, preventing a concentration of accesses on the global buffer.

(2) Specification

Specify `SNAPSHOT` (the default value) in the `pd_pageaccess_mode` operand.

(3) Considerations

When the snapshot method is specified for use, the process private memory is maintained automatically on the basis of the page size of the RDAREA where the table or index is stored. For details about calculating the size of the maintained process private memory for a HiRDB single server configuration, see *15.1.6(4) Procedure for obtaining the size of the memory required when the snapshot method is used*; for a HiRDB parallel server configuration, see *15.2.6(4) Procedure for obtaining the size of the memory required when the snapshot method is used*.

(4) Snapshot method applicability

The following table indicates whether you need to apply the snapshot method for retrievals.

When *No* is indicated for applicability, there is no impact on access performance even when `SNAPSHOT` is specified in the system definition's `pd_pageaccess_mode` operand. Consequently, the snapshot method is not applicable.

Table 11–1: Applicability of the snapshot method for retrievals

Condition		Applicability	
		Tables	Indexes
Retrievals for which <code>pd_indexlock_mode=KEY</code> is specified in the system common definition, yet the following conditions are not satisfied:		N/R	No
<ul style="list-style-type: none"> • <code>WITHOUT LOCK NOWAIT</code> specified for retrieval • <code>LOCK TABLE</code> required for retrieval 			
Retrievals using a holdable cursor		No	No
Retrievals for which the retrieval method is an index scan (<code>INDEX SCAN</code> , <code>MULTI COLUMNS INDEX SCAN</code>)	<code>WITHOUT LOCK WAIT</code> is specified	No	No
	Other than above	No	Yes
Retrievals for which the retrieval method is <code>ROWID FETCH</code>		No	N/R
Retrievals with the following columns specified: <ul style="list-style-type: none"> • <code>VARCHAR</code>, <code>MVARCHAR</code>, <code>NVARCHAR</code> columns with a defined length greater than 256 bytes • Recursive columns • Columns with abstract data types • <code>LOB</code> columns • Binary columns with a defined length greater than 256 bytes 	<code>WITHOUT LOCK WAIT</code> is specified	No	Key
	Other than above	No	Yes
Retrievals for which retrieval conditions specify a hit rate of 1 hit per page		No	No
Retrievals using a plug-in index		No	No
Retrievals in a dictionary table		No	No

Legend:

Yes: Applicable.

No: Not applicable.

Key: Applicable when the retrieval type is a key scan (`KEY SCAN`, `MULTI COLUMNS KEY SCAN`).

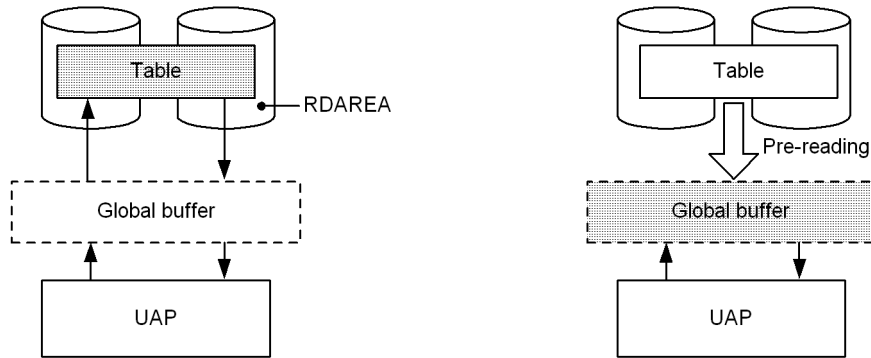
N/R: Not relevant or condition does not apply.

11.10 Global buffer pre-writing

Global buffer pre-writing is a function for reading data from a specified table or index in advance and placing it in the global buffer. The following figure provides an overview of global buffer pre-writing.

Figure 11–5: Overview of global buffer pre-writing

- When global buffer pre-writing is not performed
- When global buffer pre-writing is performed



Explanation:

- When global buffer pre-writing is not performed
When a UAP accesses a table immediately after HiRDB has started, the UAP reads data from the table because there is no data in the global buffer (physical input/output operations occur). Thereafter, when the UAP accesses this table data, no read operation occurs on those pages that have been written into the global buffer. To access other pages, read operations occur.
- When global buffer pre-writing is performed
The UAP can access the table without having to read data from it because the table data has already been written into the global buffer in advance (no physical input/output operations occur). Thereafter, when the UAP accesses this table, no read operation on the table occurs.

(1) Effects of global buffer pre-writing

The buffer hit rate improves because data is read from a specified table or index in advance. By pre-reading a table or index on which many input/output operations are expected immediately after HiRDB starts and prior to starting online applications, you can expect a high buffer hit rate.

(2) Execution method

Specify the table and index to be pre-read and execute the global buffer residence utility (`pdpgbfon`).

(3) Considerations

- You need more global buffer sectors than the actual number of pages that contain the table or index to be pre-read.
- If there are not enough global buffer sectors, the LRU management method removes the oldest page information from the global buffer (the oldest page in the global buffer accessed according to the `pd_dbbuff_lru_option` operand value in the system definition). Therefore, executing `pdpgbfon` serves no purpose when there are not enough global buffer sectors.
- When the global buffer residence utility (`pdpgbfon`) is used, the prefetch facility takes effect because data is pre-read in the order the pages were stored. When you define the database, you can reduce the execution time by specifying a prefetch count.

11.11 Local buffers

Local buffers are maintained in the process private memory and are used for input/output of data that is stored in an RDAREA on disk. The types of local buffers are as follows:

- Index local buffers
These buffers are used for input/output of index data. Index local buffers are allocated in units of indexes.
- Data local buffers
These buffers are used for input/output of data. Data local buffers are allocated in units of RDAREAs.

Local buffers are defined for each UAP in the UAP environment definition. By allocating a dedicated local buffer to a UAP, it is possible to avoid global buffer contention with other UAPs or waiting for buffer locks. For details about UAP environment definition, see the manual *HiRDB Version 9 System Definition*.

You should define local buffers when both of the following conditions apply:

- A large amount of data is to be retrieved or updated
- The RDAREA to be accessed should not be accessed by other UAPs

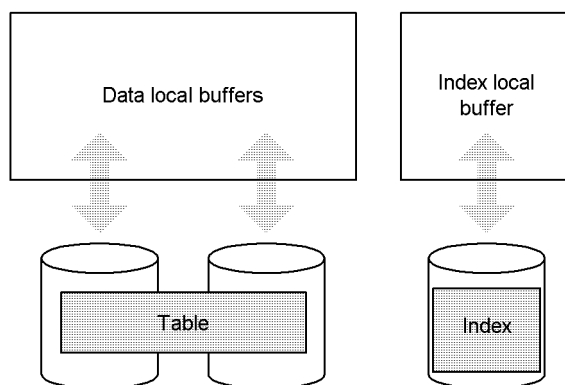
Since UAPs that are always connected to HiRDB have a major impact on the system (in terms of use of memory, server processes and the like), do not define local buffers for such UAPs.

11.11.1 Allocating index local buffers

If data local buffers and index local buffers are defined separately, data retrieval and index retrieval will be conducted independently even if they are performed concurrently. Therefore, even if a large amount of data is to be retrieved under all conditions, the frequency of index input/output can be reduced and processing time can be shortened.

The following figure provides an overview of index local buffers when table data and index data are respectively allocated to local buffers.

Figure 11–6: Overview of index local buffers



11.11.2 Allocating data local buffers

(1) Multiple RDAREAs with different page lengths

If there are multiple RDAREAs with different page lengths, all RDAREAs with the same or almost the same page length should be allocated to a single local buffer, so that memory utilization efficiency can be improved.

If multiple RDAREAs with very different page lengths are assigned to the same local buffer, the local buffer is allocated as appropriate for the RDAREA with the largest page length. When data pages are input/output on an

RDAREA with a small page length, some areas in a single local buffer sector will remain unused, thereby adversely affecting the efficiency of memory usage.

(2) Multiple RDAREAs with different UAP access methods

If multiple RDAREAs have the same page length, but their UAP access methods are different, each RDAREA should be allocated to a different local buffer. Examples of such RDAREAs include RDAREAs with different usage, RDAREAs with frequent sequential processing and infrequent update processing, and RDAREAs subject to frequent addition or update processing.

11.11.3 Allocating local buffers

To allocate index local buffers, you specify in the `-i` option of the `pdlbuffer` operand the name of the index (*authorization-identifier.index-identifier*) for which the index local buffer is to be allocated.

To allocate data local buffers, you specify in the `-r` option of the `pdlbuffer` operand the name of the RDAREA for which the data local buffer is to be allocated.

The following are examples of local buffer definitions:

```
pdlbuffer -a localbuf01 -r RDAREA01,RDAREA02 -n 1000 1
pdlbuffer -a localbuf02 -i USER01.INDX01 -n 1000 2
```

Explanation:

1. Allocates data local buffers to two RDAREAs, RDAREA01 and RDAREA02.
2. Allocates an index local buffer to index USER01.INDX01.

11.11.4 Considerations about local buffers

If a server process terminates abnormally when a local buffer is being used, abort code Phb3008 is output and HiRDB (the unit in the case of a HiRDB parallel server configuration) terminates abnormally. If there is an update page when a server process terminates abnormally, it might not always be recoverable using a transaction recovery process. In such a case, you must perform recovery processing when HiRDB restarts. For details about the measures for HiRDB processing when a failure occurs while local buffers are being used, see the *HiRDB Version 9 System Operation Guide*.

12 Designing Tables

This chapter explains items that should be examined during table design.

12.1 Items to be examined during table design

A HiRDB database is a relational database. The user must examine the design of a table, which is the logical structure of the database.

To begin with, tables must be normalized. Even among tables normalized in the same manner, table processing performance may vary depending on the method used to store the table in user RDAREAs. In some cases, operability may be more important than processing performance; therefore, tables must be designed to achieve expected results. The following table lists items to consider when you are designing a table.

Table 12–1: Items to consider when you are designing a table

Design task and items to be examined		Advantages	Disadvantages	Section
Table normalization		Table storage efficiency and processing efficiency improve.	Processing performance may be reduced if join retrieval of normalized tables is necessary during table retrieval.	12.2
Table row partitioning	Specification of table row partitioning	<ul style="list-style-type: none"> Operations can be performed in units of RDAREAs. For a HiRDB parallel server configuration, high-speed table access and workload distribution can be achieved because table access processing can be executed concurrently on multiple RDAREAs. 	<ul style="list-style-type: none"> The number of RDAREAs increases. If an index for this table is not row-partitioned, the level of concurrent executions is reduced due to index locking. 	12.3
	Key range partitioning	<ul style="list-style-type: none"> RDAREAs that contain specific table data are known. Data for each application can be stored in a separate RDAREA. 	Data cannot be stored uniformly without knowing the key ranges.	
	Flexible hash partitioning [#]	<ul style="list-style-type: none"> Data can be stored uniformly without having to know the key ranges. RDAREAs and hash function can be changed easily. It is easy to cope with the addition of CPUs or disks. 	<ul style="list-style-type: none"> It is difficult to know which table data is stored in which RDAREA. If a specific key is heavily weighted or duplicated, data cannot be stored uniformly. Uniqueness of the key cannot be checked. 	
	FIX hash partitioning [#]	<ul style="list-style-type: none"> The RDAREAs to be used to store data are determined by the key values. Data can be stored uniformly in RDAREAs without having to know the key ranges. It is easy to add CPUs or disks. Input data can be stored in the RDAREAs by creating a UAP that uses the hash function for table partitioning. 	Once data is stored in a table, user RDAREAs cannot be added, nor can the hash function be changed.	
Table matrix partitioning		<ul style="list-style-type: none"> Compared to normal key range partitioning, high-speed SQL processing and reduced operation time can be expected because data partitioned by key ranges can be partitioned further on the basis of the values in a different column. This method is applicable to a wider range of applications because key range partitioning can 	Compared to normal row partitioning, operation and management become complex because this method allows more detailed partitioning of RDAREAs.	12.4

Design task and items to be examined	Advantages	Disadvantages	Section
	be combined with hash partitioning.		
Defining a trigger	SQL can execute automatically in response to an operation on a table.	None	12.5
Creation of view table	<ul style="list-style-type: none"> If other users are given access privileges only for the view table but not the base table, the accessible range of the base table can be restricted on a row or column basis. If a view table is created beforehand using data that can be retrieved by a complicated query, the table referencing operation is simplified. A base table can be referenced or updated via its view table. 	None	12.6
Specification of FIX attribute	<ul style="list-style-type: none"> If row-by-row interface is used, access performance can be improved, even when there are many columns. Null value can be prohibited as input data for a table with the FIX attribute. If a table contains many columns, the disk space required can be reduced. 	None	12.7
Specification of primary key	Uniqueness constraint and NOT NULL constraint apply to a column for which a primary key is defined.	None	12.8
Specification of cluster key	<ul style="list-style-type: none"> Input/output time can be reduced when retrieving, updating, or deleting a row with a range specified, or when retrieving or updating data on the basis of a cluster key value. If UNIQUE is specified for the cluster key, all the values in the cluster key column must be unique. The database load utility (pdload) can be used to determine whether input data for a table is sorted in ascending or descending order of the cluster key values. When a table is being reorganized, the database reorganization utility (pdorg) can be used to determine whether the cluster key for the unloaded row matches the cluster key to be reloaded. 	<ul style="list-style-type: none"> Values in the column that constitutes the cluster key cannot be updated. The null value cannot be inserted in the column that constitutes the cluster key. When an attempt is made to add data to a table for which a cluster key is specified, there is overhead for retrieving the page that has the key value adjacent to the specified key value. 	12.9
Specification of suppress option	<ul style="list-style-type: none"> Disk space required can be reduced. Input/output time for retrieval processing can be reduced, such as retrieval of all entries. 	None	12.10

Design task and items to be examined	Advantages	Disadvantages	Section
Specification of no-split option	Data storage efficiency can be improved, thereby reducing the disk space required.	None	12.11
Specification of a binary data column	Variable-length large object data can be specified, such as document, image, and audio data	None	12.12
Specification of a character set	<p>Character string data can be stored in different character sets for each column in a table. This allows the following:</p> <ul style="list-style-type: none"> • When migrating from the VOS3 system to HiRDB, character data stored in a database can be retrieved, substituted, and compared in the collating sequence of character string data in the VOS3 system. • Retrieval, substitution, and comparison of character data in UTF-16. 	None	12.13
Specification of <code>WITHOUT ROLLBACK</code> option	Occurrences of locking can be reduced because the rows subject to update processing are unlocked automatically when table update processing is completed (such as when defining a table used for numbering applications).	None	12.14
Specification of the falsification prevention facility	Prevents table data errors or invalid updates.	There are restrictions on the facilities, SQL, utilities, and commands that can be executed on an RDAREA that contains a falsification prevented table.	12.15
Table with repetition column	<ul style="list-style-type: none"> • There is no need to join multiple tables. • The disk space required can be reduced because duplicated information is eliminated. • Better access performance can be achieved than when a separate table is used because related data items (repetition columns) are placed adjacent to each other. 	None	12.16
Table with abstract data type	Data with a complicated structure can be stored in a table to process it as normal table data.	None	12.17
Shared table	<ul style="list-style-type: none"> • Overhead caused by connection and data transfer between multiple back-end servers can be reduced. • Efficiency of parallel processing improves, such as when multiple transactions are executed concurrently. 	When a shared table is updated, all back-end servers lock the RDAREA that contains the shared table. If another application accesses another table in the same shared RDAREA, deadlock may occur.	12.18
Referential constraints	Integrity checking and data manipulation on data in multiple tables can be automated.	When referenced tables and referencing tables are updated, processing time increases because data integrity is checked.	12.19

Design task and items to be examined	Advantages	Disadvantages	Section
Check constraints	Data checking can be automated when data is added or updated.	When a table for which a check constraint has been defined is updated, processing time increases because data integrity is checked.	12.20
Compressed table	<ul style="list-style-type: none"> Table data storage efficiency improves and the database size can be reduced. Data compression processing need not be provided when UAP is developed because HiRDB compresses data. 	When data in compressed columns is manipulated by using SQL statements and utilities, there is overhead for compression and expansion processing.	12.21
Temporary table	<ul style="list-style-type: none"> Processing is not affected by other users because a dedicated table is created for each transaction or SQL session. Complex processing can be performed, such as storing intermediate processing results in a temporary table and then obtaining the final results by processing the intermediate results. No postprocessing is needed because temporary tables are deleted automatically. 	There is overhead for initializing a temporary table RDAREA when HiRDB starts or when the first INSERT statement is executed on a temporary table.	12.22

#: You should use the hash facility for hash row partitioning in the following cases:

- Hash row partitioning is to be used with the table.
- The amount of data is expected to increase.

If you add RDAREAs to handle an increase of data for a hash row partitioning table (if you increase the number of table row partitions), data may become uneven among the existing RDAREAs and newly added RDAREAs. The rebalancing facility for hash row partitioning can correct such unevenness of data when you increase the number of table row partitions. For details about the rebalancing facility for hash row partitioning, see the *HiRDB Version 9 System Operation Guide*.

12.2 Normalizing a table

It is important in terms of table storage efficiency and processing efficiency to normalize tables. The columns that constitute a table should be examined during table normalization.

This section describes the following normalization topics:

- Normalization for improved table storage efficiency
- Normalization for improved table processing efficiency

(1) Normalization for improved table storage efficiency

If a table has multiple columns that contain similar data, the table should be normalized so that all the columns contain unique data. Such normalization improves the efficiency of data storage for the table. This subsection describes how this works using the example depicted in the following figure.

Figure 12–1: Multiple columns in a table containing similar data

- Before normalization

STOCK

Product number	Product code	Product name	Cost	Stock
PNO	PCODE	PNAME	PRICE	QUANTITY
01010	101	Blouse	35.00	62
01011	101	Blouse	35.00	85
02021	202	Polo shirt	36.40	67
03530	353	Skirt	47.60	18
03531	353	Skirt	47.60	56
04121	412	Sweater	84.00	22
05910	591	Socks	2.50	300
05911	591	Socks	2.50	90
05912	591	Socks	2.50	280
06710	671	Shoes	45.00	45
06711	671	Shoes	45.00	76

These columns correspond 1:1.
Column information is redundant.

- After normalization

STOCK

PNO	PNAME	PRICE	QUANTITY
01010	Blouse	35.00	62
01011	Blouse	35.00	85
02021	Polo shirt	36.40	67
03530	Skirt	47.60	18
03531	Skirt	47.60	56
04121	Sweater	84.00	22
05910	Socks	2.50	300
05911	Socks	2.50	90
05912	Socks	2.50	280
06710	Shoes	45.00	45
06711	Shoes	45.00	76

PRODUCT

PCODE	PNAME
101	Blouse
202	Polo shirt
353	Skirt
412	Sweater
591	Socks
671	Shoes

The PCODE and PNAME columns in the STOCK table have a one-to-one correspondence before normalization, which means that the data in these columns is redundant. For this case, another table called PRODUCT can be created that consists of the STOCK table's PCODE and PNAME columns. The PRODUCT table is created so that the PCODE and PNAME columns do not contain duplicative data.

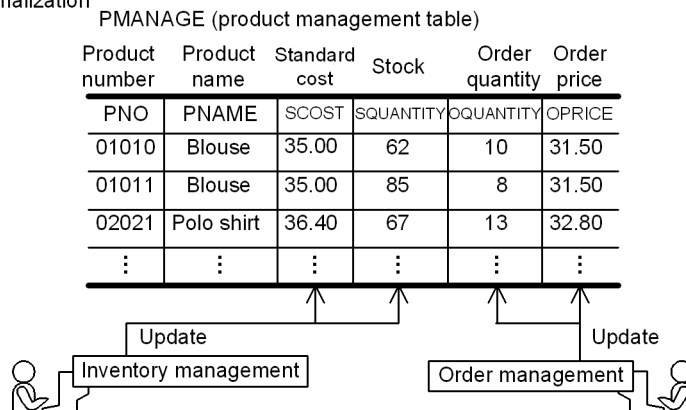
(2) Normalization for improved table processing efficiency

(a) Same table used by multiple applications

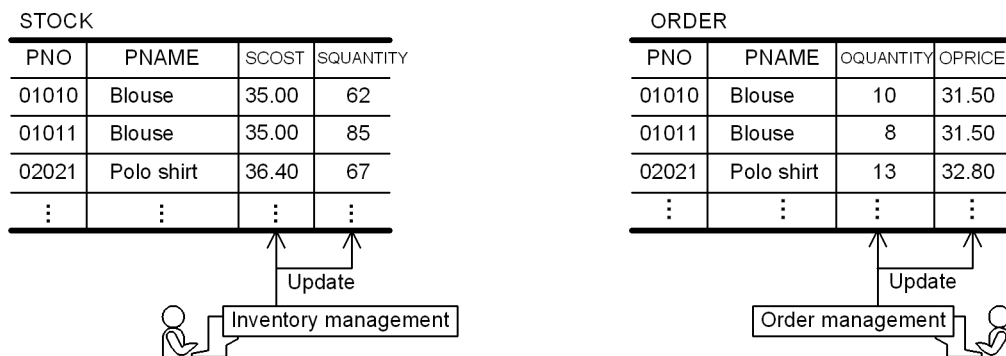
If the same table is used by multiple applications, normalization can result in a separate table for each application. This can improve the level of concurrent execution for each table. This subsection describes how this works using the example depicted in the following figure.

Figure 12–2: Same table used by multiple applications

- Before normalization



- After normalization



The PMANAGE (product management) table is used by the inventory management application and the orders management application. The PMANAGE table can be normalized to the STOCK table that is used only by the inventory management application and the ORDERS table that is used only by the orders management application.

(b) Columns with different access frequencies

If some of a table's columns are accessed frequently and some are not, normalization can result in a table consisting of the columns that are accessed frequently and a table consisting of the columns that are accessed infrequently. This subsection describes how this works, using the example shown in the following figure.

Figure 12–3: Columns with different access frequencies

- Before normalization

STOCK

PNO	PNAME	PRICE	QUANTITY
01010	Blouse	35.00	62
01011	Blouse	35.00	85
⋮	⋮	⋮	⋮
04121	Sweater	84.00	22
05910	Socks	2.50	300
05911	Socks	2.50	90
05912	Socks	2.50	280
⋮	⋮	⋮	⋮

Columns with a retrieval frequency of 90%

Columns with a retrieval frequency of 10%

- After normalization

STOCK2

PNO	QUANTITY
01010	62
01011	85
⋮	⋮
04121	22
05910	300
05911	90
05912	280
⋮	⋮

PRODUCT

PNO	PNAME	PRICE
01010	Blouse	35.00
01011	Blouse	35.00
⋮	⋮	⋮
04121	Sweater	84.00
05910	Socks	2.50
05911	Socks	2.50
05912	Socks	2.50
⋮	⋮	⋮

Taking the *STOCK* table shown in the above figure, if the retrieval frequency ratio of the *PNO* and *QUANTITY* columns to the *PNAME* and *PRICE* columns is 9:1, normalization of the *STOCK* table should result in a table (*STOCK2*) consisting of the columns that are retrieved frequently and a table (*PRODUCT*) consisting of the columns that are retrieved infrequently.

Assume that 10,000 physical input/output operations are required in order to retrieve all entries in the *STOCK* table. When the *STOCK* table is divided into the two tables *STOCK2* and *PRODUCT*, the numbers of physical input/output operations required in order to retrieve all entries in these two tables drops to 4,500 ($5,000 \times 0.9$) and 500 ($5,000 \times 0.1$), respectively. As a result, only 5,000 physical input/output operations are required to retrieve all entries, thereby improving the overall table processing efficiency.

12.3 Table row partitioning

This section describes the design method for partitioning a table by rows.

HiRDB/Workgroup Server does not support the row-partitioning of a table or an index.

12.3.1 Table row partitioning

Dividing a table and storing it in multiple user RDAREAs is called table row partitioning. A table partitioned by rows is called a row-partitioned table.

The RDAREAs used to store a row-partitioned table must be on different disks.

(1) Criteria

You should employ table row partitioning in the following circumstances:

- There is a large amount of data.
- Data access is concentrated at specific time periods.
- User RDAREAs are managed in units of table partitions (such as for data storage in tables, table reorganization, and making backups).

(2) Definition procedure

You define table row partitioning with the `CREATE TABLE` definition SQL statement. For details about how to define the table row partitioning, see *6.2 Creating a row-partitioned table*.

12.3.2 Types of table row partitioning

There are two ways to partition a table by rows:

- Key range partitioning
- Hash partitioning (flexible hash partitioning or FIX hash partitioning)

These two methods of table row partitioning are explained as follows.

(1) Key range partitioning

Key range partitioning divides a table into groups of rows on the basis of ranges of values in a specific column in the table. The column used as the condition for table row partitioning is called the partitioning key.

When this method is used, it is possible to tell which table data is stored in which RDAREA.

Row partitioning can be specified in two ways:

(a) By specifying storage conditions

Comparison operators are used to specify conditions for determining which table data is to be stored in each RDAREA. Only one range of storage condition values can be specified for each RDAREA.

(b) By specifying boundary values

Literals are used to specify the boundary values of the data to be stored in each RDAREA. Multiple ranges determined by boundary values can be specified for one RDAREA. Boundary values can also be specified with matrix partitioning. For details about matrix partitioning, see *12.4 Table matrix partitioning*.

(2) Hash partitioning

Hash partitioning uses the values of a table column as a hash function for dividing the table and storing it uniformly in RDAREAs. The column specified for partitioning the table is called the partitioning key. This method is used to

distribute the table data uniformly among the RDAREAs without having to deal with key ranges. Hash partitioning can be combined with key range partitioning with boundary values specified to achieve matrix partitioning. For details about matrix partitioning, see *12.4 Table matrix partitioning*.

The two types of hash partitioning are flexible hash partitioning and FIX hash partitioning.

When a table is divided and stored in multiple RDAREAs using flexible hash partitioning, there is no way to know which data is stored in which RDAREA. Therefore, in order to search for particular data in the table, all back-end servers containing the table are subject to search processing.

When a table is partitioned using FIX hash partitioning, HiRDB identifies which table data is stored in which RDAREA. Therefore, only the back-end server believed to contain the desired data is subject to search processing.

(a) Selecting the partitioning key

The key selected as the partitioning key should satisfy the following conditions:

- The key values are evenly distributed.
- There are few duplicated key values.

For hash partitioning, either a single column or multiple columns can be selected as the partitioning key. If a single column is specified and there are too few different values in the column for purposes of partitioning or the key values are unevenly distributed, data may not be divided uniformly. In this case, more than one column on which partitioning can be based should be specified in order to distribute the data more uniformly among the RDAREAs.

(b) Types of hash functions

The hash functions available for hash partitioning include the following:

- HASH0
- HASH1
- HASH2
- HASH3
- HASH4
- HASH5
- HASH6
- HASHA
- HASHB
- HASHC
- HASHD
- HASHE
- HASHF
- HASHZ

If the table is not a rebalancing table, or if you want to specify a hash function for the second dimension of a matrix-partitioned table

Specify a hash function between HASH0 and HASH6, or HASHZ. HASH6 provides the most uniform hashing result, so normally you specify HASH6. However, depending on the data in the partitioning key, the hashing result may not be uniform. In such a case, specify one of the other hash functions.

If the table is a rebalancing table

Specify a hash function between HASHA and HASHF. HASHF provides the most uniform hashing result, so normally you specify HASHF. However, depending on the data in the partitioning key, the hashing result may not be uniform. In such a case, specify one of the other hash functions.

For details about individual hash functions, see the descriptions of the hash functions in *Operands* under *CREATE TABLE (Define table)* in the manual *HiRDB Version 9 SQL Reference*.

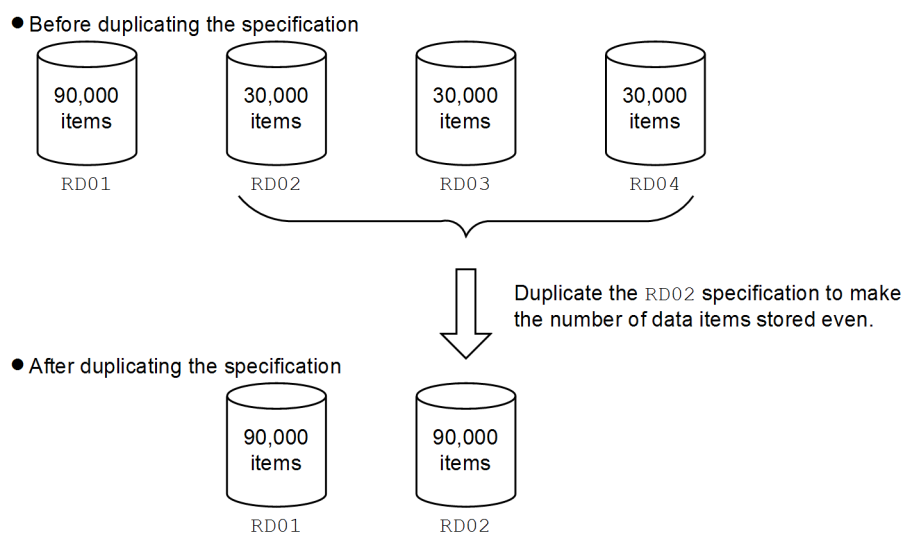
(c) Selecting a hash function

Selecting an appropriate hash function by actually storing data in a database

To select a hash function, use the following procedure:

1. Specify a hash function appropriate to the partitioning key.
2. Use the database condition analysis utility (`pddbstat`) to check the number of rows stored in each RDAREA.
3. If there is an uneven distribution in the number of rows stored in the RDAREAs, change the hash function so that a uniform number of rows is stored in each RDAREA.
4. If the method described in step 3 does not result in an even distribution of stored rows, duplicate the specification of an RDAREA with fewer rows stored to make the number of stored rows even. The following figure shows an example.

Figure 12–4: Example of duplicating a table storage RDAREA specification using hash partitioning

**Selecting an appropriate hash function by creating a UAP that uses a hash function for table partitioning**

To select a hash function, use the following procedure:

1. Create an application program that locates an uneven distribution in the number of data items in each RDAREA by using the hash functions for table partitioning (function that outputs a partitioning condition specification sequence from the data values for the partitioning key), available from a HiRDB library.
2. For each hash function, obtain the number of data items in the partitioning condition specification sequence that is output by the hash function for table partitioning, then select the most evenly distributed hash function.

For details about how to create a UAP for using a hash function for table partitioning, see the *HiRDB Version 9 UAP Development Guide*.

(d) Times when hash functions are used

Hash functions are used at the following times:

- When data is loaded in units of tables
- When data is added
- When data is reloaded in units of tables

(3) Differences among key range partitioning, flexible hash partitioning, and FIX hash partitioning

The following table lists and explains the differences between key range partitioning, flexible hash partitioning, and FIX hash partitioning.

Table 12–2: Differences between key range partitioning, flexible hash partitioning, and FIX hash partitioning

Difference	Key range partitioning	Flexible hash partitioning	FIX hash partitioning
Database design	Key ranges must be taken into account when database is designed.	There is no need to take key ranges into account when database is designed.	There is no need to take key ranges into account when database is designed.
Retrieval	Only back-end servers that may contain the applicable data according to the search condition are subject to retrieval processing.	All back-end servers containing the table are subject to retrieval processing.	When an = or an IN predicate for a partitioning column is specified in a search condition, only RDAREAs that might contain the data are searched. However, when HASH0 is used as the hash function, RDAREAs subject to retrieval processing when an = predicate, IN predicate, range condition (using <, >, <=, or >=), BETWEEN predicate, LIKE predicate (starts-with comparison), or SIMILAR predicate (starts-with comparison) is specified are searched. ^{#1}
Support for increasing the amount of data	If keys increase, data may be concentrated in some RDAREAs.	Data is already stored uniformly in RDAREAs even if data increases.	Data is already stored uniformly in RDAREAs even if data increases.
Handling of RDAREA shutdown	SQLs can be executed if their search condition is specified so that no shutdown RDAREA is accessed.	If even one of the RDAREAs containing the table subject to retrieval is shut down, SQLs cannot be executed regardless of their search condition.	SQLs can be executed if their search condition is specified so that no shutdown RDAREA is accessed. ^{#2}
Change in number of table partitions	Table must be re-created and reorganized.	ALTER TABLE can be used to add RDAREAs, and reorganization of the table is not required.	Table must be re-created and reorganized. ALTER TABLE can be used to add RDAREAs only if the table contains no data.
Data loading and reloading in units of RDAREAs	Data is checked to see if it is the correct data to be stored in the corresponding RDAREAs.	Data is not checked to see if it is the correct data to be stored in the corresponding RDAREAs.	Data is checked to see if it is the correct data to be stored in the corresponding RDAREAs.
Method for creating an input data file by RDAREA during data loading	Input data is classified by RDAREA taking into account the key ranges.	Input data is classified so that the number of data items per RDAREA becomes uniform.	An application is created using a hash function for table partitioning, ^{#3} and input data is classified by RDAREA.
Updating of partitioning key	Updating must use existing values.	Can be updated.	Updating must use existing values.
UNIQUE definition for cluster key and index definition with UNIQUE specified	UNIQUE cannot be specified.	UNIQUE cannot be specified.	UNIQUE can be specified.
Changing the partition storage conditions by ALTER TABLE	Partitioning storage conditions can be changed for the following method: <ul style="list-style-type: none"> Boundary value specification Storage condition specification (only = is used for the storage condition comparison operator)	Partition storage conditions cannot be changed. RDAREAs can be added by ALTER TABLE.	Partition storage conditions cannot be changed. RDAREAs cannot be added by ALTER TABLE.

#1: The `ASSIGN LIST` statement results in a workload on the back-end servers to which the search condition is not applied.

#2: The `ASSIGN LIST` statement handles the entire table as being shut down.

#3: For details about how to create a UAP for using a hash function for table partitioning, see the *HiRDB Version 9 UAP Development Guide*.

(4) Specification rules when table row partitioning is defined

When table row partitioning is defined, the following specification rules apply:

- For key range partitioning
 - You can specify only one partitioning key.^{#1} The partitioning key cannot be updated.
 - If you specify a storage condition,^{#2} you cannot specify the same RDAREA more than once. Although you can specify the same RDAREA more than once in an environment variable specification,^{#3} you cannot specify the same RDAREA two or more times in succession.
- For hash partitioning
 - You can specify a maximum of 16 partitioning keys.^{#1} The same partitioning key cannot be specified more than once. Flexible hash partitioning allows updating of partitioning keys, but FIX hash partitioning does not.

#1: A column or repetition column of any of the following data types cannot be specified as the partitioning key:

- `CHAR`, `VARCHAR`, `MCHAR`, or `MVARCHAR` type whose defined length is 256 bytes or greater
- `NCHAR` or `NVARCHAR` type whose defined length is 28 characters or greater
- `BLOB` type
- `BINARY` type
- Abstract data type
- `TIMESTAMP` type whose decimal places precision is greater than 0
- `TIMESTAMP` type whose default value is `CURRENT_TIMESTAMP USING BES`

#2: When multiple storage conditions are specified, the conditions are evaluated in the order they were specified, and data is stored in the RDAREA that is specified in the first storage condition whose evaluation result is true. If none of the conditions results in true, the system stores data in the RDAREA for which no storage condition was specified. If there is no such RDAREA, data is not stored in any of the RDAREAs. The table definition is invalid if it contains an RDAREA in which no row is stored as a result of evaluating the conditions.

#3: A literal is specified for a boundary value. A character string literal with a length of 0 is not permitted. If you specify multiple boundary values, they must be specified in ascending order. Also, you must specify at the end an RDAREA for which no boundary value is specified.

(5) Examples of key range partitioning (with storage condition specified)

The following figure shows an example of key range partitioning (with a storage condition specified).

Figure 12–5: Example of key range partitioning with a storage condition specified

USR01

STOCK				
PCODE	PNAME	COLOR	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29
202M	Polo shirt	Red	36.40	67
302S	Skirt	White	51.10	65
353L	Skirt	Red	47.60	18
353M	Skirt	Green	47.60	56
411M	Sweater	Blue	84.00	12
412M	Sweater	Red	84.00	22
591L	Socks	Red	2.50	300
591M	Socks	Blue	2.50	90
591S	Socks	White	2.50	280
671L	Shoes	White	45.00	45
671M	Shoes	Blue	45.00	76

↑
Partitioning key

USR02

- Row partitioning table stored in USR01

STOCK (Data: 101L-353M)

PCODE	PNAME	COLOR	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29
202M	Polo shirt	Red	36.40	67
302S	Skirt	White	51.10	65
353L	Skirt	Red	47.60	18
353M	Skirt	Green	47.60	56

- Row partitioning table stored in USR02

STOCK (Data: 411M-617M)

PCODE	PNAME	COLOR	PRICE	SQUANTITY
411M	Sweater	Blue	84.00	12
412M	Sweater	Red	84.00	22
591L	Socks	Red	2.50	300
591M	Socks	Blue	2.50	90
591S	Socks	White	2.50	280
671L	Shoes	White	45.00	45
671M	Shoes	Blue	45.00	76

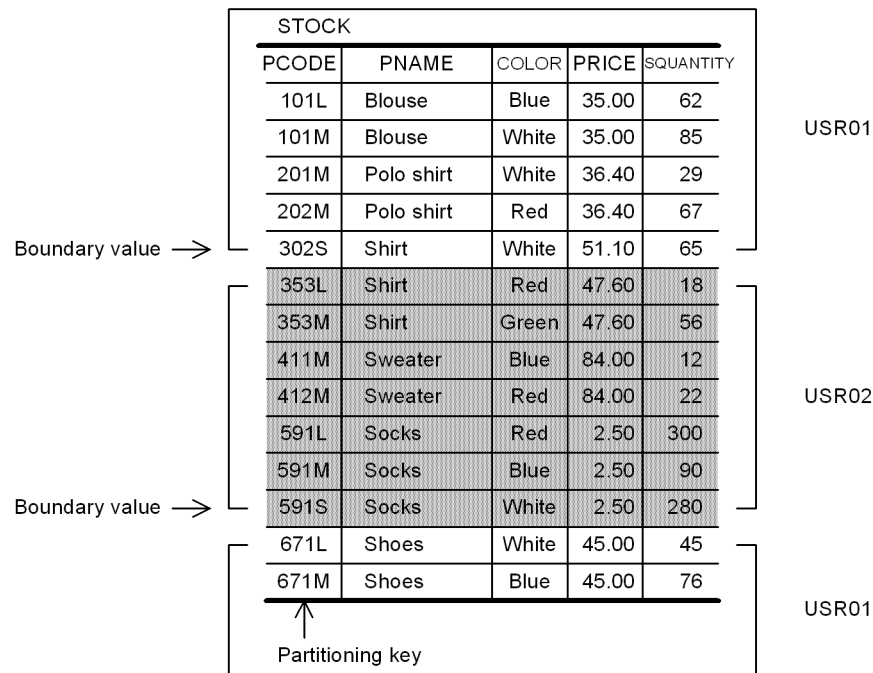
Explanation:

The STOCK table is partitioned and stored in two user RDAREAs (USR01 and USR02) using ranges of values in the product code (PCODE) column as the condition; the specified ranges are 100L–399S and 400L–699S.

(6) Example of key range partitioning (with boundary values specified)

The following figure shows an example of key range partitioning (with boundary values specified).

Figure 12–6: Example of key range partitioning with boundary values specified



- Row partitioning table stored in USR01

STOCK				
PCODE	PNAME	COLOR	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29
202M	Polo shirt	Red	36.40	67
302S	Skirt	White	51.10	65
671L	Shoes	White	45.00	45
671M	Shoes	Blue	45.00	76

- Row partitioning table stored in USR02

STOCK				
PCODE	PNAME	COLOR	PRICE	SQUANTITY
353L	Skirt	Red	47.60	18
353M	Skirt	Green	47.60	56
411M	Sweater	Blue	84.00	12
412M	Sweater	Red	84.00	22
591L	Socks	Red	2.50	300
591M	Socks	Blue	2.50	90
591S	Socks	White	2.50	280

Explanation:

The **STOCK** table is partitioned and stored in two user RDAREAs (**USR01** and **USR02**) using values in the product code (**PCODE**) column as boundary values; the specified boundary values are **302S** and **591S**.

(7) Example of flexible hash partitioning and FIX hash partitioning

The following figure shows an example of flexible hash partitioning and FIX hash partitioning.

Figure 12–7: Example of flexible hash partitioning and FIX hash partitioning

STOCK

PCODE	PNAME	COLOR	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29
202M	Polo shirt	Red	36.40	67
302S	Skirt	White	51.10	65
353L	Skirt	Red	47.60	18
353M	Skirt	Green	47.60	56
411M	Sweater	Blue	84.00	12
412M	Sweater	Red	84.00	22
591L	Socks	Red	2.50	300
591M	Socks	Blue	2.50	90
591S	Socks	White	2.50	280
671L	Shoes	White	45.00	45
671M	Shoes	Blue	45.00	76

↑
Partitioning key

- Row partitioning table stored in USR01

STOCK

PCODE	PNAME	COLOR	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
353M	Skirt	Green	47.60	56
411M	Sweater	Blue	84.00	12
412M	Sweater	Red	84.00	22
591M	Socks	Blue	2.50	90
591S	Socks	White	2.50	280
671M	Shoes	Blue	45.00	76

- Row partitioning table stored in USR02

STOCK

PCODE	PNAME	COLOR	PRICE	SQUANTITY
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29
202M	Polo shirt	Red	36.40	67
302S	Skirt	White	51.10	65
353L	Skirt	Red	47.60	18
591L	Socks	Red	2.50	300
671L	Shoes	White	45.00	45

Explanation:

The STOCK table is partitioned and stored in two user RDAREAs (USR01 and USR02) using the product code (PCODE) column as the partitioning key and using hash function 6.

The target RDAREAs for storage of actual data may differ from this example.

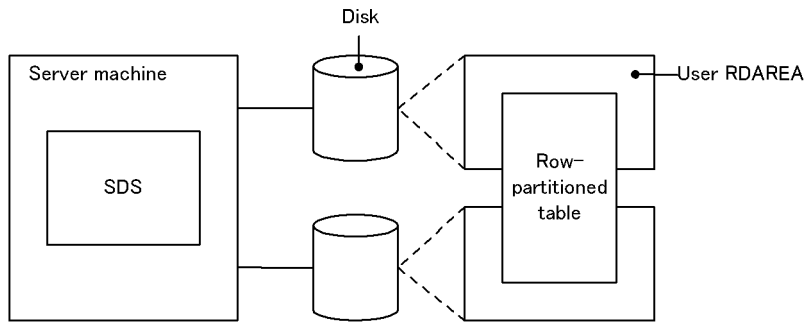
12.3.3 Forms of table row partitioning

Following are the basic forms of table row partitioning:

- Row partitioning within a server (applicable to a HiRDB single server configuration)
- Row partitioning among servers (applicable to a HiRDB parallel server configuration)

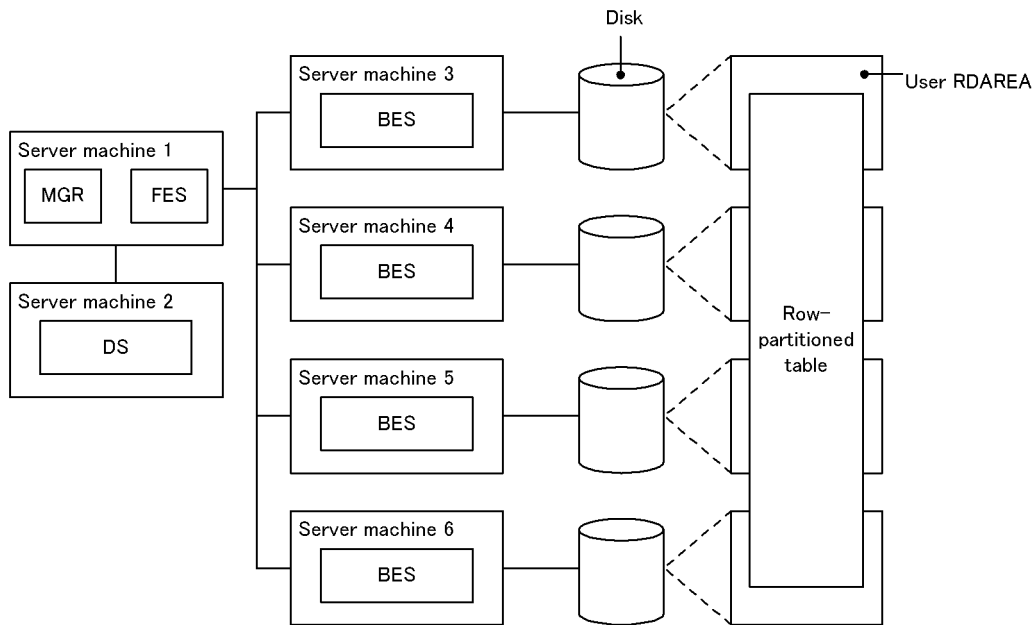
Figures 12-8 and 12-9 show these forms.

Figure 12–8: Table row partitioning form for a HiRDB single server configuration



SDS: Single server

Figure 12–9: Table row partitioning form for a HiRDB parallel server configuration



MGR: System manager
 FES: Front-end server
 DS: Dictionary server
 BES: Back-end server

12.3.4 Effects of table row partitioning

The effects obtained when a table is row-partitioned are discussed below.

(1) HiRDB single server configuration

Improved operability

Data storage in table, table reorganization, making backups, and additional operations are available for each user RDAREA.

Key range partitioning

The user RDAREAs that contain particular table data can be determined by searching the `SQL_DIV_TABLE` dictionary table. This means that if an error occurs in a user RDAREA, the unavailable data can be identified. For

details about dictionary table retrieval and the `SQL_DIV_TABLE` table, see the *HiRDB Version 9 UAP Development Guide*.

(2) HiRDB parallel server configuration

Improved performance

High-speed table access processing can be achieved, because table access processing can be handled in parallel over multiple user RDAREAs.

Workload of table access processing can be distributed to multiple back-end servers.

Improved operability

Same as for a HiRDB single server configuration.

12.3.5 Design considerations

(1) Considerations common to both a HiRDB single server configuration and HiRDB parallel server configuration

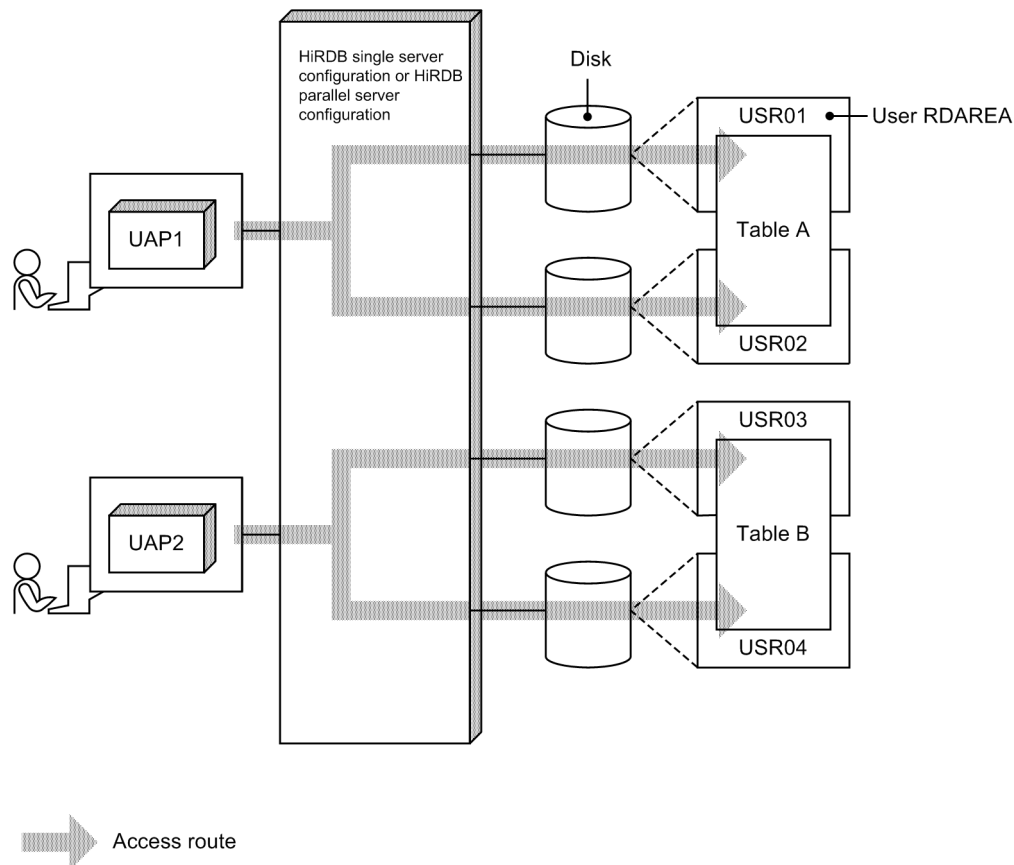
Following are the design considerations that are common to HiRDB single server configurations and HiRDB parallel server configurations:

(a) Row partitioning taking into account contention among disk accesses

If multiple UAPs access separate tables concurrently, these tables should be partitioned and stored in separate user RDAREAs on separate disks.

The following figure provides an overview of row partitioning, taking into account contention among disk accesses.

Figure 12–10: Overview of row partitioning, taking into account contention among disk accesses



Explanation:

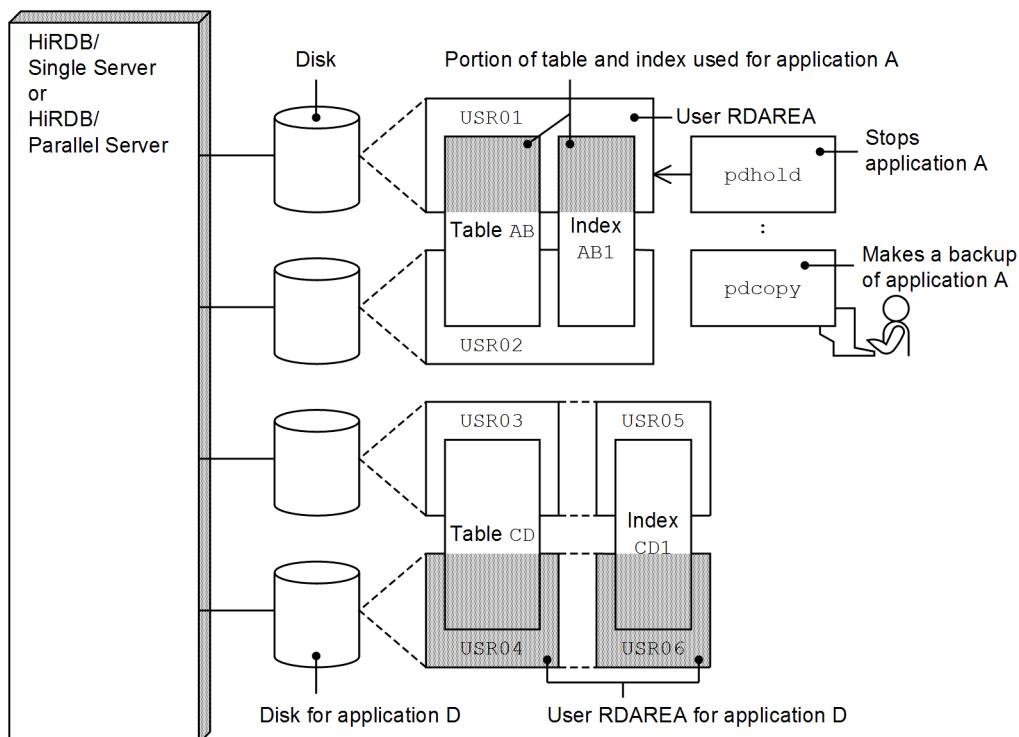
Tables A and B are partitioned and stored in two sets of user RDAREAs, USR01-USR02 and USR03-USR04, which are on separate disks. If UAP1 and UAP2 attempt to access tables A and B concurrently, no contention occurs, thereby reducing their wait time.

If these tables are stored in user RDAREAs on the same disk, access contention occurs on the disk when multiple UAPs attempt to access the tables concurrently. In this case, one of the UAPs is placed in wait status until the other UAP completes its access processing, resulting in an increase in wait time.

(b) Row partitioning taking into account operability

This subsection provides an overview of row partitioning, taking into account operability with reference to the following figure.

Figure 12-11: Overview of row partitioning, taking into account operability



Explanation:

- *Storing table and index in the same user RDAREAs*

If operability for table creation, table reorganization, backing up of user RDAREAs, RDAREA recovery, and other such operations is more important than retrieval performance, a row-partitioned table and its indexes should be stored in the same user RDAREAs. Operations on individual user RDAREAs become easy.

In the example shown in Figure 12-11, the portion of table AB for application A is stored together with its index in dedicated user RDAREA USR01. This enables the `pdhold` command (for shutting down RDAREAs) to be used to terminate application A. Additionally, it simplifies backup processing for each application that uses the database copy utility (`pdcopy`).

- *Placing related user RDAREAs on the same disk*

If a row-partitioned table and its indexes are stored in multiple user RDAREAs, the related user RDAREAs should be placed on the same disk. This enables user RDAREAs to be used individually by disk.

In the example shown in Figure 12-11, the portion of table CD for application D is stored together with its index in user RDAREAs USR04 and USR06 on the same disk. This enables applications to be executed by disk.

(2) HiRDB parallel server configuration

Following are the design considerations for HiRDB parallel server configurations:

(a) Row partitioning taking into account workload for disk accesses

- *Row partitioning over multiple back-end servers*

If multiple user RDAREAs are placed on the disk at one back-end server and the portions of a table stored in individual RDAREAs all have high access frequency, the workload for disk accesses becomes high at this back-end server.

Therefore, a frequently accessed table should be partitioned and stored in user RDAREAs on different disks at multiple back-end servers. In this case, the table should be partitioned so that the table access frequency becomes uniform among all the back-end servers containing the table.

- *Parallel disk accesses over multiple server machines*

If a table stored in a user RDAREA at a back-end server in a server machine with a low CPU workload is accessed mostly for input/output operations, the workload of disk accesses is not uniform among the multiple servers, thereby affecting adversely the efficiency of parallel processing.

If there is capacity in the CPU, more back-end servers and user RDAREAs should be installed in this server to improve the degree of parallel disk access processing.

(b) Row partitioning taking into account the degree of parallel input/output processing

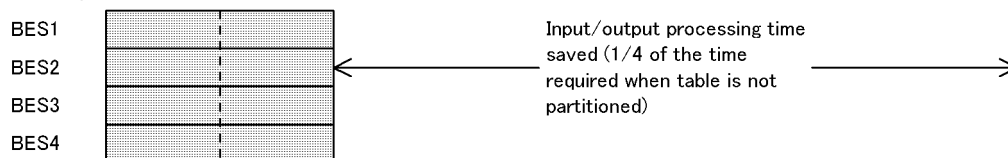
If a table is partitioned and stored in as many server machines as possible, the input/output processing time can be reduced by parallel processing. If there is a limit to the number of server machines available for table row partitioning, the same effects can be achieved by increasing the number of back-end servers and disks for each server machine. The following figure provides an overview of input/output processing performance based on the number of back-end servers used for table row partitioning.

Figure 12–12: Overview of input/output processing performance based on the number of back-end servers used for table row partitioning

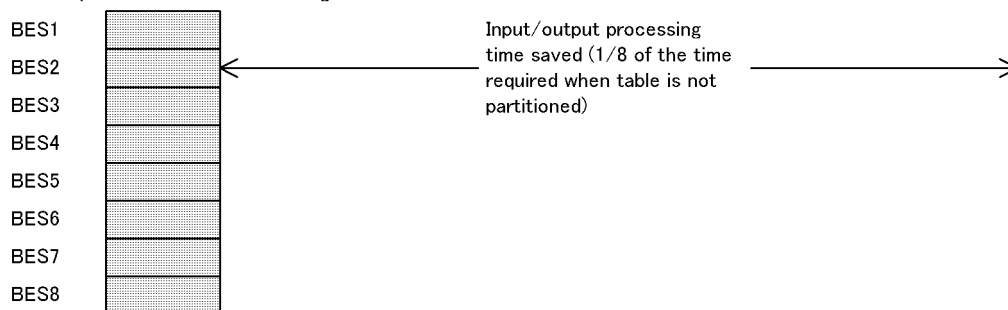
- Input/output processing time when table is not partitioned


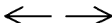


- Table partitioned and stored in four back-end servers



- Table partitioned and stored in eight back-end servers



BES : Back-end server
 : Input/output processing time
 : Input/output processing time saved

If a table is partitioned and stored in too many back-end servers, there is an increase in the amount of communication required to return each back-end server's processing results to the front-end server. Therefore, the appropriate number of back-end servers must be determined, taking into account the type of database operation and SQL processing (whether SQLs are used to retrieve a large amount of data from a large table).

(c) Row partitioning taking into account table access frequency

A table must be partitioned so that table access frequency becomes uniform at each back-end server.

To do this, the considerations discussed below should be taken into account.

Key range partitioning

- When table row partitioning is defined, specify `UNIQUE` for the partitioning key so that the amount of data becomes uniform.
- When a table is partitioned and the number of accesses to the data in a specific key range is expected to be higher than in the other key ranges, divide the data in the heavy-accesses key range by finer key ranges.

Flexible hash partitioning or FIX hash partitioning

- Change the hash function so that the data is distributed uniformly.
- Select a partitioning key without uneven distribution or duplication so that the data is distributed uniformly.

Even when a table is partitioned and stored in multiple back-end servers, the performance of parallel processing of the table can vary depending on whether the table is partitioned so that uniform access frequency can be achieved.

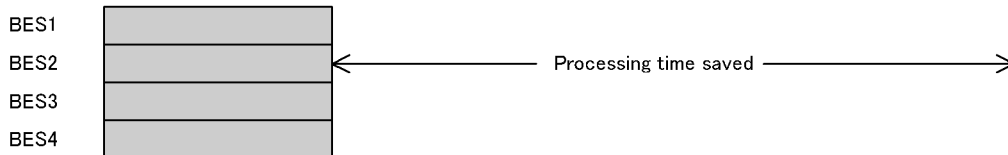
The following figure shows differences in parallel processing performance depending on table access frequency.

Figure 12–13: Performance of parallel processing depending on table access frequency

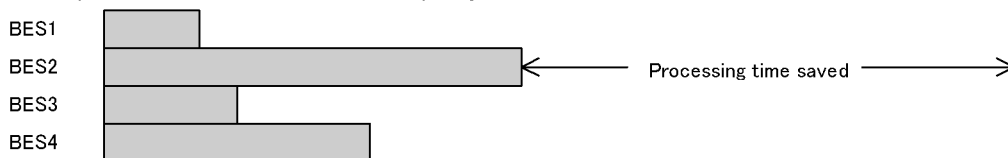
- Processing time when table is not partitioned



- Table partitioned so that uniform access frequency is achieved



- Table partitioned so that uniform access frequency is not achieved



BES : Back-end server

 : Input/output processing time

 : Input/output processing time saved

Explanation:

The processing time that can be saved depends on whether the table is partitioned so that uniform access frequency is achieved. If the access frequency is not uniform, table processing does not terminate until processing at back-end server BES2 is completed, so the benefit of parallel processing is not obtained.

(d) Row partitioning taking into account complicated retrieval processing

For purposes of table partitioning taking into account complicated retrieval processing, such as retrieving or joining a large amount of data, the table should be designed using the following procedure:

1. Determining the disk processing time and the number of disks to be used

Obtain the disk access frequency (utilization factor) from the size of the data and the processing patterns, distribute data to disks on the basis of this value, and determine the object value for disk processing time. If join processing is to be executed, exclude the work disk required for join processing (the number of HiRDB file system areas where work table files are created for sort/merge processing) when distributing data. Exclude the time required for join processing from the object value of disk processing time. Determine the number of disks on the basis of data distribution to disks.

2. Determining the number of server machines

Obtain the overhead time for processing at the server machines on the basis of the data processing patterns. Determine the number of server machines (where back-end servers are installed) so that the disk processing time and overhead time at the server machines become uniform.

3. Determining the number of server machines used for join processing

Obtain the overhead time for join processing at the server machines on the basis of the data processing patterns. Then, determine the number of floating machines on the basis of this value and the disk processing time.

A floating machine is a server machine where a floating server is installed, which is a back-end server dedicated to complicated retrieval processing such as join processing. User RDAREAs cannot be allocated to a back-end server defined as a floating server.

4. Determining the number of work disks

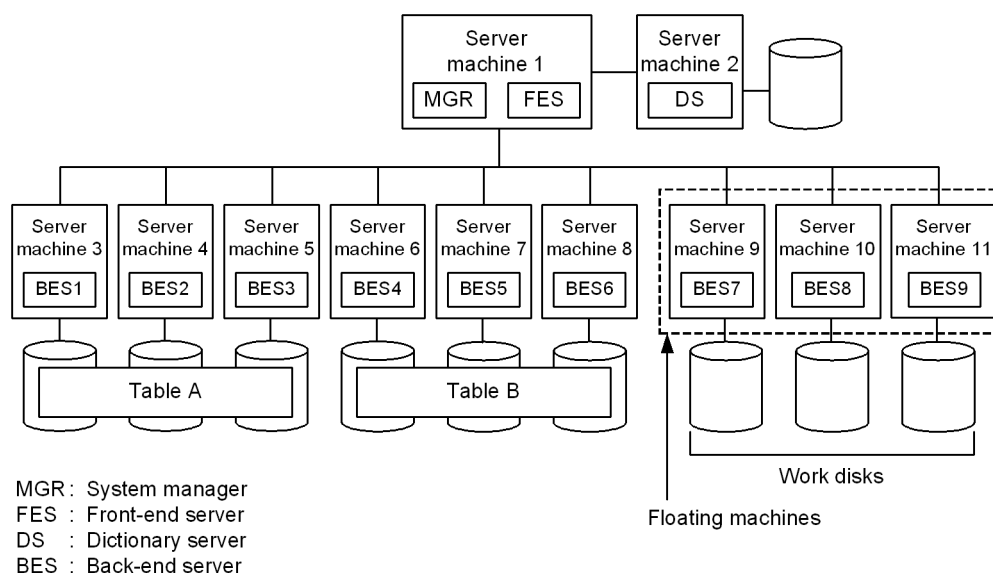
The data subject to join processing is distributed uniformly from each back-end server to the floating server. Determine the number of work disks (number of HiRDB file system areas used to create work table files) on the basis of the expected amount of this data.

5. Determining the system configuration

Determine the overall system configuration on the basis of the numbers of server machines and disks determined above.

The following figure shows an overview of a system configuration involving table row partitioning, designed using the above procedure and taking into account complex retrieval processing.

Figure 12–14: System configuration involving table row partitioning, taking into account complex retrieval processing



Note: BES7-BES9 are floating servers.

Explanation:

Back-end servers BES1-BES3 and BES4-BES6 read the data subject to join processing from Tables A and B, respectively. Floating servers BES7-BES9 receive data from back-end servers BES1-BES6 and execute parallel match processing.

This type of system configuration can reduce the workload of back-end servers BES1-BES6 and reduce processing time. If no floating server is installed, one of the back-end servers BES1-BES6 must execute the join processing.

12.3.6 Notes on table row partitioning

1. When a table is row-partitioned, its indexes must also be row-partitioned. If a table is partitioned and stored in multiple RDAREAs while an index is stored in a single user RDAREA, the level of concurrent execution may be reduced due to locking on the index. For details about index row partitioning, see *13.3 Index row partitioning*.
2. You should use the hash facility for hash row partitioning in the following cases:
 - Hash row partitioning is to be used with the table.
 - The amount of data is expected to increase.

If you add RDAREAs to handle an increase of data for a hash row partitioning table (if you increase the number of table row partitions), data may become uneven among the existing RDAREAs and newly added RDAREAs. The rebalancing facility for hash row partitioning can correct such unevenness of data when you increase the number of table row partitions. For details about the rebalancing facility for hash row partitioning, see the *HiRDB Version 9 System Operation Guide*.

12.4 Table matrix partitioning

Partitioning a table by a combination of partitioning methods using two of the table columns as the partitioning key is called *matrix partitioning*. The first column used as the partitioning key is called the *first dimension partitioning column*, and the second column used as the partitioning key is called the *second dimension partitioning column*. Matrix partitioning involves key range partitioning with boundary values specified for the first dimension partitioning column and then partitioning the resulting data further by the second dimension partitioning column. The following partitioning methods can be specified for the second dimension partitioning column:

- Key range partitioning with boundary values specified
- Flexible hash partitioning[#]
- FIX hash partitioning[#]

[#]

You can specify hash functions HASH0 to HASH6. You cannot specify HASHA to HASHF.

A table that has been matrix partitioned is called a *matrix-partitioned table*.

In order to matrix partition tables, HiRDB Advanced High Availability is required.

(1) Effects of table matrix partitioning

The effectiveness of partitioning on the basis of partitioning keys formed from multiple columns is as follows:

- High-speed SQL processing
High-speed SQL processing can be obtained by parallel execution of SQL processing and by maximizing the retrieval range by retrieving on the basis of multiple keys.
- Reduced operating time
More precise partitioning makes it possible to reduce the size of an RDAREA. This reduces the time required for reorganization, for making backups, and for error recovery.

(2) Criteria

We recommend using key range partitioning with boundary values specified for both partitioning columns when the following conditions are met:

- Data was partitioned by the first dimension partitioning column, and there is a large amount of data corresponding to the various boundary values.
- Multiple columns need to be specified in the search condition for a UAP that accesses the table and you wish to limit the RDAREAs that are accessed by multiple columns. Alternatively, you wish to limit the RDAREAs that are accessed only by column *n* specified in the SQL statement.

We recommend that you combine key range partitioning with boundary values specified and hash partitioning when the following conditions are met:

- Data was partitioned by the first dimension partitioning column, and there is a large amount of data corresponding to the various boundary values.
- You wish to uniformly segment the range of data that was partitioned by the first dimension partitioning column.

In the following cases, we recommend duplicative specification of the RDAREA name specified in hash partitioning when key range partitioning with boundary values is combined with hash partitioning. Specifying the same RDAREA more than once can reduce the number of RDAREAs actually used while keeping the number of partitions unchanged. The amount of data stored in each RDAREA can also be kept even. For details about duplicative specification of RDAREA names, see *Figure 12-4 Example of duplicating a table storage RDAREA specification using hash partitioning*.

- You want to keep the re-organization processing time constant for each RDAREA by keeping the size of individual RDAREAs constant, but it is difficult to specify a boundary value that will divide data amounts evenly when you are partitioning using first dimension partitioning column boundary values. As a result, processing time cannot be kept constant.

- You want to reduce the number of partitions because this affects retrieval performance.

(3) Specification

You use the `CREATE TABLE` definition SQL statement with the `PARTITIONED BY MULTIDIM` operand to specify the following:

- The table's allocation to RDAREAs
- The matrix partitioning parameters (partitioning key, partitioning method)

The definition rules are as follows:

- When key range partitioning with boundary values specified is specified for the second dimension partitioning column
 - You can specify two partitioning keys (partitioning key for the first dimension partitioning column + partitioning key for the second dimension partitioning column). The partitioning keys for the first and second dimension partitioning columns cannot be the same.
- When hash partitioning is specified for the second dimension partitioning column
 - You can specify 2-16 partitioning keys. When flexible hash partitioning is used, only the partitioning key for the second dimension partitioning column can be updated. When FIX hash partitioning is used, the partitioning keys cannot be updated.

For an example definition, see (4) *Matrix partitioning example*.

(4) Matrix partitioning example

(a) Combination of key range partitioning with boundary values specified

Boundary values are specified for the `registration_date` and `store_number` columns of the `CUSTOMER_TABLE`, and the table is matrix-partitioned by `registration_date` and `store_number`. The customer data is stored in user RDAREAs (USR01 to USR06) as shown below. The number of user RDAREAs required for storage, based on the formula $(\text{number of boundary values} + 1) \times (\text{number of boundary values} + 1)$, is $3 \times 2 = 6$ in this example:

Registration date	Store number	
	100 or below	Above 100
2000 or earlier	USR01	USR02
2001	USR03	USR04
2002 or later	USR05	USR06

The following is the SQL statement to define this matrix-partitioned table:

```
CREATE FIX TABLE CUSTOMER_TABLE
  (registration_date DATE, store_number INT, customer_name NCHAR(10))
  PARTITIONED BY MULTIDIM(
    registration_date (('2000-12-31'),
                      ('2001-12-31')),
    store_number ((100))
  ) IN ((USR01, USR02), (USR03, USR04), (USR05, USR06))
```

Explanation

1. Specifies the name of the first dimension partitioning column (name of the first column that is used as the partitioning key) and its list of boundary values.
2. Specifies the name of the second dimension partitioning column (name of the second column that is used as the partitioning key) and its list of boundary values.

The following figure shows an example of matrix partitioning.

Figure 12–15: Example of matrix partitioning (combination of key range partitioning with boundary values specified)

Customer table		
Registration date	Store number	Customer name
1999-10-31	001	Jones, James
2000-01-25	110	Stevens, Kim
2001-06-30	005	Berry, Richard
2001-12-24	120	Taylor, Marion
2002-01-07	050	Smith, John
2002-06-13	130	Black, Fred
2002-08-24	099	Powell, Tom

↑
First
partitioning
key
↑
Second
partitioning
key

- Matrix partitioning table stored on USR01

Registration date	Store number	Customer name
1999-10-31	001	Jones, James
- Matrix partitioning table stored on USR02

Registration date	Store number	Customer name
2000-01-25	110	Stevens, Kim
- Matrix partitioning table stored on USR03

Registration date	Store number	Customer name
2001-06-30	005	Berry, Richard
- Matrix partitioning table stored on USR04

Registration date	Store number	Customer name
2001-12-24	120	Taylor, Marion
- Matrix partitioning table stored on USR05

Registration date	Store number	Customer name
2002-01-07	050	Smith, John
2002-08-24	099	Powell, Tom
- Matrix partitioning table stored on USR06

Registration date	Store number	Customer name
2002-06-13	130	Black, Fred

(b) Combination of key range partitioning with boundary values specified and hash partitioning

This subsection describes an example of applying FIX hash partitioning to a second dimension partitioning column.

This example matrix-partitions the CUSTOMER_TABLE by specifying boundary values for registration_date and using a hash function to partition store_number and region_code into three segments. The customer data is stored in user RDAREAs (USR01 to USR09) as shown below. The number of RDAREAs needed for storage is $(\text{number of boundary values} + 1) \times (\text{desired partitions to be obtained by hash function})$; therefore, $3 \times 3 = 9$ RDAREAs are needed for this example.

Registration date	Store number and region code (divided into 3 partitions by hash function)		
2002 or earlier	USR01	USR02	USR03
2003	USR04	USR05	USR06
2004 or later	USR07	USR08	USR09

The following SQL statement defines the table to be matrix-partitioned:

```
CREATE FIX TABLE CUSTOMER_TABLE
  (registration_date DATE, store_number INT, region_code INT, customer_name
  NCHAR(10))
  PARTITIONED BY MULTIDIM
  (registration_date (('2002-12-31'),('2003-12-31')), ...1.
  FIX HASH HASH6 BY store_number, region_code ...2.
  ) IN ((USR01,USR02,USR03),
```

```
(USR04,USR05,USR06) ,
(USR07,USR08,USR09) )
```

Explanation:

1. Specifies the name of the first dimension partitioning column (name of the first column to be used as the partitioning key) and its list of boundary values.
2. Specifies the name of the second dimension partitioning column (name of the second column that is used as the partitioning key) and the hash function name.

The following figure shows another example of matrix partitioning.

Figure 12–16: Example of matrix partitioning (combination of key range partitioning with boundary values specified and hash partitioning)

Customer table			
Registration date	Store number	Region code	Customer name
2002-01-31	001	01	James Smith
2002-05-25	110	03	John Johnson
2002-06-30	005	01	Robert Williams
2003-01-24	120	03	Michael Jones
2003-03-07	050	01	William Brown
2003-04-13	130	03	David Miller
2003-08-24	099	02	Richard Davis
2003-11-15	010	01	Mary Jackson
2003-12-24	020	01	Charles Wilson
2004-01-27	080	02	Patricia White
2004-03-24	170	04	Linda Harris
2004-07-24	015	01	Thomas Taylor

↑	↑	↑	
First	Second	Second	
partitioning	partitioning	partitioning	
key	key	key	

• Matrix partitioning table stored in USR01			
Registration date	Store number	Region code	Customer name
2002-01-31	001	01	James Smith

• Matrix partitioning table stored in USR02			
Registration date	Store number	Region code	Customer name
2002-05-25	110	03	John Johnson

- Matrix partitioning table stored in USR03

Registration date	Store number	Region code	Customer name
2002-06-30	005	01	Robert Williams

- Matrix partitioning table stored in USR05

Registration date	Store number	Region code	Customer name
2003-03-07	050	01	William Brown
2003-11-15	010	01	Mary Jackson

- Matrix partitioning table stored in USR07

Registration date	Store number	Region code	Customer name
2004-01-27	080	02	Patricia White

- Matrix partitioning table stored in USR09

Registration date	Store number	Region code	Customer name
2004-07-24	015	01	Thomas Taylor

- Matrix partitioning table stored in USR04

Registration date	Store number	Region code	Customer name
2003-01-24	120	03	Michael Jones
2003-08-24	099	02	Richard Davis

- Matrix partitioning table stored in USR06

Registration date	Store number	Region code	Customer name
2003-04-13	130	03	David Miller
2003-12-24	020	01	Charles Wilson

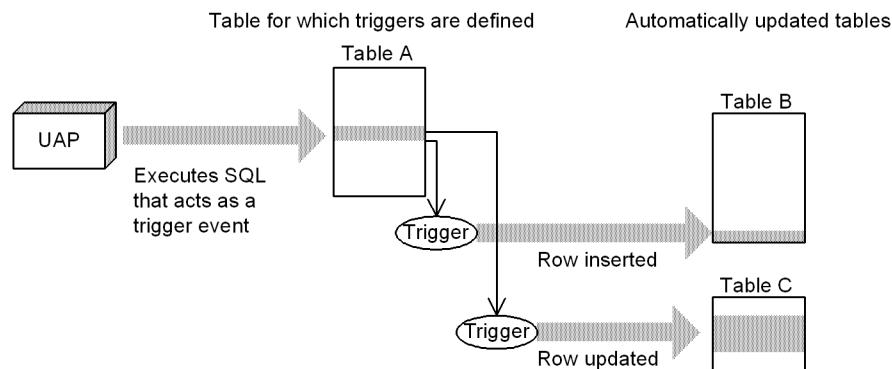
- Matrix partitioning table stored in USR08

Registration date	Store number	Region code	Customer name
2004-03-24	170	04	Linda Harris

12.5 Defining a trigger

By defining a trigger, an SQL statement can be executed automatically in response to some operation on a table (updating, insertion, deletion). A trigger specification involves a table, an SQL statement that serves as the event to activate the trigger (the *trigger event*), an automatically executed SQL statement (*trigger event SQL*), and the conditions under which the trigger is activated (the trigger action search conditions). When an SQL statement that matches the trigger action search conditions is executed on a table for which a trigger has been defined, the triggered SQL statement executes automatically. The following figure provides an overview of triggers.

Figure 12–17: Overview of triggers



Explanation:

When the UAP executes a trigger event SQL statement, triggers defined for Table A are called. If trigger action search conditions are satisfied, triggered SQL statements are executed automatically (in this case, a row is inserted into Table B and a row is updated in Table C).

Prerequisite:

Before you define a trigger, you need to create an RDAREA for the data dictionary LOB. Use the database structure modification utility (pdmod) to create the RDAREA for the data dictionary LOB.

When a trigger is defined for a table, all existing functions, procedures, and trigger SQL objects that the table uses become invalid and have to be re-created. The trigger SQL object will also become invalid and will have to be re-created if any of the resources used by the trigger (such as tables or indexes) are defined, modified, or deleted. For details, see *12.5.4 Trigger management*.

12.5.1 Application standards

Triggers are recommended when a UAP performs the following processing:

- A table must be updated whenever another table is updated
- Whenever a table is updated, columns in the updated row must be updated (the columns are related)

For example, assume that whenever a price is updated in the product management table the change has to be recorded also in the product management history table. If a trigger were not used, it would be up to the UAP to always update the product management history table whenever it updates the product management table. If a trigger is used, however, the product management history table would be updated automatically, which means that the UAP that updates the product management table does not have to even be aware of the product management history table. In this way, when triggers are applied appropriately, the burden on UAP developers is lessened.

12.5.2 Defining a trigger

(1) Preparation for definition

When a trigger is defined, the SQL objects of the trigger action are created automatically on the basis of the triggered SQL statement and are stored in a data dictionary LOB RDAREA. Therefore, when triggers are to be defined, sufficient space must be available in the data dictionary LOB RDAREA. For details about estimating the size of the data dictionary LOB RDAREA, see *16.5 Determining the size of a data dictionary LOB RDAREA*.

In order to execute a trigger event SQL statement, you must take into account the triggered SQL objects when you specify the SQL object buffer length. For details about estimating the buffer length used by SQL objects, see the manual *HiRDB Version 9 System Definition*.

(2) Definition method

The following definition SQLs are used to define triggers and to re-create and delete SQL objects.

- **CREATE TRIGGER**

This statement defines a trigger. Triggers can be defined only for tables that are owned by the definer; they cannot be created for tables that are owned by other users. This statement specifies the following items:

- **Timing of the trigger action**

You can execute the trigger action either before (**BEFORE**) or after (**AFTER**) table manipulation. A trigger whose trigger action time is **BEFORE** is called a **BEFORE** trigger, and a trigger whose trigger action is **AFTER** is called an **AFTER** trigger.

- **Trigger event**

The events that can cause a trigger action are the **INSERT**, **DELETE**, and **UPDATE** statements.

- **Table for which trigger is defined**

A trigger can be defined only for a base table.

- **Row alias before and after execution of the trigger event SQL statement (old or new values alias)**

For the row that is updated by the trigger event SQL statement, specify the name before SQL statement execution (old correlation name) or the name after SQL statement execution (new values correlation name). You can use these aliases to specify the details of the trigger action.

- **Trigger action**

There are three factors of the trigger action:

- The triggered SQL statement (the SQL statement that executes automatically)
- The trigger action search conditions (the conditions under which the triggered SQL statement is executed)
- Whether the action is executed at the row level or the statement level

A triggered SQL statement is executed only when the trigger action search conditions are satisfied. If no conditions are provided, the triggered SQL statement is executed every time the trigger event SQL statement executes.

- **ALTER TRIGGER**

This statement re-creates the SQL object for a trigger that has already been defined. The **ALTER ROUTINE** definition SQL statement can also be used for re-creation.

- **DROP TRIGGER**

This statement deletes a trigger.

(3) Trigger definition example

(a) Example of using a trigger

The following is an example of defining a trigger for the product management table so that if there is an increase in the value of the Price column that exceeds 10,000 yen, the pre-update and post-update prices will be inserted into the product management history table.

```
CREATE TRIGGER TR1      ...Trigger name
  AFTER                ...Timing of the trigger action
```

```

UPDATE OF price      ...Triggering event
ON product_management_table...Table the trigger is defined for
REFERENCING OLD ROW AS X1      ...Pre-update row alias
NEW ROW AS Y1      ...Post-update row alias
FOR EACH ROW      ...Whether for the entire statement or for each row
WHEN(Y1.price - X1.price > 10000)
      ...Trigger action search condition

INSERT INTO product_management_history_table VALUES      ...Triggered SQL
statement
      (X1.item_no, X1.price, Y1.price)

```

- Product management table

Item No.	Product name	Price
180	Television	350.00
220	Video	200.00
250	Amp	800.00
300	Speakers	750.00
⋮	⋮	⋮

- Product management history table

Item No.	Pre-update	Post-update
180	200.00	350.00

UPDATE product_management_table SET Price = 950.00 WHERE Number = 300

180	Television	350.00
220	Video	200.00
250	Amp	800.00
300	Speakers	950.00
⋮	⋮	⋮

180	200.00	350.00
300	750.00	950.00

} Added by trigger

UPDATE product_management_table SET Price = 850.00 WHERE Number = 250

180	Television	350.00
220	Video	200.00
250	Amp	850.00
300	Speakers	950.00
⋮	⋮	⋮

180	200.00	350.00
300	750.00	950.00

} Not updated, as trigger action condition not met

(b) Example of a trigger action that uses an SQL control statement (assignment statement)

An assignment statement is an SQL statement that assigns a specified value to a specified column. A trigger can use an assignment statement before executing its action on a table. When an assignment statement is used in a trigger action, a relationship can be established between columns.

The following example shows two trigger definitions and their actions of updating the value of the *Bonus* column in response to updates of the value in the *Position* column of the staff table.

- A trigger that sets a salary bonus value for a row that is inserted into the staff table; the bonus is set at 8% if the Position is A, 10% if the Position is B, and 0% otherwise.

```

CREATE TRIGGER bonus_trigger_1
BEFORE
INSERT
ON staff_table
REFERENCING NEW ROW AS X1
FOR EACH ROW
SET X1.bonus=CASE X1.position
                WHEN 'A' THEN X1.salary*0.08
                WHEN 'B' THEN X1.salary*0.1
                ELSE 0 END

```

- A trigger that sets a salary bonus value for a row in the staff table in response to a change in the *Position* or *Salary* column; the bonus is set at 8% if the Position is A, 10% if the Position is B, and 0% otherwise

```

CREATE TRIGGER bonus_trigger_2
BEFORE

```



```

UPDATE OF position, salary
ON staff_table
REFERENCING NEW ROW AS X1
FOR EACH ROW
SET X1.bonus=CASE X1.position
                WHEN 'A' THEN X1.salary*0.08
                WHEN 'B' THEN X1.salary*0.1
                ELSE 0 END

```

- Staff table

Employee number	Name	Position	Salary	Bonus
S99245	Stevens, Kim	A	1900	152
L98003	Jones, James	B	2200	220

```
INSERT INTO staff_table VALUES ('R97023','Tom Powell','A','180000,0)
```

Post-insert

S99245	Stevens, Kim	A	1900	152
L98003	Jones, James	B	2200	220
R97023	Powell, Tom	A	1800	144

```
UPDATE staff_table SET position='C' WHERE Employee-
Number='S99245'
```

Post-update

S99245	Stevens, Kim	C	1900	0
L98003	Jones, James	B	2200	220
R97023	Powell, Tom	A	1800	144

Explanation

The INSERT statement acts as a triggering event, `bonus_trigger_1` is executed, and then a row is added. The INSERT statement causes the data in Bonus to be set to 0, and then the result of the assignment is stored.

Next, the UPDATE statement acts as a triggering event, `bonus_trigger_2` is executed, and the data in Bonus is updated to 0.

(c) Example where the triggering action uses SQL control statements (a compound statement)

A compound statement is an SQL statement that executes multiple SQL statements within a single statement. An update to a table can act as a triggering event, such that the triggered SQL statement is a compound statement that enables the single trigger to update multiple tables.

The following example defines a trigger that enables updates to the master inventory table to be reflected in the Glasgow inventory table and the Edinburgh inventory table. If a compound statement were not used, it would be necessary to define two separate triggers.

```

CREATE TRIGGER local_stock_table_update_trigger
AFTER
UPDATE OF stock_count
ON inventory_master
REFERENCING NEW ROW post_update
              OLD ROW pre_update
BEGIN
    UPDATE Glasgow_stock SET stock_count=post_update.stock_count
    WHERE product_code=pre_update.product_code;
    UPDATE Edinburgh_stock SET stock_count=post_update.stock_count
    WHERE product_code=pre_update.product_code;
END

```

(d) Example where the trigger action contains an SQL diagnostic statement (SIGNAL statement)

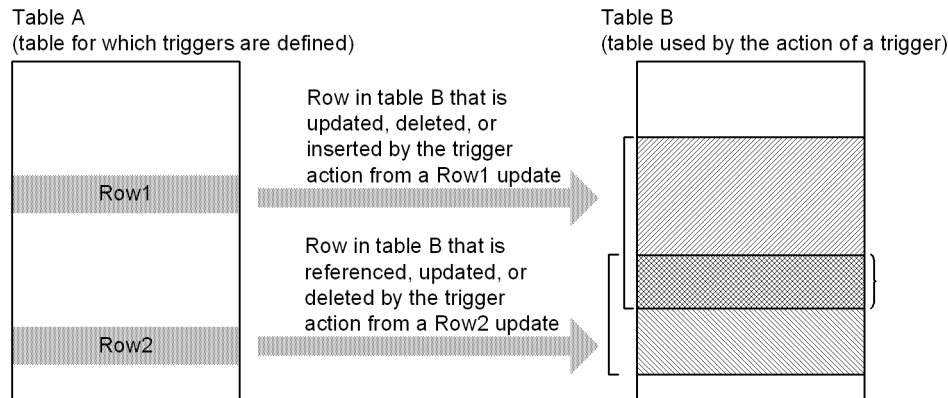
The SIGNAL statement causes an error to occur. If, before the action on a table, a trigger action is executed that specifies the SIGNAL statement, then if the action is invalid the SIGNAL statement will execute to prevent the action.

The following example defines a trigger where there is an attempt to update information for someone else, and before the staff information table is updated the SIGNAL statement issues an error and the update is prevented.

```
CREATE TRIGGER update_prevention_trigger
BEFORE
UPDATE
ON staff_information
REFERENCING OLD ROW AS X1
WHEN (X1.employee_name<>USER) SIGNAL SQLSTATE '99001'
```

12.5.3 Trigger considerations

Depending on the definition of row-level triggers, when a trigger event SQL statement is executed, the result may vary according to the internal HiRDB processing (different contents after an update).



Explanation

When row 1 of Table A is updated, a trigger causes a row to be updated, deleted, or inserted in Table B. When row 2 is updated, a trigger causes the same row in Table B to be referenced, updated, or deleted (there is an overlapping portion). The update sequence between row 1 and row 2 depends on internal HiRDB processing, such that a different result may occur.

12.5.4 Trigger management

(1) Trigger definition

When a trigger is defined, all existing functions, procedures, and trigger SQL objects that the table uses become invalid and have to be re-created. By referencing the `SQL_ROUTINE_RESOURCES` dictionary table before a trigger is defined, you can check the functions, procedures, and trigger SQL objects that will become invalid. Check the SQL objects that will become invalid so that you can re-create them.

(a) Checking the functions, procedures, and trigger SQL objects that will become invalid when a trigger is defined

The following example shows how to check the functions, procedures, and trigger SQL objects that will become invalid when a trigger definition is defined. In the case of a trigger, what will become invalid is the trigger identifier (`TRIGGER_NAME`). In the case of functions and procedures, `TRIGGER_NAME` becomes NULL.

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME,
A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE='R'
AND B.BASE_OWNER='authorization-identifier-of-owner-of-table-for-which-trigger-
is-defined'
AND B.BASE_NAME='table-for-which-trigger-is-defined'
AND (B.column-name#='Y'
OR ( B.INSERT_OPERATION IS NULL
AND B.UPDATE_OPERATION IS NULL
AND B.DELETE_OPERATION IS NULL))
```

#: To retrieve the SQL objects that will become invalid when a trigger is defined with INSERT as the triggering event, specify INSERT_OPERATION as the column name; when UPDATE is the triggering event, specify UPDATE_OPERATION; and when DELETE is the triggering event, specify DELETE_OPERATION.

(2) Re-creating a trigger SQL object

When a table, index, or other resource already used by a trigger is defined, modified, or deleted, the trigger SQL objects become invalid. Also, defining or deleting indexes for a table that a trigger is using will cause the index information for the trigger SQL objects to become invalid.

If a trigger SQL object becomes invalid, or if the SQL object's index information becomes invalid, the trigger event SQL statement cannot be executed. To prevent a trigger SQL object or the SQL object's index information from becoming invalid, the trigger SQL objects must be re-created with the ALTER TRIGGER or ALTER ROUTINE definition SQL statements.

(a) How to check the resources used by a trigger

You can check information on the resources that a trigger is using by referencing the SQL_ROUTINE_RESOURCES, SQL_TRIGGER_USAGE, and SQL_ROUTINE_PARAMS dictionary tables.

- SQL example for checking the resources used by a trigger action condition

```
SELECT B.* FROM MASTER.SQL_TRIGGERS A, MASTER.SQL_TRIGGER_USAGE B
WHERE A.TRIGGER_SCHEMA='schema-name'
      AND A.TRIGGER_NAME='trigger-identifier'
      AND A.TRIGGER_SCHEMA=B.TRIGGER_SCHEMA
      AND A.TRIGGER_NAME=B.TRIGGER_NAME
```

- SQL example for checking the column resources used by a trigger in specifying old and new value aliases

```
SELECT B.* FROM MASTER.SQL_TRIGGERS A, MASTER.SQL_ROUTINE_PARAMS B
WHERE A.TRIGGER_SCHEMA='schema-name'
      AND A.TRIGGER_NAME='trigger-identifier'
      AND A.TRIGGER_SCHEMA=B.ROUTINE_SCHEMA
      AND A.SPECIFIC_NAME=B.SPECIFIC_NAME
```

- SQL example for checking resources other than the above used by a trigger

```
SELECT B.* FROM MASTER.SQL_TRIGGERS A, MASTER.SQL_ROUTINE_RESOURCES B
WHERE A.TRIGGER_SCHEMA='schema-name'
      AND A.TRIGGER_NAME='trigger-identifier'
      AND A.TRIGGER_SCHEMA=B.ROUTINE_SCHEMA
      AND A.SPECIFIC_NAME=B.SPECIFIC_NAME
```

(b) How to check triggers that will be deleted before deleting columns in a table

If all columns that act as triggering events are deleted, the trigger will be deleted. The following is an example of an SQL for checking the triggers that will be deleted before deleting columns from a table:

```
SELECT A.TRIGGER_SCHEMA, A.TRIGGER_NAME
FROM MASTER.SQL_TRIGGERS A
WHERE A.N_UPDATE_COLUMNS>0
      AND A.TABLE_SCHEMA='authorization-identifier-of-owner-of-table-from-which-
columns-will-be-deleted'
      AND A.TABLE_NAME='table-identifier-of-table-from-which-columns-will-be-
deleted'
      AND NOT EXISTS(SELECT * FROM MASTER.SQL_TRIGGER_COLUMNS B
                     WHERE B.TRIGGER_SCHEMA=A.TRIGGER_SCHEMA
                           AND B.TRIGGER_NAME=A.TRIGGER_NAME
                           AND B.TABLE_SCHEMA=A.TABLE_SCHEMA
                           AND B.TABLE_NAME=A.TABLE_NAME
                           AND B.COLUMN_NAME NOT IN('name-of-column-to-be-
deleted', ...))
```

(c) How to check the functions, procedures, and trigger SQL objects or SQL object index information that will become invalid before defining, modifying, or deleting a table or index

The following is an SQL example of checking for the functions, procedures, and trigger SQL objects or SQL object index information that will become invalid before defining, modifying, or deleting a table or index. If a trigger will become invalid, the trigger identifier (TRIGGER_NAME) is obtained. If it is a function or a procedure, the value of TRIGGER_NAME becomes NULL.

- Table (including view table) modification or deletion, or index definition (specify the schema name and identifier of the table that defines the index)

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME,
A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE IN('R','V')
AND B.BASE_OWNER='table(view-table)-owner-authorization-identifier'
AND B.BASE_NAME='table(view-table)-identifier'
```

- Index deletion

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME,
A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE='I'
AND B.BASE_OWNER='index-owner-authorization-identifier'
AND B.BASE_NAME='index-identifier'
```

- Function or procedure deletion

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME,
A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE='P'
AND B.BASE_OWNER='function(procedure)-owner-authorization-identifier'
AND B.BASE_NAME='routine-identifier'
```

- Trigger deletion

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME,
A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE='T'
AND B.BASE_OWNER='trigger-owner-authorization-identifier'
AND B.BASE_NAME='trigger-identifier'
```

- Schema deletion

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME,
A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_OWNER='schema-name'
```

- Deletion of user-defined type

```
SELECT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_NAME='identifier-of-data-type-to-be-deleted'
AND B.BASE_TYPE='D'
UNION
SELECT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINES C INNER JOIN MASTER.SQL_ROUTINE_RESOURCES B
ON C.SPECIFIC_NAME=B.BASE_NAME
LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE C.ROUTINE_ADT_OWNER='owner-authorization-identifier-of-user-defined-
type-to-be-deleted'
AND C.ROUTINE_ADT_NAME='type-identifier-of-user-defined-type-to-be-deleted'
AND B.BASE_TYPE='P'
```

- (d) How to check the functions, procedures, and trigger SQL objects or SQL object index information that has become invalid as a result of defining, modifying, or deleting a table or index

To check the trigger SQL objects or SQL object index information that have become invalid because of definition, modification, or deletion of a table or index, refer to the `TRIGGER_VALID` and `INDEX_VALID` columns of the `SQL_TRIGGER` dictionary table. If the entry in the `TRIGGER_VALID` column is `N`, the trigger SQL object has become invalid. If the entry in the `INDEX_VALID` column is `N`, the index information of that trigger SQL object has become invalid.

The following is an SQL example of checking for the functions, procedures, and trigger SQL objects and SQL object index information that has become invalid because of definition, modification, or deletion of a table or index. If a trigger has become invalid, the trigger identifier (`TRIGGER_NAME`) is obtained. For functions and procedures, the value of `TRIGGER_NAME` becomes `NULL`.

```
SELECT 'TRIGGER', TRIGGER_SCHEMA AS "SCHEMA", TRIGGER_NAME AS "NAME",
       TRIGGER_VALID AS "OBJECT_VALID", INDEX_VALID
FROM MASTER.SQL TRIGGERS
WHERE TRIGGER_VALID='N' OR INDEX_VALID='N'
UNION
SELECT 'ROUTINE', ROUTINE_SCHEMA, ROUTINE_NAME, ROUTINE_VALID, INDEX_VALID
FROM MASTER.SQL ROUTINES
WHERE ROUTINE_VALID='N' OR INDEX_VALID='N'
```

12.5.5 Error recovery

Trigger source code is stored in a data dictionary `RDAREA`, and trigger SQL objects are stored in a data dictionary `LOB RDAREA`. The log collection mode for the data dictionary `RDAREA` is `ALL`, and the log collection mode for the data dictionary `LOB RDAREA` is `PARTIAL`. Therefore, if an error occurs, the source code can be recovered to its most recent status from the backup and the log. The SQL objects, however, can only be recovered to their status at the time of the most recent backup. Therefore, the following considerations are important:

- Always have a recent backup on hand
Make frequent backups of the data dictionary `LOB AREA`, so that if an error occurs you can recover from a recent backup. Use the `pdcopy` command with `-M x` or `-M r` specified.
For details about how to make backups, see the *HiRDB Version 9 System Operation Guide*.
- Re-create the trigger SQL objects
If you do not have a recent backup of the data dictionary `LOB RDAREA`, use the `pdmod` command to reinitialize the data dictionary `LOB RDAREA`. Then execute `ALTER ROUTINE` with `ALL` specified, which will re-create all trigger SQL objects.

12.6 Creating a view table

Tables can be classified into base tables and view tables. A base table is an actual table. A view table is a virtual table defined by selecting rows and columns from the base table.

(1) Relationship between base tables and view tables

The following figure shows the relationship between a base table and a view table.

Figure 12–18: Relationship between a base table and a view table



Explanation:

This example uses base table `STOCK` to create view table `VSTOCK`, which consists of the product code (`PCODE`), stock quantity (`SQUANTITY`), and unit price (`PRICE`) columns for those rows with `SOCKS` in the product name (`PNAME`) column.

Let's assume that a branch office needs to reference only the three data items product code, stock quantity, and unit price for the products whose product name is `SOCKS`. For this purpose, base table `STOCK` is set to be inaccessible, and view table `VSTOCK` is set to be accessible for referencing purposes only (`SELECT` privilege). In this way, data can be protected while allowing necessary information to be referenced.

(2) Effects of creating view tables

The effects of creating view tables are discussed as follows.

Improved security

To improve security for a specific table, the table should be used as a base table and view tables should be created from it. Doing this enables only selected columns and rows to be disclosed. Row and column levels of security can be achieved by granting access privileges to the view tables only.

Improved operability

- If a table is retrieved on the basis of specifying a complicated query, a view table that contains the data obtained from that query should be created, so that there is no need to issue the complicated query again. This simplifies table referencing operations.
- A view table can be used to reference or update its base table. As a result, when the base table definition is modified, there is no need to modify the SQL statements or the view table definition depending on the nature of the modification.

(3) How to create view tables

View tables are created with the `CREATE VIEW` definition SQL statement. The `CREATE VIEW` statement can define the following view tables:

- View tables made from selected rows and columns of base tables

- View tables with columns determined from set functions, date operations, time operations, concatenation operations, scalar functions, or arithmetic operations performed on values from columns of base tables
- One view table based on a maximum of 64 base tables
- View tables based on the result of grouping retrieval
- View tables based on base tables that are owned by other users (limited to base tables or owned by other users and for which the `SELECT` privilege has been granted)

Rules

1. A single view table can be defined with up to 30,000 columns.
2. Columns cannot be added to a view table, and indexes cannot be defined.
3. The owner of a view table defined from base tables owned by that user holds all privileges (row retrieval, add, delete, update) to that view table.
4. The owner of a view table defined from base tables owned by another user holds the same privileges that he or she holds for those base tables. However, if the view table definition has any of the following definitions, only row retrieval is allowed, regardless of whether the security facility is used:
 - View tables for which the columns contain multiple specifications of the same columns from base tables
 - View tables for which the columns contain the results of literals; the `USER`, `CURRENT_DATE`, and `CURRENT_TIME` value functions; arithmetic operations; date operations; time operations; concatenation operations; or scalar functions
 - Multiple base tables have been specified
 - `DISTINCT`, set functions (`COUNT (*)`, `AVG`, `MAX`, `MIN`, `SUM`), grouping (`GROUP BY` clause), or group conditions (`HAVING` clause) have been specified

If the security facility has not been used, view tables other than those noted above can be freely updated by other users. However, read-only view tables (`READ ONLY` specification) cannot be updated by other users regardless of the security facility.

(4) Deleting view tables

You use the `DROP VIEW` definition SQL statement to delete view tables. When a view table is deleted, all related access privileges are also deleted.

12.7 Specifying the FIX attribute

The FIX attribute is an attribute assigned to a table whose row length is fixed.

(1) Effects of specifying the FIX attribute

The effects of specifying the FIX attribute for a table are discussed as follows.

Improved performance

- The performance of retrieving a specific column becomes constant regardless of the order of the column definitions. Additionally, the column retrieval time is reduced, compared to when the FIX attribute is not specified.
- Access performance is improved even when there are many columns because a UAP can use an interface for each row.

Improved operability

If the null value is found in the input data when a column of a table with the FIX attribute is being updated, it can be excluded as an error.

Reduction of required disk space

The physical row length is 2 bytes shorter per column than when the FIX attribute is not specified. If a table contains many columns, disk space is saved.

(2) Criteria

If the null value will not be used in any column and no column is of variable length, the FIX attribute should be specified during table definition.

When these conditions are not satisfied, the following should be evaluated:

- If columns will be added to a table in the future, define a reserved column during table definition. Once you have defined a reserved column, you can add columns to the table even after data has been stored.
- Use the 0 (numeric data) or the space (character data) instead of the null value. Note that the null value is treated differently in search conditions and set functions than other values.
- Change variable-length data with a small maximum value or small range of actual lengths to fixed-length data. Note that variable-length data is handled differently in search conditions.

(3) Specification

To assign the FIX attribute to a table, `FIX` is specified in the `CREATE TABLE` definition SQL (that is, `CREATE FIX TABLE` is specified).

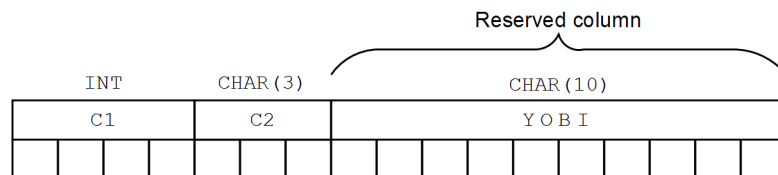
(4) Notes

A reserved column is an area reserved for future use (for adding columns). You cannot perform insert or update processing using desired values on a reserved column. HiRDB stores in the reserved column as many null characters (0x00) as the defined length of the reserved column. Therefore, when you estimate the size of an RDAREA for storing tables, you must include the reserved column in determining the number of table storage pages. The following figure shows how to use a reserved column.

Figure 12–19: How to use a reserved column

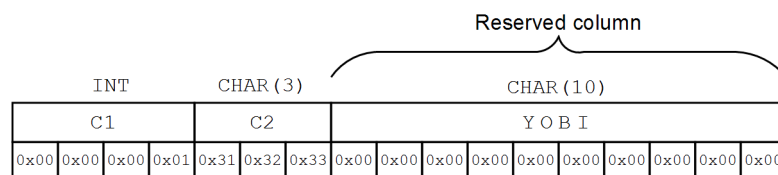
■ Defining a table containing a reserved column

```
CREATE FIX TABLE TABLE01 (C1 INT,C2 CHAR(3),YOBI CHAR(10) FOR RESERVED)
```



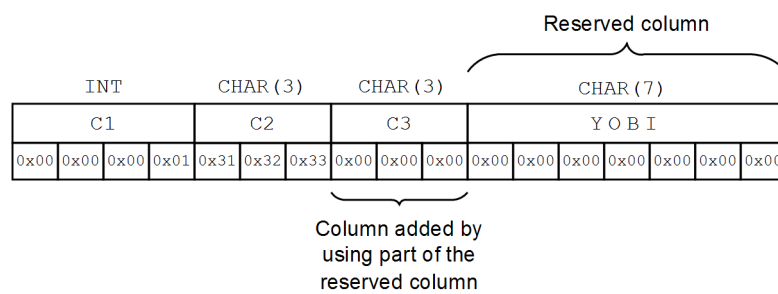
■ Inserting data into a table containing a reserved column

```
INSERT INTO TABLE01 VALUES (1,' 123' )
```



■ Adding a column to a table containing a reserved column

```
ALTER TABLE TABLE01 ADD C3 CHAR(3) INTO YOBI
```



12.8 Specifying a primary key

A primary key is used to identify a unique table row. If you define a primary key, an index is created for the specified column.

(1) Effects of defining a primary key

The uniqueness constraint and `NOT NULL` constraint apply to a column for which a primary key is defined. The uniqueness constraint does not allow a duplication of data in the key (a column or a group of columns). That is, all data in the key is always unique. The `NOT NULL` constraint does not allow a null value in any of the columns in the key.

(2) Criteria

Define a primary key for a column that can identify a unique row. If there is more than one column that can identify a unique row (candidate key), select one of the candidate keys as the primary key. Among the keys in the table, define the one that is most important and that is to be controlled by the uniqueness and `NOT NULL` constraints as the primary key.

(3) Specification

To define a primary key, specify the `PRIMARY KEY` option in the `CREATE TABLE SQL` definition statement.

Note that a primary key can also be added or deleted by using the `ALTER TABLE SQL` definition statement. To add a primary key, specify the `ADD PRIMARY KEY` option. To delete a primary key, specify the `DROP PRIMARY KEY` option.

12.9 Specifying a cluster key

A cluster key is a column that is specified as the key for storing rows in ascending or descending order of the specified column values. If a cluster key is specified for one or more columns in a table, the table rows can be stored in ascending or descending order of the values in the cluster key column(s).

When a cluster key is specified for a table, an index is created automatically for the specified column(s).

(1) Effects of specifying a cluster key

The effects of specifying a cluster key for a table are discussed as follows.

Improved performance

Input/output time can be saved when retrieving, updating, or deleting rows with a range specified or when retrieving or updating rows on the basis of the cluster key values.

Improved operability

- If you define a cluster key with `UNIQUE` specified, the uniqueness and `NOT NULL` constraints apply to the cluster key. In this case, when rows are inserted, no duplicated value is allowed in any row in the cluster key column. Note that you cannot define a cluster key with `UNIQUE` specified for a table partitioned by flexible hash partitioning.
- If you define a cluster key with `PRIMARY` specified, the uniqueness and `NOT NULL` constraints apply to the cluster key. In this case, when rows are inserted, no duplicated value is allowed in any row in the cluster key column. Additionally, no null value can be stored in any of the columns that constitute the cluster key. Note that you cannot define a cluster key with `PRIMARY` specified for a table partitioned by flexible hash partitioning.
- When creating a table, you can use the database load utility (`pdload`) to determine whether the input data is arranged in ascending or descending order of the cluster key values.
- When reorganizing a table, you can use the database reorganization utility (`pdorg`) to determine whether the unloaded cluster key matches the cluster key to be reloaded.

(2) Criteria

The cluster key should be specified in the following cases:

- Many applications accumulate and access data in ascending or descending order of the key values.
- Then table's keys will not be changed.
- The table has fixed-length rows.

(3) Specification

To define a cluster key for a table, specify the `CLUSTER KEY` option in the `CREATE TABLE` definition SQL statement.

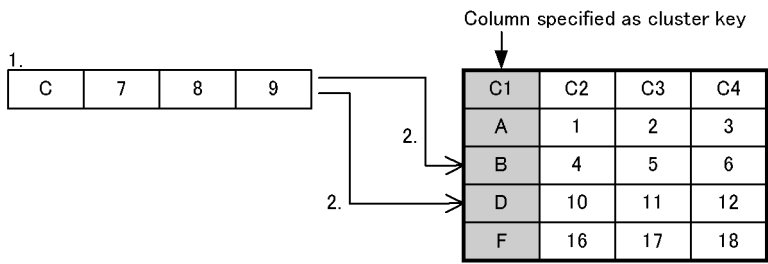
(4) Design considerations

To improve retrieval efficiency after data is added, some unused space should be set in the pages containing the table. For details about how to set space in the pages containing the table, see *14.3 Pages*.

(5) Notes

- The values in a column that constitutes the cluster key cannot be updated.
- The null value cannot be inserted in a column that constitutes the cluster key.
- When data is added to a table with a cluster key specified, there is overhead involved in searching for the page with the key values that are adjacent to the key value being added. The following figure provides an overview of this.

Figure 12–20: Overview of overhead when data is added to a table with a cluster key specified



Explanation:

- 1. Data with *C* in the cluster key column is added.
- 2. There is overhead involved in searching for the key values on either side of the *C* column.

12.10 Specifying the suppress option

The option for omitting part of the data in a table in order to reduce the data length for storage is called the suppress option.

When the suppress option is specified, only the significant digits of the table's decimal data (excluding leading zeros) and the storage data length are stored when the data is stored.

(1) Effects of specifying the suppress option

The effects of specifying the suppress option are discussed as follows.

Improved performance

- The amount of disk space that is required is reduced because the stored data is shorter than the actual data.
- Reducing the required amount of disk space results in a reduction in the input/output time for retrieval processing, such as retrieval of all entries.

(2) Criteria

The suppress option should be specified in the following cases:

- When a table contains much decimal data and there are many significant digits.
- When the table will be accessed by many retrieval applications, such as for retrieval of all entries, but few updating applications.

(3) Specification

To specify the suppress option, enter the `SUPPRESS` option in the `CREATE TABLE` definition SQL statement.

(4) Notes

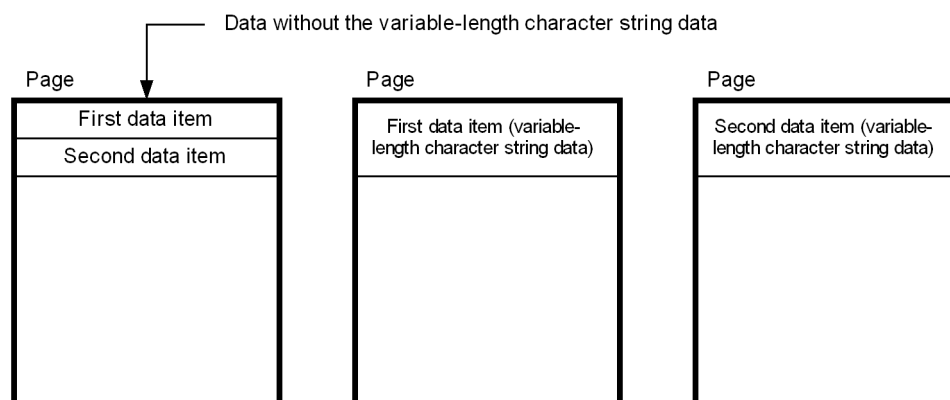
- If the number of significant digits in decimal data equals the defined length or equals 1, the data is stored with a length of *defined length* + 1. In this case, the length of the stored data is greater than when the suppression option is not specified.
- The suppression option cannot be specified for a table with the `FIX` attribute.

12.11 Specifying the no-split option

If any of the data types below is defined for a table and the actual data length of that data type is 256 bytes or greater, the system stores a row of data in multiple pages. The figure following the list shows the data storage method used to do this.

- VARCHAR
- MVARCHAR
- NVARCHAR

Figure 12–21: Data storage method when actual variable-length character string data is 256 bytes or greater



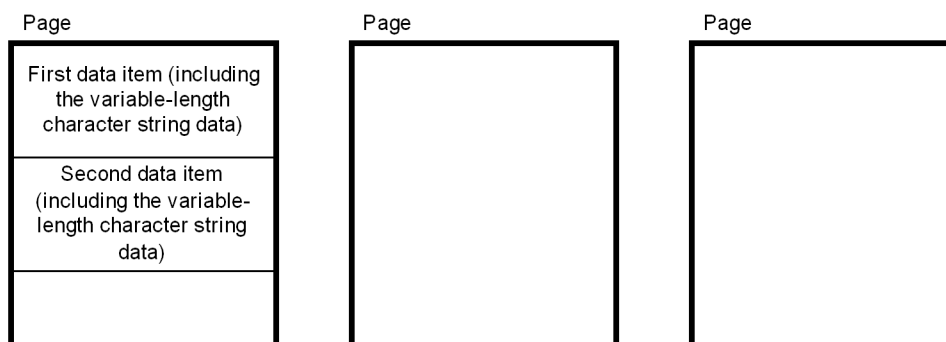
Explanation:

The variable-length character string data is stored in a page separate from the remaining data, adversely affecting the data storage efficiency. In these cases, use the no-split option to improve the data storage efficiency.

(1) Criteria

If you specify the no-split option, the system stores one row of actual variable-length character string data in one page, even if the data length is 256 bytes or greater. The following figure shows the data storage method used when the no-split option is specified.

Figure 12–22: Data storage method used when the no-split option is specified



Explanation:

An entire row of data is stored in the same page. Therefore, the data storage efficiency is better than when the no-split option is not specified.

(2) Specification

To specify the no-split option, specify the `NO SPLIT` option in the `ALTER TABLE`, `CREATE TABLE`, or `CREATE TYPE SQL` definition statement.

(3) Notes

- If the total length of a row of data exceeds the page length, data is split (one row of data is stored in multiple pages) even when the no-split option is specified.
- If you specify the no-split option when the actual variable-length character string data is 255 bytes or less, the column data becomes longer by one byte than when the no-split option is not specified.
- If the no-split option is specified, variable-length character columns will not be split even if the actual data length exceeds 256 bytes. In such a case, a page will be able to store fewer rows than if the no-split option were not specified. For this reason, if a retrieval does not collect column data from variable-length character columns for which an index scan determined the no-split option was appropriate, more pages may be accessed than if the no-split option were specified, and retrieval performance may be deteriorate. However, there is no effect when key scans and table scans are performed.

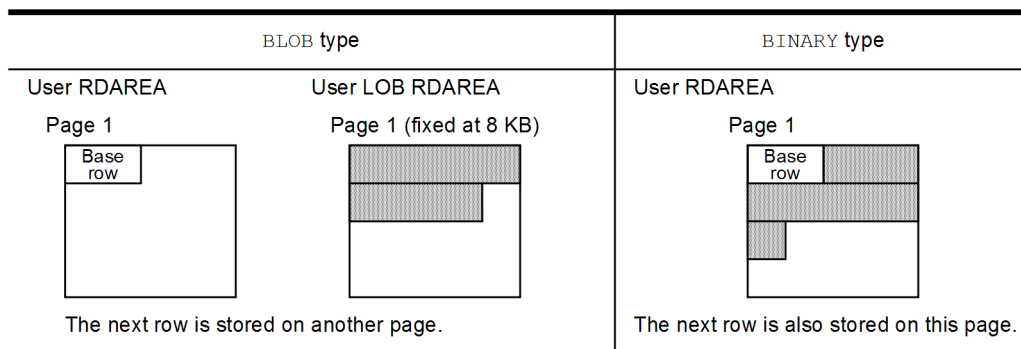
12.12 Specifying a binary data column

There are two data types for defining columns that store documents, images, audio, and other variable-length binary data:

- `BLOB` type (columns specified as the `BLOB` type are called LOB columns)
- `BINARY` type

A comparison of how these types store data follows.

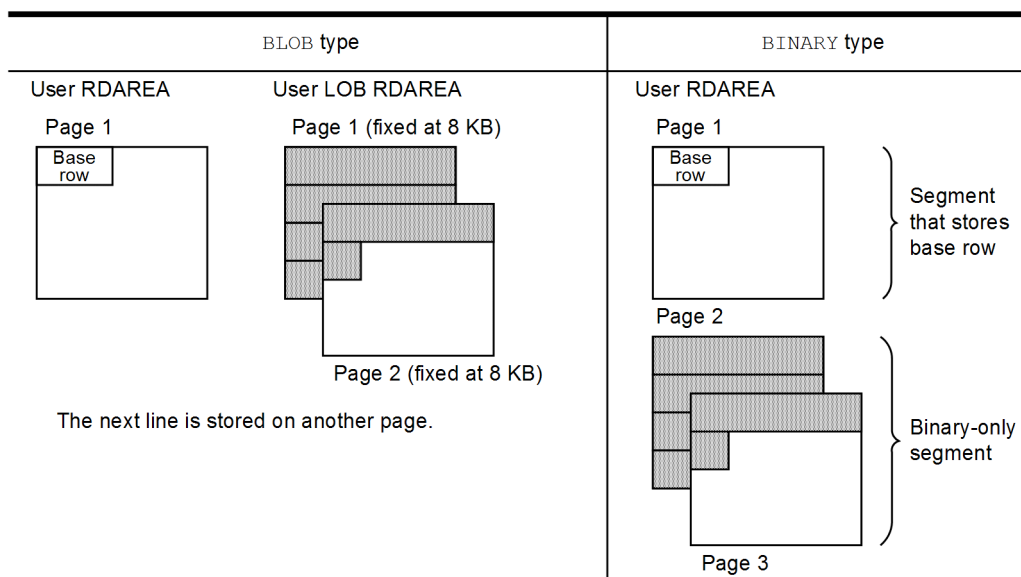
Data length (base row + binary data length) does not exceed 1 page



Legend:

: Binary data

Data length (base row + binary data length) does exceed 1 page



Legend:

: Binary data

Explanation:

A column for which the `BLOB` type is specified differs from a column with other attributes in that it is stored in a user LOB RDAREA whose page length is fixed at 8 kilobytes.

In the case of a column for which the `BINARY` type is specified, if all of the column data that composes the table can be stored on one page, then each row is stored on one page regardless of the actual data length. However, if one row cannot be stored on one page, it is stored on multiple pages in binary-only segments.

12.12.1 BLOB type

(1) Design considerations

- If the BLOB type is to be used, you must create a user LOB RDAREA.
- BLOB data is stored at an 8-kilobyte boundary, giving BINARY-type data better storage efficiency. However, if there is large object data, such that the 8-kilobyte boundary can be ignored, there is not much difference.

(2) Specification

Specify the BLOB type as the data type for the column when you use the CREATE TABLE definition SQL statement.

(3) Notes

The BLOB type cannot be used with the following items:

- Tables with the FIX attribute
- Index definitions
- Partitioning keys

12.12.2 BINARY type

(1) Design considerations

Fewer rows of BINARY-type data can be stored on a page than is the case with BLOB-type data. Because of this, if search conditions are specified on some columns where the BINARY type cannot be retrieved, the BINARY type will require more frequent input/output operations than the BLOB type, thus reducing retrieval performance. However, if indexes are defined for columns that are not the BINARY type and an index scan is performed, the performance difference between the BINARY and BLOB types disappears.

(2) Specification

Specify the BINARY type as the data type for the column when you use the CREATE TABLE definition SQL statement.

(3) Notes

- The BINARY type cannot be used with the following items:
 - Tables with the FIX attribute
 - Index definitions
 - Partitioning keys
 - Columns that make external references
- If the binary data length exceeds one page, the data is stored in binary-only segments, which are different from the segments that store the base rows. When binary data that exceeds one page is stored, RDAREAs might run out of space if there are no unused pages in binary-only segments, even if there are unused pages in segments that store base rows. When an RDAREA does run out of space, you can check whether it was caused by binary data or base row data by using the database analysis utility (pddbstat) to access the usage statuses of binary-only segments and segments that store base rows.
- When you define a BINARY type column, make a precise estimate of the column's defined length (maximum length). If it is unnecessarily long, you might be unable to execute data loads that use memory proportional to the column's defined length (when you execute data loads for tables that define BINARY columns, make sure that there is memory equivalent to the defined length of the BINARY columns).

12.12.3 BLOB type and BINARY type usage

The following table lists recommended data types for each binary data usage.

Table 12–3: Recommended data type for each binary data usage

Binary data usage		Average size of recommended data type		Explanation
		32,000 bytes or less	More than 32,000 bytes	
Frequency of binary data specification in projection columns	High	BINARY	BLOB	<ul style="list-style-type: none"> For 32,000 bytes or less, BINARY type can also use block transfer. A row's entire data is together, yielding better performance. For more than 32,000 bytes, BLOB type can use less memory processing large object data, yielding better performance.
	Low	BLOB [#]	BLOB	BLOB type requires a smaller data size for base rows than BINARY type, so if there are many pieces of data, BLOB will yield better performance. However, if indexes are defined for the non-BINARY columns, index scans will eliminate the difference between BINARY and BLOB types. Index scans are recommended, but the larger the segment size the smaller the performance difference.
SQL descriptor flexibility		BINARY	Equal	For 32,000 bytes or less, if there are no index definitions, BINARY type SQL descriptors can be roughly the same as the VARCHAR descriptor. Therefore, the SQL descriptor range is broader than for the BLOB type. For details, see the manual <i>HiRDB Version 9 SQL Reference</i> .
Data storage efficiency		BINARY		BINARY type has better storage efficiency. However, for large data objects, for which the 8-kilobyte boundary can be ignored, there is little difference.
Frequent additions/updates		BLOB		The larger the concatenation data size, the better the performance of BLOB type.
Frequent partial extractions		Equal	BLOB	If partial extraction is done on a BINARY type with a large stored data size, the performance will be very poor. Further, the greater the frequency of partial extraction, the poorer the performance. If partial extraction will be required frequently against large data, BLOB type is recommended.
Operability is emphasized		BINARY		BLOB type requires special operations, such as backing up the user LOB RDAREA.
If you cannot determine from the above methods or for a possible future policy change		BINARY [#]	BLOB	If data of more than 32,000 bytes is handled, BLOB type is recommended. If the data size is relatively small, BINARY type is recommended.

[#]: If the data size is near to the page size, and if the BINARY type is used with a large number of table scans, performance will be much poorer than for the BLOB type. To avoid this, change from a table scan to an index scan. Index scans are recommended, and even with a large segment size the performance difference is small.

12.13 Specifying a character set

A *character set* is an attribute of character data. A character set has the following three attributes.

- Usage format
These are rules for representing characters. For example, in a given character set, A might be represented by the single-byte code `X'41'`, but in another character set, it might be represented by `X'C1'`. These sorts of rules for character representation are called the *usage format*.
- Character repertoire
This is the set of characters that can be represented. For example, a given character set might allow a backslash to be represented, but another character set might not. The collection of characters that can be represented is called the *character repertoire*.
- Default collating sequence
These are the rules for comparing two character string data items. For example, the collating sequence might be `'1' > 'A'` in a given character set but `'A' > '1'` in another. All character sets have default collating sequences.

The character set that is used when none is specified is called the *default character set*.

(1) Effects of defining a character set

When you define a character set, character string data can be stored in a different character set for each table column. This makes it possible, if EBCDIK is specified as the character set, to retrieve, substitute, and compare character data stored in a database in the collating sequence of VOS3 system character string data when you migrate from a VOS3 system to HiRDB. Specifying UTF-16 as the character set allows retrieval, substitution, and comparison of character data in UTF-16.

(2) Character sets that can be used by HiRDB

The following character sets can be used by HiRDB.

- EBCDIK
To use EBCDIK, specify `sjis` as the character code type when setting up HiRDB.
- UTF16
To use UTF16, specify `utf-8` or `utf-8_ivs` as the character code type when setting up HiRDB.

(3) Specifying a character set

Specify the character set in the character data type. For details about formats and rules that apply when specifying character sets, see the manual *HiRDB Version 9 SQL Reference*.

(4) Notes

- Data cannot be loaded when the input data and the column have different character sets.
- When UTF16 is specified as the character set, the data stored in the database is big endian. When you perform an operation that uses an embedded variable or a `?` parameter on a column that specifies UTF16 for its character set, make the value specified in the embedded variable or `?` parameter big endian as well. If you use little endian, you need to convert the character code, which degrades performance.

12.14 Specifying the WITHOUT ROLLBACK option

When a table is updated (including addition and deletion processing) while the `WITHOUT ROLLBACK` option is in effect, the updated rows are released immediately from locked status, so that the rows become no longer subject to rollback.

(1) Effects of specifying the WITHOUT ROLLBACK option

The effects of specifying the `WITHOUT ROLLBACK` option for a table are discussed as follows.

Improved performance

There are fewer occurrences of locked status because lock control is released upon completion of update processing.

(2) Criteria

This option is suitable for a table that is subject to concentrated update processing, such as when numbering is performed.

A numbering application, such as one that handles form numbers or document numbers, may manage a table by assigning numbers and incrementing the assigned numbers. If processing is concentrated, such an application may be placed frequently in lock-release wait status because the lock status cannot be released until `COMMIT` is issued. If the `WITHOUT ROLLBACK` option is specified for the table in this case, the lock status is released when increment processing is completed, thereby reducing occurrences of lock-release wait status. The following figure shows an example of a numbering application.

Figure 12-23: Example of a numbering application

● Numbering management table

Type	Numbering
Form number	23
Document number	17

← Processing 1 assigns the form number (23)




Assigns the number with the row on locked status, then increments.
Releases the row from locked status after increment processing is completed.

Type	Numbering
Form number	24
Document number	17



Type	Numbering
Form number	24
Document number	17

← Processing 2 assigns the form number (24).

Legend:  : Locked status

This example manages one type of numbers with one row.

Following is an example of defining the numbering management table shown in Figure 12-23:

```
CREATE TABLE numbering-management-table (type NCHAR(4),
                                         numbering INT)
                                         :
                                         WITHOUT ROLLBACK
```

Because a missing number may occur for the following reason, this option should be used only with applications that can handle missing numbers:

- A table defined with the `WITHOUT ROLLBACK` option specified is no longer subject to rollback once its rows are updated. If the UAP or HiRDB is restarted after abnormal termination, a table for an application that uses the assigned numbers can be rolled back correctly, but it is impossible to know how far back the table with the `WITHOUT ROLLBACK` option specified has been rolled back. In this case, assigned numbers may not be used by the application.

(3) Notes

- If the database load utility (`pdload`) or database reorganization utility (`pdrorg`) is executed with the log acquisition mode specified, a table with the `WITHOUT ROLLBACK` option specified is also rolled back in the same manner as with normal tables.
- If update processing is concentrated, such as in the case of a numbering application, a dedicated RDAREA and global buffer should be allocated.
- In the case of a table for which the `WITHOUT ROLLBACK` option is specified, you can create an index only when a row update on the index component column is a same-value update. Because lock is not released during row insertion or deletion, rollback occurs in the same manner as for a normal table.

12.15 Specifying the falsification prevention facility

The falsification prevention facility provides a means for prohibiting all users, including the table owner, from updating table data. This facility protects important data from accidental modification or unauthorized tampering. Tables to which this facility has been applied are called falsification prevented tables. The following table lists operations that can be executed on falsification prevented tables.

Table 12–4: Operations permitted on falsification prevented tables

Operation	Falsification prevented table	
	Deletion prevented duration specified	Deletion prevented duration not specified
Insert (INSERT)	Yes	Yes
Retrieve (SELECT)	Yes	Yes
Update by column (UPDATE)	Yes ^{#1}	Yes ^{#1}
Update by row (UPDATE)	No	No
Delete (DELETE)	Yes ^{#2}	No
Delete all rows (PURGE TABLE)	No	No
Data manipulation SQL other than the above	Yes	Yes

Legend:

Yes: Can be executed.

No: Cannot be executed.

#1: Only updatable columns can be updated.

#2: Only data that has passed the deletion prevented duration can be deleted. If no deletion prevented duration is specified, the table data cannot be deleted.

Applicability standards

The falsification prevention facility is recommended for use with tables when it is important to prevent the table data from accidental modification or unauthorized tampering.

12.15.1 Specification

Specify the CREATE TABLE definition SQL with the INSERT ONLY operand (falsification prevention option) specified. Alternatively, you can change the definition of an existing table to a falsification prevented table by specifying the INSERT ONLY option in ALTER TABLE.

When you are defining a table or changing a table's definition, you can define the following types of columns:

- Updatable column

If you define an updatable column, you can update data for each column as follows:

- Always updatable (UPDATE specified)
- Updatable from null value to a non-null value only once (UPDATE ONLY FROM NULL specified)

You can define updatable columns at the following times:

- When CREATE TABLE is executed
- Before ALTER TABLE (CHANGE INSERT ONLY) is executed
- When ALTER TABLE (ADD *column-name*) or ALTER TABLE (CHANGE *column-name*)[#] is executed

#: `ALTER TABLE (CHANGE column-name)` cannot be executed on falsification prevented tables. If you are changing the definition of an existing table to a falsification prevented table, you must have executed this statement in advance.

- Insert history maintenance column

If you define an insert history maintenance column, you can specify a deletion prevented duration. Because the `DROP TABLE` statement cannot be executed if there is data in such a table (see *12.15.2(1) Definition SQL*), the table and its data are both protected from deletion when the deletion prevented duration is omitted. Therefore, you should specify a deletion prevented duration only if the period over which the data needs to be maintained has been clearly determined or can be determined.

Because there are limitations[#] on the operations that can be performed on RDAREAs by the database reorganization utility or the `pdrels` command, it is recommended that you store each falsification prevented table in a single RDAREA.

#: You must execute a command shutdown on the RDAREA in order to use the database reorganization utility to reorganize a falsification prevented table. If the database reorganization utility terminates abnormally, then if there are any other tables or indexes defined in the RDAREA they will become unavailable, because you cannot release the shutdown until the reorganization has been completed. For details, see *12.15.2 Restrictions*.

12.15.2 Restrictions

Data in a falsification prevented table cannot be updated or deleted. Therefore, there are restrictions on the SQL statements, utilities, and commands that can be executed on a falsification prevented table and any RDAREA in which it is stored.

(1) Definition SQL

Some definition SQL statements cannot be executed on falsification prevented tables. The following table lists restricted definition SQL statements and the restrictions that apply to them.

Table 12–5: Restricted definition SQL statements and the restrictions that apply to them

SQL statement	Restrictions
CREATE TABLE	If all columns have the updatable column attribute, the falsification prevention option cannot be specified.
ALTER TABLE	<ul style="list-style-type: none"> • Table names and column names cannot be changed. • The falsification prevention facility cannot be applied to a table containing data. For details about how to apply the falsification prevention facility to an existing table, see <i>12.15.3 Changing a falsification-unprevented table to a falsification prevented table</i>. • The falsification prevention facility cannot be released. • No existing column can be changed to an updatable column, or no updatable column can be changed to a normal column. • Updatable columns must be defined before the falsification prevention facility is applied.[#] • Setting, releasing, and duration of deletion prevented duration cannot be changed. • If a deletion prevented duration is specified for a falsification prevented table, the insert history maintenance column that specifies the deletion prevented duration cannot be deleted. • Partition storage conditions cannot be changed.
DROP TABLE	Cannot be executed if there is data in a falsification prevented table.

#: To specify an updatable column for an existing table and apply the falsification prevention facility, you must execute `ALTER TABLE` on the column and the table. To apply the falsification prevention facility:

1. Use `ALTER TABLE` to change the attribute of a desired column to updatable.
2. Use `ALTER TABLE (CHANGE INSERT ONLY)` to apply the falsification prevention facility to the table.

(2) Utilities

The operation of utilities is restricted on falsification prevented tables and the RDAREAs that store them. The following table lists restricted utilities and the restrictions that apply to them. There are no restrictions on utilities not listed in the table.

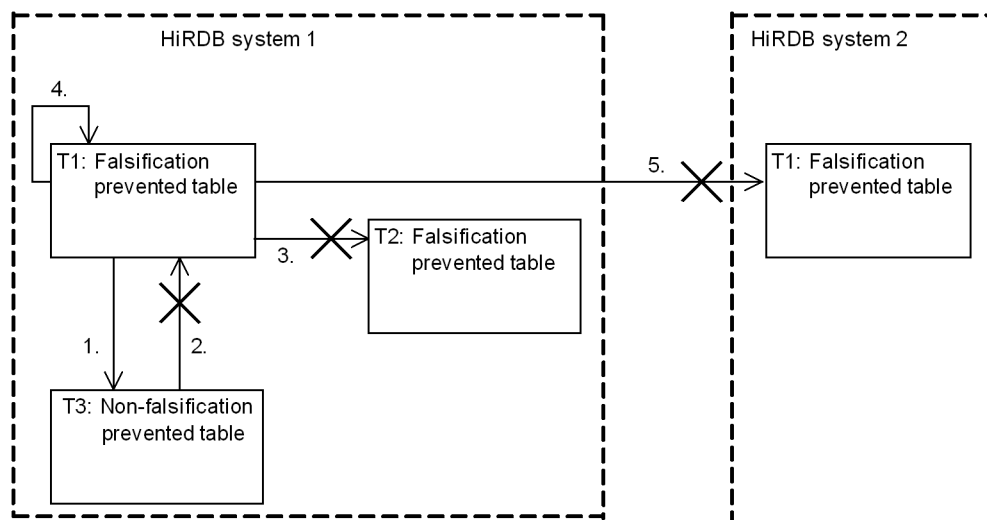
Table 12–6: Restricted utilities and the restrictions that apply to them

Utility	Restrictions
Database creation utility (<code>pdload</code>)	<ul style="list-style-type: none"> Cannot be used in the creation mode (<code>-d</code> option specified). Cannot be executed if the table is in reload-not-completed data status.[#]
Database structure modification utility (<code>pdmod</code>)	<ul style="list-style-type: none"> Cannot re-initialize a falsification prevented table's storage RDAREA (<code>initialize rdarea</code>). The following facilities cannot be used: <ul style="list-style-type: none"> Registering a generation in the HiRDB file system area (<code>create generation</code>) Deleting a generation from the HiRDB file system area (<code>remove generation</code>) Defining an RDAREA replica (<code>replicate rdarea</code>) Copying RDAREA configuration information (<code>define copy rdarea</code>) Integrating RDAREAs (<code>recast rdarea</code>)
Database reorganization utility (<code>pdrorg</code>)	<p>Table reorganization (<code>-k rorg</code>)</p> <ul style="list-style-type: none"> Cannot execute if the related table storage RDAREA is not in command shutdown status. Cannot perform reorganization using a UOC (<code>unlduoc</code> statement). Cannot reorganize the synchronization point specification (<code>option job</code> statement). Cannot execute if related tables are in reload-not-completed data status.[#] <p>Table unload (<code>-k unld</code>)</p> <ul style="list-style-type: none"> Cannot be executed unless the <code>-w</code> option is specified. <p>Table reload (<code>-k reld</code>)</p> <ul style="list-style-type: none"> Cannot execute if the related table storage RDAREA is not in command shutdown status. Can execute if the related table is in reload-not-completed data status[#] (can only re-execute if table reloading terminates abnormally during table reorganization) Cannot reorganize the synchronization point specification (<code>option job</code> statement). Cannot reload to another table (for details, see Figure 12-24). <p>Batch index creation (<code>-k ixmk</code>), re-creation (<code>-k ixrc</code>), reorganization (<code>-k ixor</code>)</p> <ul style="list-style-type: none"> Cannot execute if related tables are in reload-not-completed data status.[#]
Rebalance utility (<code>pdrbal</code>)	Cannot execute if related tables are in reload-not-completed data status. [#]

[#]: If reorganization is executed for a falsification prevented table, but because of an error or some other reason the reload is not completed, the table is placed in a status called reload-not-completed data status, which status is also applied to the storage RDAREAs of the falsification prevented table. You can check whether an RDAREA is in reload-not-completed data status with the database condition analysis utility, the RDAREA unit analysis utility (logical analysis), or by means of a table unit status analysis. Reload-not-completed data status can be released when

table reorganization (table reloading) completes normally. For details about the reload-not-completed data status, see the manual *HiRDB Version 9 Command Reference*.

Figure 12–24: Reloading to another table



Explanation:

1. Falsification prevented table T1 can be reloaded to non-falsification prevented table T3, because falsification prevented table T1 loses its restrictions.
2. Non-falsification prevented table T3 cannot be reloaded to falsification prevented table T1, because the data in falsification prevented table T1 is protected.
3. Falsification prevented table T1 cannot be reloaded to falsification prevented table T2, because the data in falsification prevented table T2 is protected.
4. Falsification prevented table T1 can be reloaded back into itself, because falsification prevented table T1 loses its protection.
5. Falsification prevented table T1 on HiRDB system 1 cannot be reloaded into falsification prevented table T1 on HiRDB system 2, because the data in falsification prevented table T1 on HiRDB system 1 loses its protection when it is in falsification prevented table T1 on HiRDB system 2.

(3) Operation commands

There are restrictions on the commands that can be used on falsification prevented tables and the RDAREAs in which falsification prevented tables are stored. The following table lists the restricted operating commands.

Table 12–7: Restricted commands and their restrictions

Operation command	Restrictions
RDAREA shutdown (pdhold)	<p>If you cannot execute a reload in order to complete the reorganization of a falsification prevented table so that the status of the RDAREA that stores the table can be changed from reload-not-completed data status, the following options cannot be executed:</p> <ul style="list-style-type: none"> • Reference-possible shutdown: <code>-i</code> • Backup shutdown: <code>-b</code>
RDAREA shutdown release (pdrels)	<p>If you do not reload an RDAREA that stores a falsification prevented table that is in reload-not-completed data status, you cannot release the shutdown before the table reorganization is completed, because that would leave 0 records of data.</p>

(4) Restrictions on related products

Restrictions on related products include the following:

- Inner replica facility

The inner replica facility cannot be used with RDAREAs that store falsification prevented tables. The falsification prevention facility cannot be applied to a table that is stored in an RDAREA using the inner replica facility.

- Replication facility

For falsification prevented tables, do not use the replication facilities (HiRDB Dataextractor and HiRDB Datareplicator) to copy data and reflect the result. An attempt to do so could result in a mismatch in the data in the reflection source and the reflection result, causing an error.

12.15.3 Changing a falsification-unprevented table to a falsification prevented table

This subsection describes how to change an existing table to a falsification prevented table. Either of the following methods can be used to change a table to a falsification prevented table:

- Using HiRDB Control Manager's falsification prevention wizard
- Using HiRDB commands

Tables in which data is stored cannot be changed into falsification prevented tables. When data is stored in the table, first unload the table data, and then change the table definition using `ALTER TABLE`.

(1) Using HiRDB Control Manager's falsification prevention wizard

The following describes the procedure for changing a table to a falsification prevented table by using HiRDB Control Manager's falsification prevention wizard.

Procedure

This example changes table `T1` to a falsification prevented table with the following conditions set:

- Column `COL_NOTE` is an updatable column.
- Column `COL_DATE` is an insert history maintenance column, and the deletion prevented duration is set to 10 years.

1. Start HiRDB Control Manager - Console. For details about how to start HiRDB Control Manager - Console, see the *HiRDB Version 9 System Operation Guide*.

2. Register the applicable HiRDB server. For details about how to register the management HiRDB system, see the *HiRDB Version 9 System Operation Guide*. If it is already registered, this step is not necessary.

3. From the **Table Operations** tab menu, choose **Falsification Prevented Table Migration Wizard**.

The Falsification Prevention Wizard - 1/5 window appears.

Select the name of the table to be changed to a falsification prevented table, and then in the **Password** text box, enter the password of the table owner. The value entered here is case-sensitive.

4. Click the **Next** button.

The Falsification Prevention Wizard - 2/5 window appears.

In this window, set the deletion prevented duration. If you do not set the deletion prevented duration, make sure that **Do not delete table data** is selected. If you set a deletion prevented duration, select **Allow deletion of data in table after a certain period of time**, and then set **Insert history maintenance column name** and **Deletion prevented duration**.

5. Click the **Next** button.

The Falsification Prevention Wizard - 3/5 window appears.

In this window, set the updatable columns. If no updatable columns are set, make sure that **Do not allow per-column data updates** is selected. If you set updatable columns, select **Allow updates of some columns**.

Next, from the **Columns** list, select the names of the columns to make updatable. Then, click the right arrow button to add them to the **Updatable columns** list.

6. Click the **Next** button.

The Falsification Prevention Wizard - 4/5 window appears.

In this window, specify the directory for the temporary files that are used while the table is being changed. The default values appear under **Temporary file storage locations**. Make changes if necessary.

7. Click the **Next** button.

The Falsification Prevention Wizard - 5/5 window appears.

Check the conditions that were set. If you want to change the settings, click the **Back** button to return to the previous screens.

8. When you are satisfied, click the **Execute** button. Processing for changing the table begins.

Important note

After the falsification prevention wizard has been executed, make backups of the dictionary RDAREA and the RDAREAs required for the recovery of the table data. You can make backups by executing HiRDB Control Manager's backup wizard.

Reference note

In HiRDB Control Manager, a falsification prevented table has a different appearance from a normal table. You can verify that the changed table has the falsification prevention attribute by selecting **Tables** from the **Map** tab menu to display the Table Map window.

(2) Using HiRDB commands

The following procedure shows how to change a table into a falsification prevented table using HiRDB commands.

Procedure

Change table T1 stored in an RDAREA (RDAREA01) into a falsification prevented table.

1. Using the `pdhold` command, place a backup hold on the RDAREA that stores the falsification-unprevented table and the RDAREA for the data dictionary (RDDIC01).

```
pdhold -r RDAREA01,RDDIC01 -b
```

2. Swap the system log files of the servers to which the RDAREA to be backed up belongs (`bes01` and `dic01`).

```
pdlogswap -d sys -s bes01 -w
pdlogswap -d sys -s dic01 -w
```

3. Execute the database copy utility (`pdcopy`) to make a backup of the RDAREA.

For details about how to make backups, see the *HiRDB Version 9 System Operation Guide*.

```
pdcopy -m /hirdb/rdarea/mast/mast01 -M r -p /usr/hirdb/pdcopy/pdcopy01 -b /usr/
hirdb/pdcopy/backup/backup01 -r RDAREA01,RDDIC01
```

4. Use the `pdrels` command to release the hold on the data dictionary RDAREA.

```
pdrels -r RDDIC01
```

5. Use the database reorganization utility (`pdorg`) to unload data from the falsification-unprevented table. Make sure that you specify the `-W` option so that the unloaded data can be used as input data for the database load utility (`pdload`). For details about the control statement file (`control_file`), see the manual *HiRDB Version 9 Command Reference*.

```
pdorg -k unld -t T1 -W bin control_file
```

6. Use the `pdrels` command to release the hold on the user RDAREA. Do not access RDAREAs after this until you again place a hold on the RDAREA in step 9. If the table you are working with is updated during this period, data might lose integrity.

```
pdrels -r RDAREA01
```

7. Use `PURGE TABLE` to delete all data from the falsification-unprevented table.

```
PURGE TABLE T1
```

8. Use `ALTER TABLE` with the falsification prevention option specified to change the table to a falsification prevented table.

```
ALTER TABLE T1 CHANGE INSERT ONLY
```

9. Using the `pdhold` command, place a hold on the RDAREA that stores the falsification-prevented table.

```
pdhold -r RDAREA01
```

10. Use the database load utility (`pdload`) to load the data that was unloaded in step 5. For details about the control statement file (`control_file`), see the manual *HiRDB Version 9 Command Reference*.

```
pdload -b -W T1 control_file
```

11. Use the `pdrels` command to release the hold on the RDAREA.

```
pdrels -r RDAREA01
```

12. Using the `pdhold` command, place a backup hold on the RDAREA to be backed up.

```
pdhold -r RDAREA01,RDDIC01 -b
```

13. Swap the system log files of the servers to which the RDAREA to be backed up belongs (`bes01` and `dic01`).

```
pdlogswap -d sys -s bes01 -w
pdlogswap -d sys -s dic01 -w
```

14. Execute the database copy utility (`pdcopy`) to make a backup of the RDAREA.

For details about how to make backups, see the *HiRDB Version 9 System Operation Guide*.

```
pdcopy -m /hirdb/rdarea/mast/mast01 -M r -p /usr/hirdb/pdcopy/pdcopy01 -b /usr/
hirdb/pdcopy/backup/backup01 -r RDAREA01,RDDIC01
```

15. Use the `pdrels` command to release the hold on the RDAREA.

```
pdrels -r RDAREA01,RDDIC01
```

To determine the timing of setting the falsification prevention option, check the values in the `SQL_TABLES` data dictionary table. The following table lists the meanings of values in the `SQL_TABLES` table.

Table 12–8: Meanings of values in the `SQL_TABLES` table

Falsification prevention option setting	SQL_TABLES	
	Value of <code>INSERT_ONLY</code> column	Value of <code>CHANGE_TIME_INSERT_ONLY</code>
Not specified	Null	Null
Specified during execution of <code>CREATE TABLE</code>	Y	Null
Specified during execution of <code>ALTER TABLE</code>	Y	Date and time the table was changed to falsification prevented table

12.15.4 Error operation

Because RDAREAs that store falsification prevented tables cannot be reinitialized (`initialize rdarea`), these RDAREAs cannot be recovered using re-initialization recovery. Recovery must be with the database recovery utility (`pdrstr`). If the RDAREA is full, expand it with the `expand rdarea` statement.

12.16 Table containing a repetition column

HiRDB permits definition of a table that contains a column in which multiple elements can be stored in each row. In other words, a table can be defined with repetition columns.

Elements are the items that are repeated in the rows of a repetition column. To define such a table, it must be created conventionally as shown in the following figure. The following figure shows an example of two tables defined without repetition columns.

Figure 12–25: Example of tables defined without repetition columns

STAFF_TABLE			FAMILY_TABLE			
NAME	SEX	QUALIFICATION	NAME	FAMILY	RELATIONSHIP	SUPPORT
JIM JONES	MALE	DATA PROCESSING CLASS 1	JIM JONES	RICHARD	FATHER	1
JIM JONES	MALE	NETWORK	JIM JONES	MARY	MOTHER	1
JIM JONES	MALE	DATA PROCESSING CLASS 2	JIM JONES	CATHY	WIFE	1
MARK TYLER	MALE	DATA PROCESSING CLASS 2	JIM JONES	TERRY	ELDEST SON	1
MARK TYLER	MALE	ENGLISH LEVEL 2	JIM JONES	CHRISTIE	SECOND DAUGHTER	1
JOHN WHITE	MALE	SYSTEM ADMINISTRATION	MARK TYLER	ALLEN	FATHER	0
TOM JOHNSON	MALE		MARK TYLER	LISA	WIFE	1
			JOHN WHITE	NANCY	MOTHER	1

To access these two tables, they must first be joined. Joining tables results in disadvantages, such as complicating the SQL syntax. If repetition columns are used, one table containing all the information in two tables can be created without having to join them.

The following figure shows an example of a table containing repetition columns.

Figure 12–26: Example of table containing repetition columns

STAFF_TABLE						
NAME	QUALIFICATION		SEX	FAMILY	RELATIONSHIP	SUPPORT
JIM JONES	DATA PROCESSING CLASS 1		MALE	RICHARD	FATHER	1
	NETWORK			MARY	MOTHER	1
	DATA PROCESSING CLASS 2			CATHY	WIFE	1
				TERRY	ELDEST SON	1
				CHRISTIE	SECOND DAUGHTER	1
MARK TYLER	DATA PROCESSING CLASS 2		MALE	ALLEN	FATHER	0
	ENGLISH LEVEL 2			LISA	WIFE	1
JOHN WHITE	SYSTEM ADMINISTRATION		MALE	NANCY	MOTHER	1
TOM JOHNSON			MALE			

Elements of a repetition column

Row

Note: Blank cells contain the null value.

Explanation:

QUALIFICATION, FAMILY, RELATIONSHIP, and SUPPORT are repetition columns.

(1) Effects of defining repetition columns

A table with multiple values and multiplicity can be expressed in rows. Therefore, the following effects can be expected:

- There is no need to join multiple tables.

- Disk space can be saved because no information is duplicated.
- Because related data items (repeated data) are stored adjacent to each other, higher access performance can be achieved than when separate tables are used.

(2) Specification

To specify a repetition column, specify the `ARRAY` option in the `CREATE TABLE` definition SQL statement.

An example of defining a table containing repetition columns is shown as follows. This definition is based on the `STAFF_TABLE` shown in Figure 12-26. This example assumes that a multicolumn index has been defined for `RELATIONSHIP` and `SUPPORT`.

Example

```
CREATE TABLE STAFF_TABLE
(NAME NVARCHAR(10),
QUALIFICATION NVARCHAR(20) ARRAY[10],
SEX NCHAR(1),
FAMILY NVARCHAR(5) ARRAY[10],
RELATIONSHIP NVARCHAR(5) ARRAY[10],
SUPPORT SMALLINT ARRAY[10]);

CREATE INDEX SUPPORTIDX ON STAFF_TABLE (RELATIONSHIP, SUPPORT);
```

Note

`SUPPORTIDX` is an index name assigned to `STAFF_TABLE`.

(3) Notes

- A repetition column cannot be specified for the following data types:
 - `BLOB` type
 - `BINARY` type
 - Abstract data type
- A repetition column cannot be specified for a column for which a cluster key is specified.
- If `FIX` is specified for a table, repetition columns cannot be specified.
- Neither storage conditions, hash partitioning, nor the suppress option can be specified for a repetition column.
- If key range partitioning is used, a repetition column cannot be specified as the column for which boundary values are specified.
- The `NOT NULL` constraint cannot be specified for a repetition column.

12.17 Table containing an abstract data type

An abstract data type can be defined as the data type of a column in a table. Tables containing abstract data types can be created.

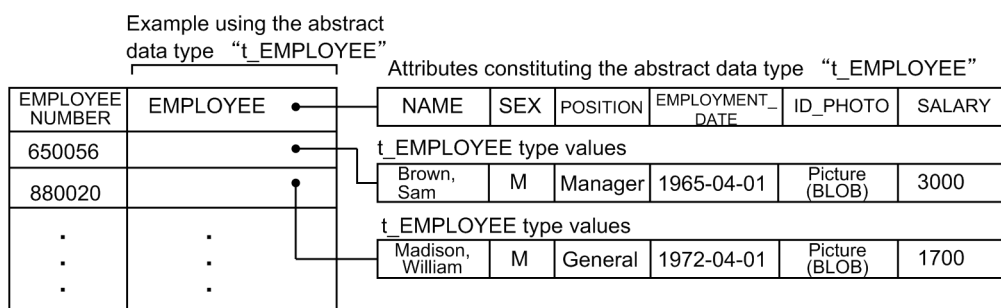
An abstract data type provides a structure that enables complicated data that cannot be handled at all by existing data types to be handled easily. HiRDB allows users to define such a data type as an abstract data type. Creating an abstract data type involves using the definition SQL to define attributes indicating the structure and to define the operations to be performed on the values.

An abstract data type can be treated as a data type of a table, in the same manner as any other data type provided by the HiRDB system, such as the numeric and character types.

The following figure shows the data structure of a table that contains an abstract data type. In this figure, the EMPLOYEE column of the STAFF_TABLE is set to abstract data type t_EMPLOYEE.

Figure 12–27: Data structure of a table containing an abstract data type

- Staff table, including the abstract data type “t_EMPLOYEE”



(1) Effects of defining an abstract data type

- Data with a complicated structure can be treated as a single value.
- Mapping with an object-oriented application is simplified by combining data and its manipulation procedure.
- By combining data and its manipulation procedure and using the manipulation procedure as an external interface, data can be handled without having to know the internal information about the data.

(2) Overview of inheritance

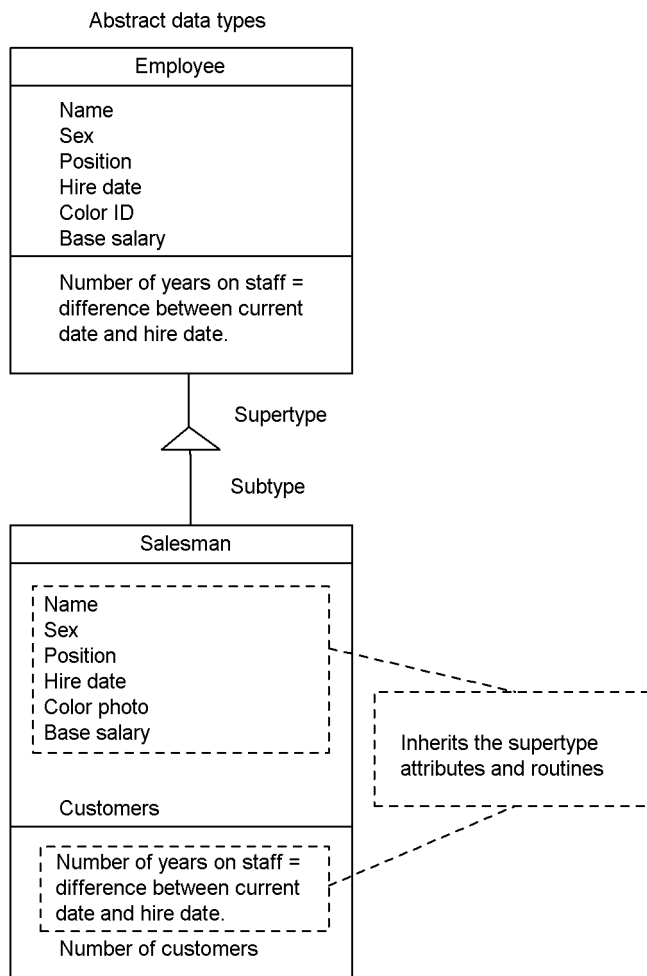
(a) Inheritance

A new abstract data type can be derived from an existing abstract data type by inheriting attributes and the manipulation procedure. When this is done, the base type is called the supertype and the derived type is called a subtype. Transferring a supertype's attributes and function to a subtype is called inheritance.

The relationship between a supertype and a subtype can be expressed as a hierarchy. Therefore, a complicated concept model can also be expressed as a hierarchy using an abstract data type.

The following figure shows a hierarchical structure based on the relationship between supertype and subtype abstract data types. In this figure, the subtype OPERATOR is derived from the abstract data type EMPLOYEE.

Figure 12–28: Hierarchical structure based on the relationship between supertype and subtype abstract data types

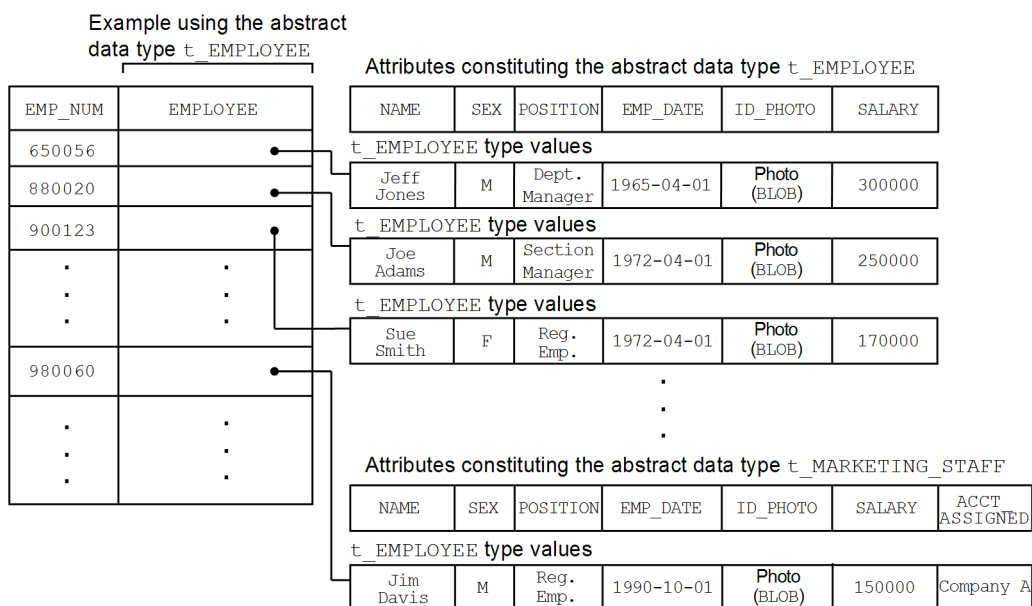


(b) Substitutability

The values of a subtype can be treated as values of its supertype. This is called substitutability. The following figure shows the data structure of a table that contains an abstract data type with a value substituted by making use of substitutability.

Figure 12–29: Data structure of a table containing an abstract data type (using substitutability)

- Example using the abstract data type `t_EMPLOYEE`



(c) Override

A routine defined as a high-order abstract data type (supertype) can be overwritten with a low-order abstract data type (subtype) that has the same name. Defining a routine by overwriting in such a manner is called override. When override is used, the name of a routine called need not be changed depending on its type.

(3) Effects of using inheritance

When inheritance is used, the following effects can be expected:

- The characteristics of the high-order abstract data type (data and manipulation procedure) can be used by a low-order abstract data type.
- The subtype enables a data definition to be shared without having to define from the beginning. This simplifies database definition.
- When override is used, the name of a routine called need not be changed depending on its type.

(4) Defining an abstract data type

The `CREATE TYPE` definition SQL is used to define an abstract data type. `CREATE TYPE` defines the attributes indicating the structure of the abstract data type and defines the procedure for manipulating the values. If inheritance is used, the subtype clause of `CREATE TYPE` is specified. For an example of a `CREATE TYPE` definition, see 6.5 *Creating a table containing a user-defined abstract data type*.

(a) Defining a constructor function

A constructor function to be used to generate values for an abstract data type can be defined. The HiRDB system provides a default constructor function that can be used when an abstract data type is defined. The default constructor function generates values whose attributes are all the null value.

(b) Defining a routine

A routine can be defined in an abstract data type definition as an interface for manipulating the values of an attribute.

(c) Specifying an encapsulation level

An encapsulation level can be specified to control accesses to the attributes comprising an abstract data type and a routine. An encapsulation level can be specified for a routine that is used to manipulate attributes and the abstract data type's values. There are three encapsulation levels:

- **PUBLIC**

This encapsulation level is used in the definition of an abstract data type other than the applicable abstract data type or its subtypes or to allow accesses to attribute values from an application or to allow a routine to be used.

- **PRIVATE**

To prevent internal information from being modified directly by an application, this encapsulation level is used to allow accesses to attribute values only in the definition of the applicable abstract data type or to allow a routine to be used. To use an SQL to access an attribute value or to use a routine, functions must be defined.

- **PROTECTED**

To protect information from being referenced directly by an application for security purposes, this encapsulation level is used to allow accesses to attribute values only within the definition of the applicable abstract data type and its subtypes or to allow a routine to be used.

Once an encapsulation level is specified within the definition of an abstract data type, the encapsulation level remains in effect until another encapsulation level is specified. If no encapsulation level is specified, **PUBLIC** is assumed. The range of data access and routine usage privilege depends on the encapsulation level. The following table lists encapsulation levels and privileges.

Table 12–9: Encapsulation levels and privileges

Encapsulation level	Access source			
	Within the definition of the abstract data type	Within the definition of its subtype abstract data types	Within the definition of another abstract data type than those on the left	Application
PUBLIC	P	P	P	P
PRIVATE	P	--	--	--
PROTECTED	P	P	--	--

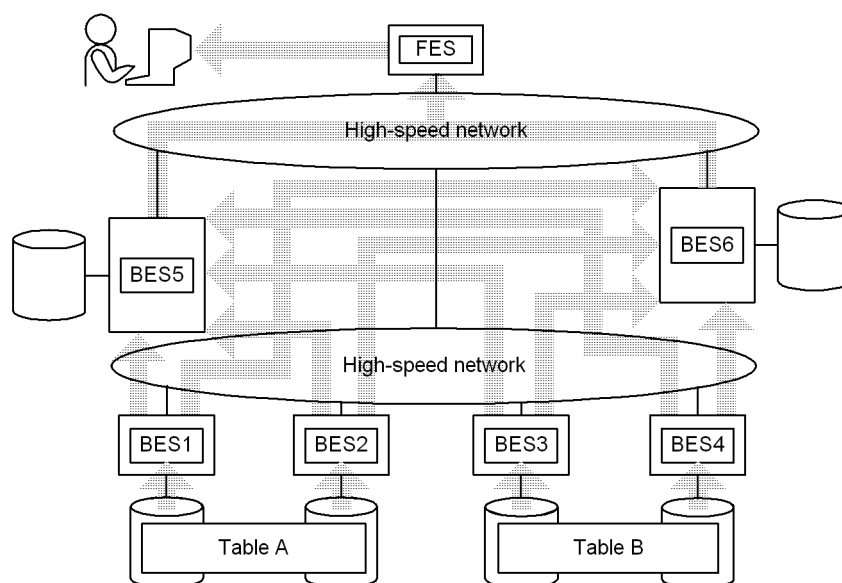
P: Accesses to attribute values and use of routines are permitted.

--: Accesses to attribute values and use of routines are not permitted (if violated, an SQL error results).

12.18 Shared tables

In the case of a HiRDB parallel server configuration, when multiple tables are joined, table data is read from the back-end servers where individual tables are located and then matching is performed at a separate back-end server. This means that multiple servers are connected to transfer data. If the range of data to be searched for matches is located on a single back-end server, matching can be completed at a single back-end server by creating that data as a shared table. A *shared table* is a table stored in a shared RDAREA that can be referenced by all back-end servers. An index defined for a shared table is called a *shared index*. Only an *updatable back-end server* can update shared tables. Other back-end servers are referred to as *reference-only back-end servers*. Because there are limitations on updating a shared table, it is important that you do not update shared tables during online operations. For details about updating shared tables, see 12.18.3 *Manipulating shared tables*. Figure 12-30 shows join processing without using a shared table, and Figure 12-31 shows join processing using a shared table.

Figure 12-30: Join processing without using a shared table



Explanation:

This example joins tables A and B.

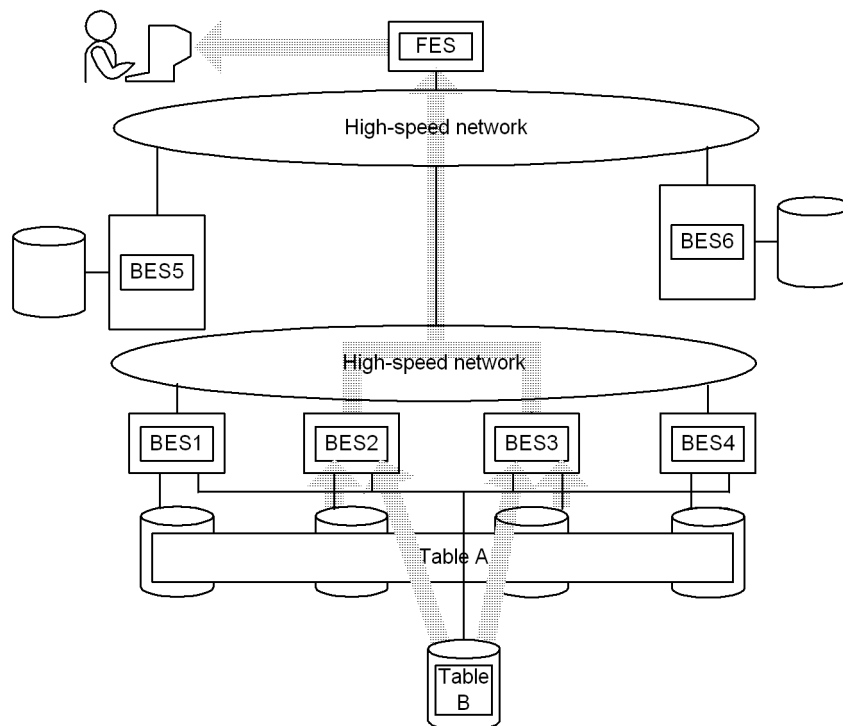
BES1, BES2: Retrieve data from table A and transfer it to BES5 and BES6 for matching.

BES3, BES4: Retrieve data from table B and transfer it to BES5 and BES6 for matching.

BES5, BES6: Perform matching and join processing and then transfer data to the FES.

FES: Merges the joined data and sends the result to the user.

Figure 12–31: Join processing using a shared table

**Explanation:**

This example joins tables A and B. Table B is a shared table that contains shared data. The search ranges are located in back-end servers BES2 and BES3.

BES1, BES4, BES5, BES6: No processing.

BES2, BES3: Retrieve data from tables A and B, perform merge processing, and then transfer the data to the FES.

FES: Sends the results to the user.

Shared tables and shared indexes can also be defined for a HiRDB single server configuration. This provides SQL and UAP compatibility with a HiRDB parallel server configuration. Shared tables and shared indexes are usually used with a HiRDB parallel server configuration because they are especially effective in HiRDB parallel server configurations. The following subsections describe the use of shared tables with a HiRDB parallel server configuration. For details about using shared tables with a HiRDB single server configuration, see *12.18.7 Using shared tables with a HiRDB single server configuration*.

12.18.1 Effects and criteria

(1) Effects of shared tables

Because join processing can be completed by a single back-end server, the overhead associated with connecting between back-end servers and transferring data is reduced. Additionally, the number of back-end servers required for each transaction can be reduced, thereby improving the efficiency of parallel processing, particularly in the event of multiple executions.

(2) Criteria

We recommend that you create as a shared table a table that typically involves minor update processing but which is referenced by multiple transactions, such as for join processing.

12.18.2 Definition method

Specify `SHARE` in the `CREATE TABLE` definition SQL statement (specify as `CREATE SHARE FIX TABLE`). Note that the shared table must satisfy the following conditions:

- The shared table is an unpartitioned `FIX` table.
- The `RDAREA` for storing the shared table and shared index is a shared `RDAREA` (`SDB` is specified in the `-k` option of the `pdfmkfs` command).
- The `WITHOUT ROLLBACK` option is not specified.
- It is not a referencing table for which a referential constraint has been defined.

12.18.3 Manipulating shared tables

(1) Searching

Because a shared table can be referenced by all back-end servers, HiRDB selects the back-end server that is most suitable for searching the shared table. When a shared table is updated, deadlock may occur between the search and update processing because all back-end servers apply lock. To avoid deadlock, we recommend that you search a shared table as follows:

- Specify `WITHOUT LOCK` or `WITHOUT LOCK NOWAIT` as the lock option.
- When you search a shared table for updating purposes, specify the `FOR UPDATE` clause.

If a `LOCK` statement with `IN EXCLUSIVE MODE` specified is executed on a shared table, the `RDAREA` containing the target shared table and shared index is locked. If the same `RDAREA` is accessed, this lock occurs even if the table to be searched is not the target of the `LOCK` statement. Therefore, if another transaction is executing a `LOCK` statement with `IN EXCLUSIVE MODE` specified, the shared table cannot be accessed even when `WITHOUT LOCK NOWAIT` is specified. This means that the shared table cannot be searched while the `LOCK` statement with `IN EXCLUSIVE MODE` specified is executing.

For details about the rules by which HiRDB allocates back-end servers, see *12.18.5 Rules used to allocate back-end servers that search shared tables*.

(2) Updating

To update a shared table, you must specify `IN EXCLUSIVE MODE` in the `LOCK` statement to lock the shared `RDAREAs` of all back-end servers. In the case of an `UPDATE` statement that does not change index key values, there is no need to issue the `LOCK` statement. An update to the shared table and shared index is written to the disk when the `COMMIT` statement is issued.

If you are using a local buffer to update a shared table, make sure that you issue the `LOCK` statement. If the shared table is updated without issuance of the `LOCK` statement and the server process terminates abnormally, the abort code `Phb3008` is output (the unit may terminate abnormally).

(a) Updating involving `LOCK` statement issuance

To update a shared table with issuance of a `LOCK` statement:

1. Issue the `LOCK` statement with `IN EXCLUSIVE MODE` specified.

The `LOCK` statement locks not only the specified shared table but also the shared `RDAREAs` that contain the shared table and shared index. The global buffer for the shared `RDAREA` is disabled at the reference-only back-end server.

2. Execute the `INSERT`, `UPDATE`, or `DELETE` statement for the shared table.

The updatable back-end server applies the update information to the file.

Because the shared `RDAREA` is locked until the `LOCK` statement is released, all accesses to other shared tables in the same shared `RDAREA` are placed in wait status.

3. Release the `LOCK` statement.

Notes

- Issue the `LOCK` statement at the beginning of the UAP. If any local server process has an open cursor to a table in the related shared RDAREAs, the `LOCK` statement results in an error.
- When you create a procedure and trigger to update a shared table, specify the `LOCK` statement. If you execute the `LOCK` statement from a procedure and trigger, locking does not take place at the point where the transaction starts. This may result in an error.
- The shared table, the shared RDAREA containing the shared table, and the shared RDAREA containing the shared index are locked at all back-end servers. If any application accesses a table or index in the corresponding RDAREA, deadlock or server-to-server global deadlock may occur.
- If the unit for an updatable back-end server terminates abnormally and does not restart before a shared table updating transaction is completed, and the following search is executed, a lock timeout error occurs (KFPA11770-I message is displayed):
 - A reference-only back-end server on another unit searches a table in the RDAREA that contains the shared table being updated or an index defined for that table.

(b) Updating without LOCK statement issuance

If you do not issue a `LOCK` statement, you can execute only an `UPDATE` statement that does not change index key values. Use this method only for minor changes.

To update a shared table without issuing a `LOCK` statement:

1. To place all back-end servers in the same status, distribute the update information to all back-end servers.
2. The updatable back-end server applies the update information to the database.

The reference-only back-end server updates information in the global buffer and retains the update information without applying it to the file until the `COMMIT` statement is issued. If the transaction rolls back, the data is restored in the global buffer.

Notes

- At a reference-only back-end server, if all global buffers are under update processing and there is no available page before the `COMMIT` statement is issued, the transaction rolls back. Therefore, when you are not issuing a `LOCK` statement, do not update a large amount of data.
- The rows to be updated are locked at all back-end servers. If any application accesses the corresponding table at the same time, deadlock or server-to-server global deadlock may occur. To avoid deadlock, we recommend that you use the `UPDATE` statement to update only one row per transaction.
- If the unit for an updatable back-end server terminates abnormally and does not restart before a shared table updating transaction is completed, and the following search is executed, a lock timeout error occurs (KFPA11770-I message is displayed):
 - A reference-only back-end server on another unit searches a table in the RDAREA that contains the shared table being updated or an index defined for that table.

12.18.4 Limitations on shared tables

- A shared table cannot be searched while a `LOCK` statement with `IN EXCLUSIVE MODE` specified is executing.
- The `ASSIGN LIST` statement cannot create a list for shared tables.
- A shared table cannot be specified as a replication target.

12.18.5 Rules used to allocate back-end servers that search shared tables

This subsection explains the rules that are used for allocating back-end servers that search shared tables. The allocation rules vary between cases in which only shared tables are specified in a single SQL statement, and cases in which multiple tables that contain shared tables are specified in a single SQL statement. The following subsections explain allocation rules separately for each case.

(1) When all tables specified in a single SQL statement are shared tables

When all tables specified in a single SQL statement are shared tables, the allocation of back-end servers that search shared tables is determined by conditions such as what kind of search is performed by the immediately preceding SQL statement in the same transaction.

The following table lists the rules used for allocating back-end servers that search shared tables (when all tables specified in a single SQL statement are shared tables). Item 1 has the highest priority.

Table 12–10: Rules used for allocating back-end servers that search shared tables (when all tables specified in a single SQL statement are shared tables)

Item	Search conditions	Back-end server allocated
1	One of the following conditions is met: <ul style="list-style-type: none"> A <code>LOCK TABLE</code> statement that specifies <code>IN EXCLUSIVE MODE</code> is executed on the shared table to be searched. A search is performed using a <code>FOR UPDATE</code> clause. 	HiRDB allocates an updatable back-end server.
2	Shared tables ^{#2} are used in different SQL statements in the same transaction (or across multiple transactions ^{#1}).	From among the shared tables that were searched in the same transaction, HiRDB allocates the back-end server used by the SQL statement that most recently accessed a shared table. For an example of how the back-end servers are allocated in this case, see <i>Example 1</i> .
3	Row-partitioned tables ^{#3} are used in different SQL statements in the same transaction (or across multiple transactions ^{#1}), and there is a restriction ^{#4} that partitioning column searches are performed by only one back-end server.	From among the row-partitioned tables for which partitioning column searches are restricted to being performed by one back-end server in the same transaction, HiRDB randomly allocates a back-end server from among the back-end servers used by the SQL statement that most recently accessed a row-partitioned table. For an example of how the back-end servers are allocated in this case, see <i>Example 2</i> .
4	Unpartitioned tables ^{#5} are used in different SQL statements in the same transaction (or across multiple transactions ^{#1}).	From among the unpartitioned tables searched in the same transaction, HiRDB randomly allocates a back-end server from among the back-end servers used by the SQL statement that most recently accessed an unpartitioned table. For an example of how the back-end servers are allocated in this case, see <i>Example 3</i> .
5	Row-partitioned tables are used in different SQL statements in the same transaction (or across multiple transactions ^{#1}).	From among the row-partitioned table searched in the same transaction, HiRDB randomly allocates a back-end server from among the back-end servers used by the SQL statement that most recently accessed a row-partitioned table. For an example of the how back-end servers are allocated in this case, see <i>Example 4</i> .
6	There is a back-end server in the same unit as the connected front-end server.	HiRDB randomly allocates a back-end server from among the back-end servers in the same unit as the front-end server. For an example of the how back-end servers are allocated in this case, see <i>Example 5</i> .
7	Cases other than search conditions 1 through 6	HiRDB randomly allocates a back-end server. For an example of the how back-end servers are allocated in this case, see <i>Example 6</i> .

#1

Refers to use of the BES connection holding facility, holdable cursor, and local buffers in AP units.

#2

Exceptions are shared tables on which a `LOCK TABLE` statement that specifies `IN EXCLUSIVE MODE` is executed, and shared tables that perform searches using `FOR UPDATE` clauses.

#3

Exceptions are flexible hash partitions and partitions in a single back-end server.

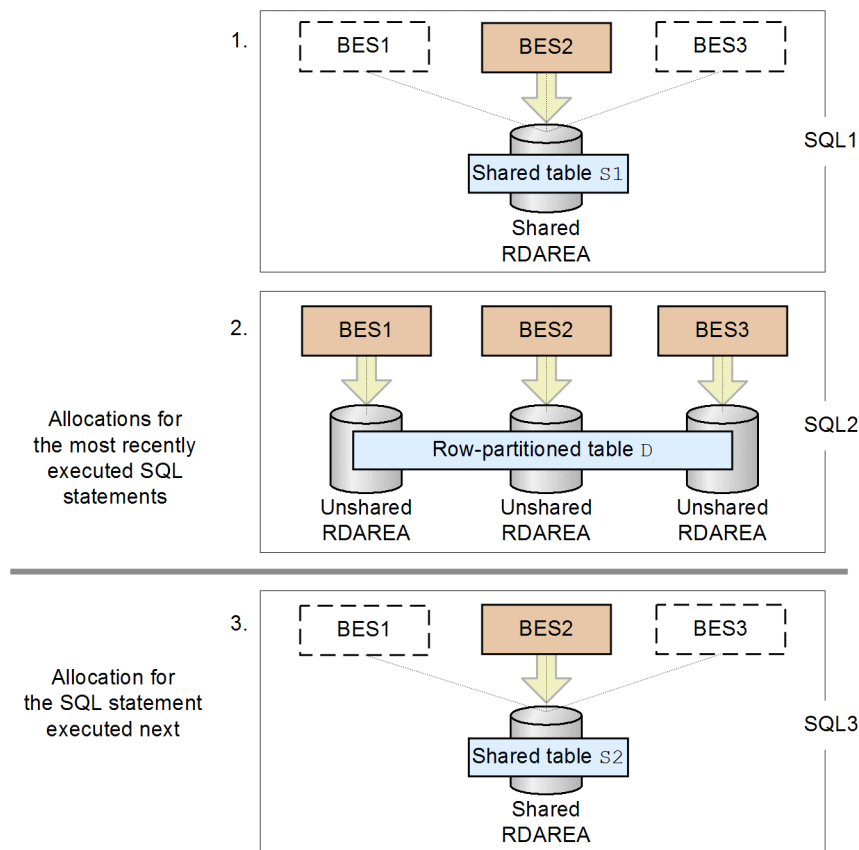
#4

This is a condition (predicate or a predicate on which an OR is executed) that specifies only the column of a single table in the search conditions.

#5

Includes shared tables on which a LOCK TABLE statement that specifies IN EXCLUSIVE MODE is executed, and shared tables that perform searches using FOR UPDATE clauses.

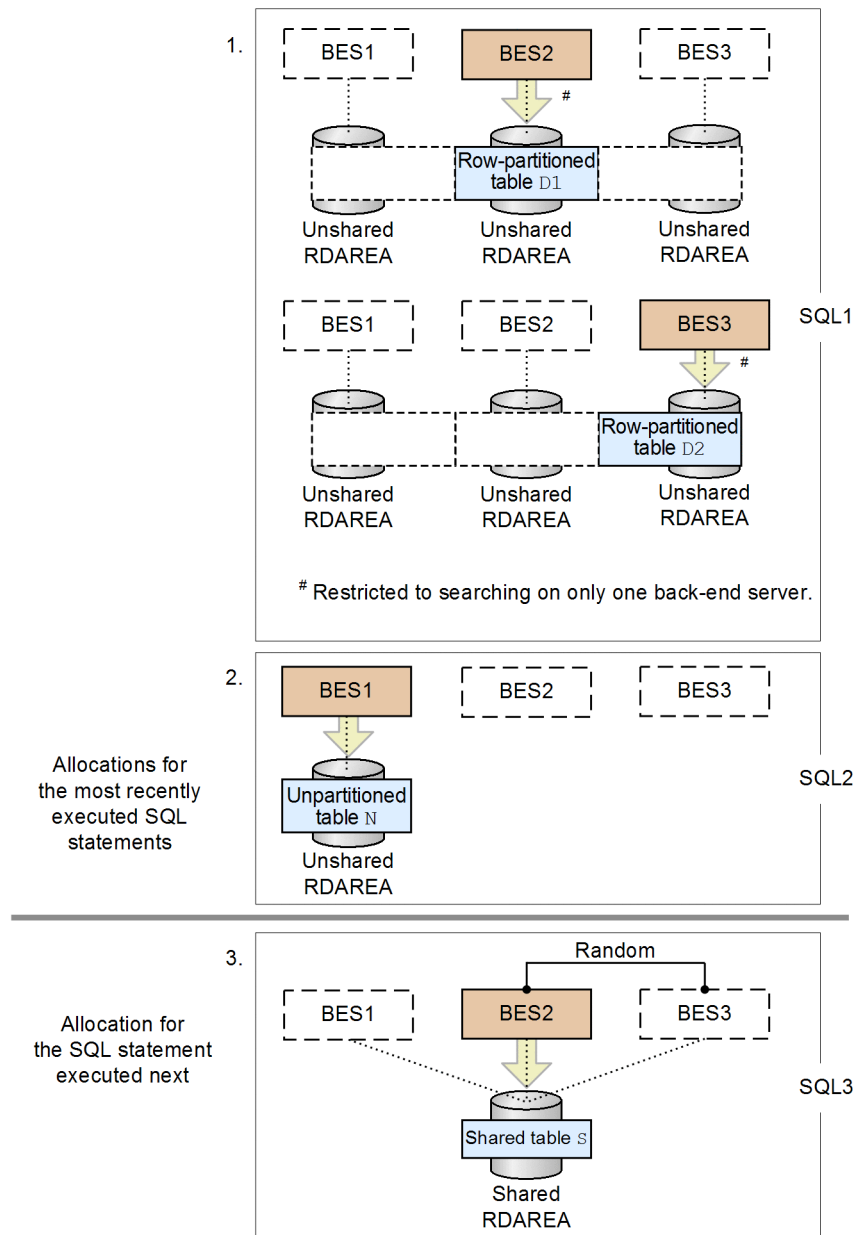
(a) Example 1



Explanation

1. In SQL1, the back-end server used to access-shared table S1 is BES2.
2. In SQL2, the back-end servers used to access row-partitioned table D are BES1, BES2, and BES3.
3. When shared table S2 is accessed in SQL3 immediately after SQL1 and SQL2, BES2, which was used to access shared table S1, is allocated.

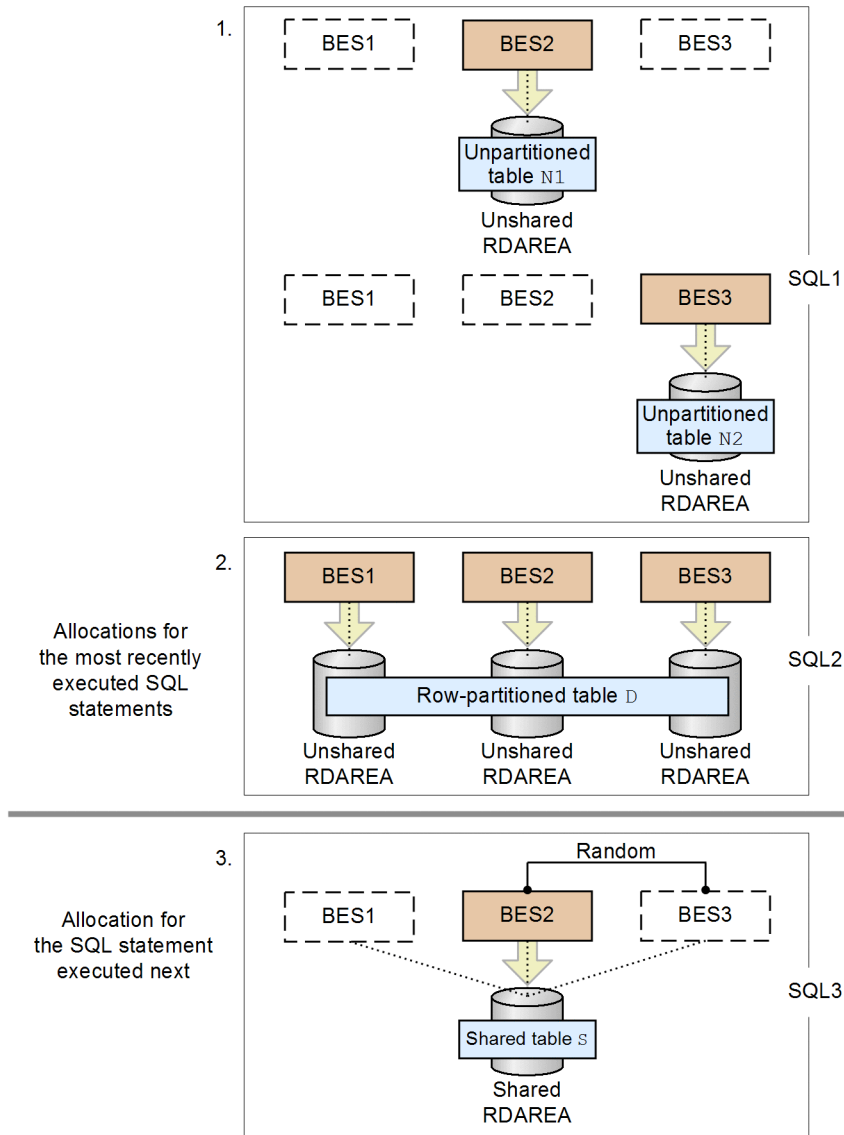
(b) Example 2



Explanation

1. In SQL1, row-partitioned table D1 and row-partitioned table D2 are searched with the restriction that only one back-end server can search in a partitioning column. The back-end server used to access row-partitioned table D1 is BES2, and the back-end server used to access row-partitioned table D2 is BES3.
2. In SQL2, the back-end server used to access unpartitioned table N is BES1.
3. When shared table S is accessed in SQL3 immediately after SQL1 and SQL2, BES2 is randomly selected from among the back-end servers that were used to access row-partitioned tables D1 and D2, and is allocated (since allocation is random, it might be BES3 in some cases).

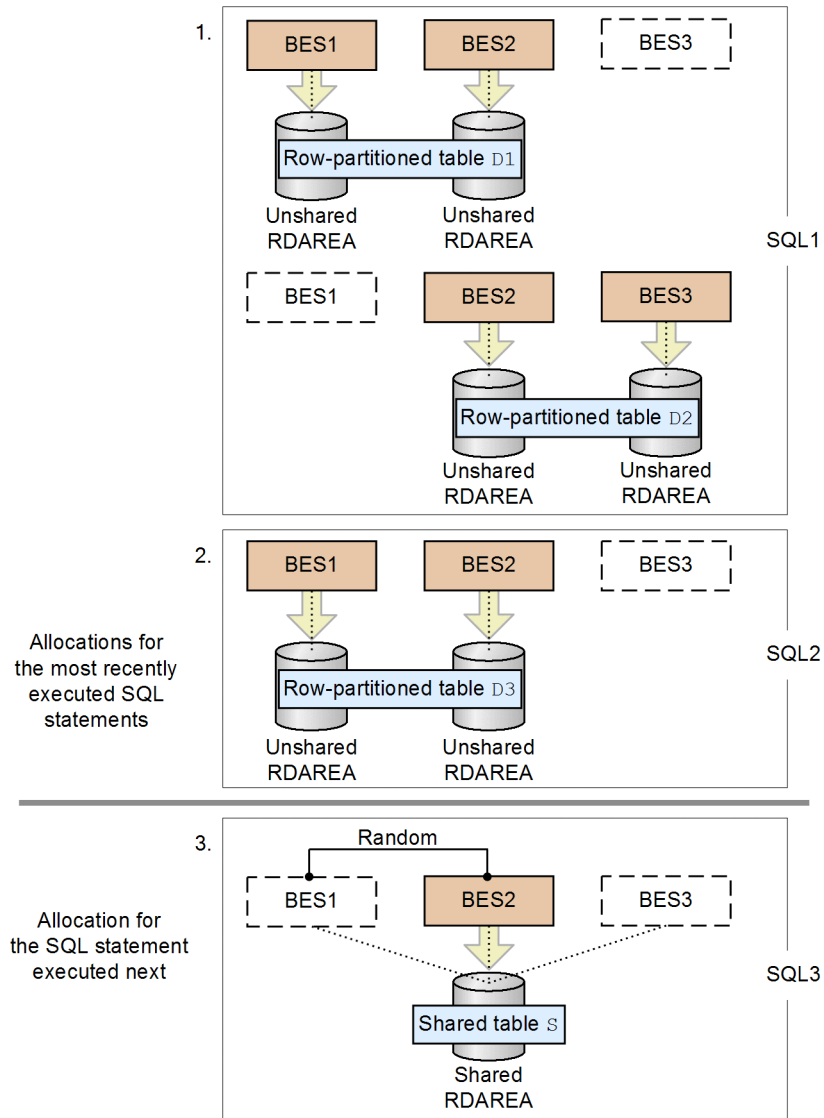
(c) Example 3



Explanation

1. In SQL1, unpartitioned table N1 and unpartitioned table N2 are searched. The back-end server used to access unpartitioned table N1 is BES2, and the back-end server used to access unpartitioned table N2 is BES3.
2. In SQL2, the back-end servers used to access row-partitioned table D are BES1, BES2, and BES3.
3. When shared table S is accessed in SQL3 immediately after SQL1 and SQL2, BES2 is randomly selected from among the back-end servers used to access unpartitioned tables N1 and N2, and is allocated. (since allocation is random, it might be BES3 in some cases).

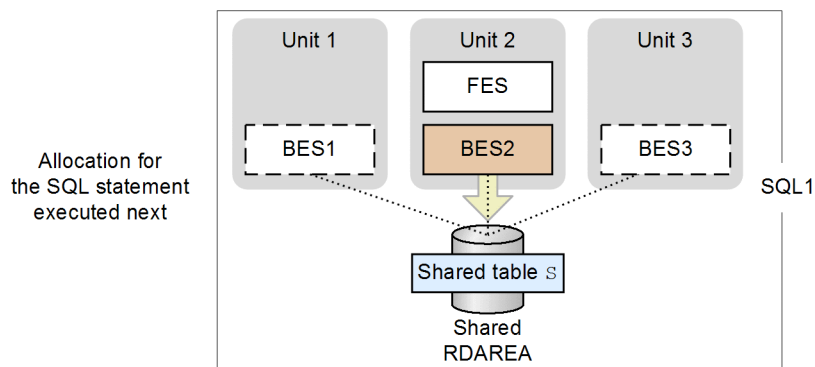
(d) Example 4



Explanation

1. In SQL1, row-partitioned table D1 and row-partitioned table D2 are searched. The back-end servers that are used to access row-partitioned table D1 are BES1 and BES2, and the back-end servers that are used to access row-partitioned table D2 are BES2 and BES3.
2. In SQL2, the back-end servers that are used to access row-partitioned table D3 are BES1 and BES2.
3. When shared table S is accessed in SQL3 immediately after SQL1 and SQL2, BES2 is randomly selected from among the back-end servers used to access row-partitioned table D3, and is allocated (since allocation is random, it might be BES1 in some cases).

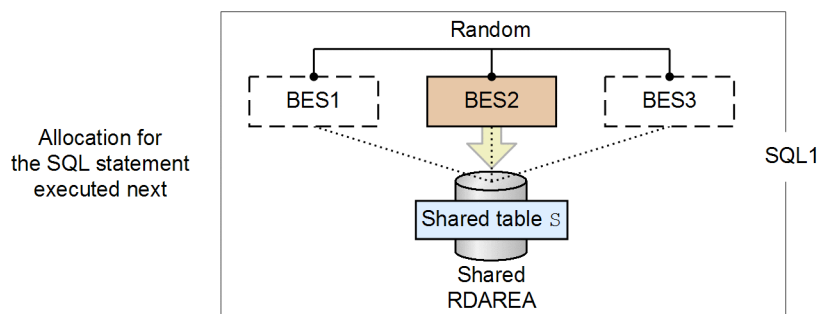
(e) Example 5



Explanation

When shared table *S* is accessed immediately after a transaction starts, the back-end server BES2 in the same unit as the front-end server is allocated.

(f) Example 6



Explanation

When shared table *S* is accessed immediately after a transaction starts, the randomly selected back-end server BES2 is allocated (since allocation is random, it might be BES1 or BES3 in some cases).

(2) When multiple tables that contain shared tables are specified in a single SQL statement

When multiple tables that include a shared table are specified in a single SQL statement, allocation of the back-end servers that search the shared table is determined by conditions such as the sort of search that is performed in that single SQL statement. When both shared and unshared tables are included in a single SQL statement, HiRDB allocates a back-end server to the unshared table before allocating a back-end server to the shared table.

The following table lists the rules used for allocating back-end servers that search shared tables (when multiple tables that contain shared tables are specified in a single SQL statement). Item 1 has the highest priority.

Table 12–11: Rules used for allocating back-end servers that search shared tables (when multiple tables that contain shared tables are specified in a single SQL statement)

Item	Search conditions	Back-end server allocated
1	One of the following conditions is met: <ul style="list-style-type: none"> A LOCK TABLE statement that specifies IN EXCLUSIVE MODE is executed on the shared table to be searched. A search is performed using a FOR UPDATE clause. 	HiRDB allocates an updatable back-end server.
2	Shared tables ^{#1} are used in the same SQL statement.	When multiple shared tables are used in a single SQL statement, HiRDB allocates the same back-end server. When all the tables are

Item	Search conditions	Back-end server allocated
		shared tables, HiRDB allocates according to <i>Table 12-10 Rules used for allocating back-end servers that search shared tables (when all tables specified in a single SQL statement are shared tables)</i> . For an example of how back-end servers are allocated in this case, see <i>Example 7</i> .
3	Row-partitioned tables ^{#2} are used in a single SQL statement, and there is a restriction ^{#3} to perform searches on only one back-end server in a partitioning column.	HiRDB randomly allocates a back-end server from among the back-end servers used to access the row-partitioned tables being used in a single SQL statement. For an example of how back-end servers are allocated in this case, see <i>Example 8</i> .
4	Unpartitioned tables ^{#4} are used in the same SQL statement.	HiRDB randomly allocates a back-end server from among the back-end servers used to access the unpartitioned tables being used in a single SQL statement. For an example of how back-end servers are allocated in this case, see <i>Example 9</i> .
5	Row-partitioned tables are used in the same SQL statement.	HiRDB randomly allocates a back-end server from among the back-end servers used to access the row-partitioned tables being used in a single SQL statement. For an example of how back-end servers are allocated in this case, see <i>Example 10</i> .

#1

Exceptions are shared tables that execute LOCK TABLE statements that specify IN EXCLUSIVE MODE, and shared tables that perform searches using FOR UPDATE clauses.

#2

Exceptions are flexible hash partitions and partitions in a single back-end server.

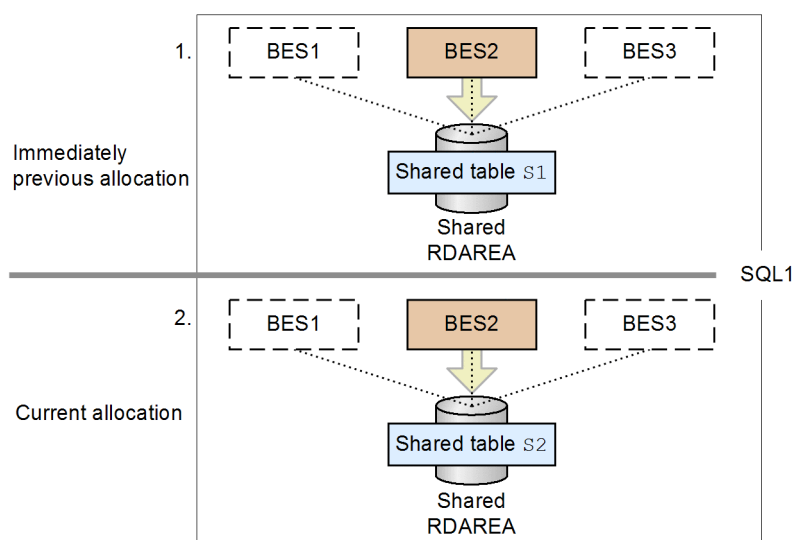
#3

This is a condition (predicate or a predicate on which an OR is executed) that specifies only the column of a single table in the search conditions.

#4

Includes shared tables on which a LOCK TABLE statement that specifies IN EXCLUSIVE MODE is executed, and shared tables that perform searches using FOR UPDATE clauses.

(a) Example 7

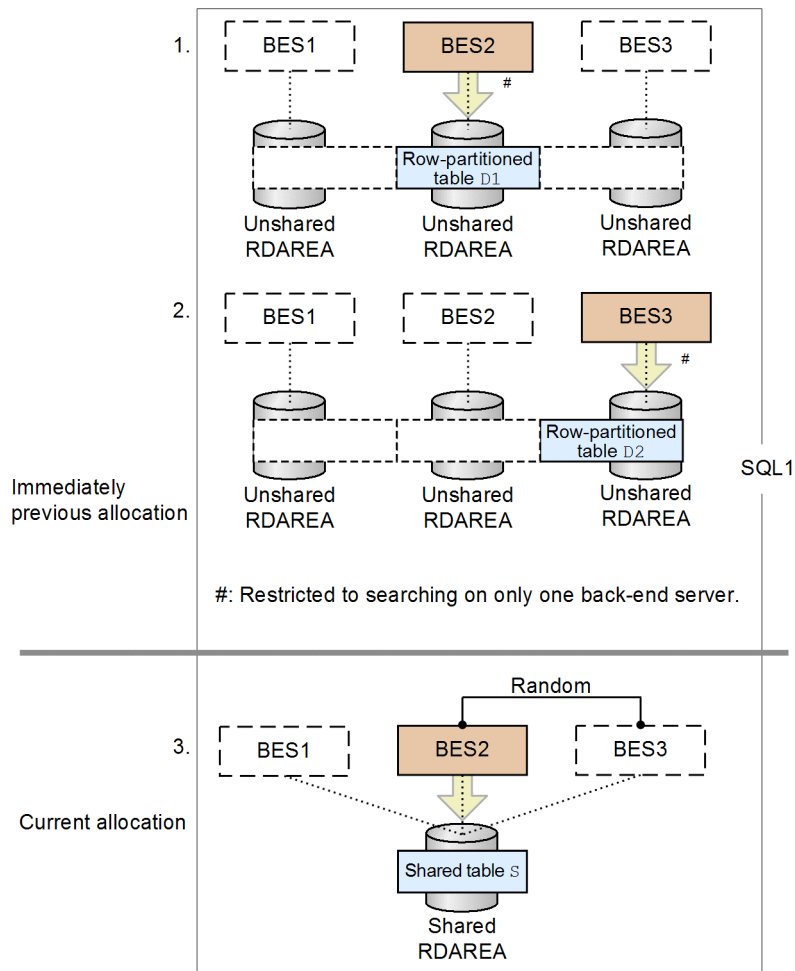


Explanation

In the same SQL statement, shared tables S1 and S2 are searched.

1. The back-end server used to access shared table *S*1 is *BES*2.
2. When shared table *S*2 is accessed immediately after step 1, back-end server *BES*2, which was used to access shared table *S*1, is allocated.

(b) Example 8

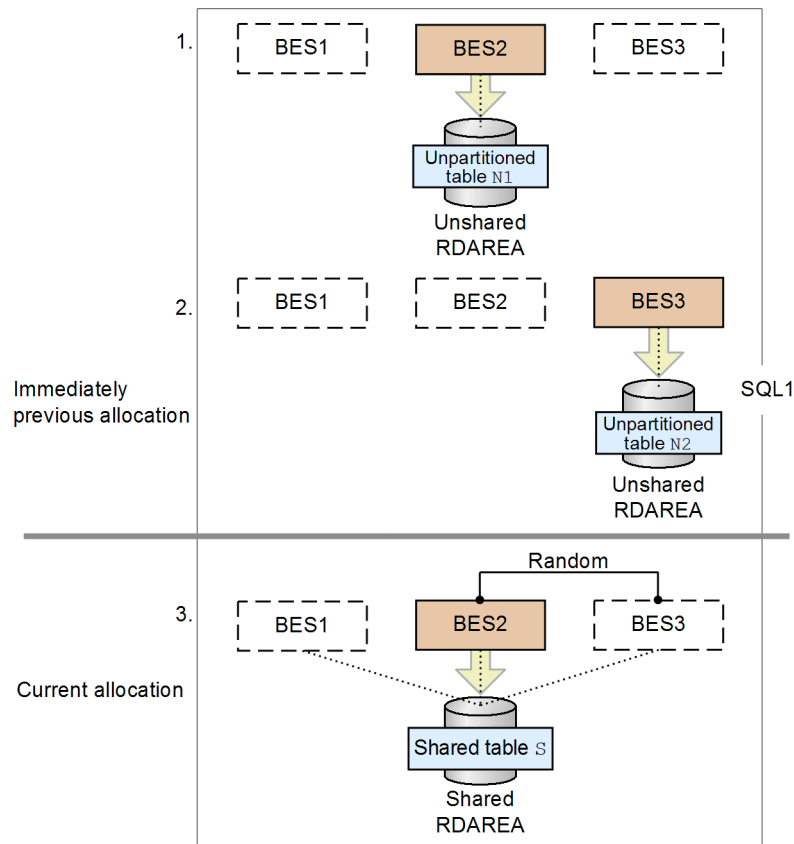


Explanation

In the same SQL statement, row-partitioned table *D*1, row-partitioned table *D*2, and shared table *S* are searched.

1. Row-partitioned table *D*1 is searched with the restriction that only one back-end server can search in a partitioning column. The back-end server used to access row-partitioned table *D*1 is *BES*2.
2. Row-partitioned table *D*2 is searched with the restriction that only one back-end server can search in a partitioning column. The back-end server used to access row-partitioned table *D*2 is *BES*3.
3. When shared table *S* is accessed immediately after steps 1 and 2, *BES*2 is randomly selected from among the back-end servers that were used to access row-partitioned tables *D*1 and *D*2, and is allocated (since allocation is random, it might be *BES*3 in some cases).

(c) Example 9

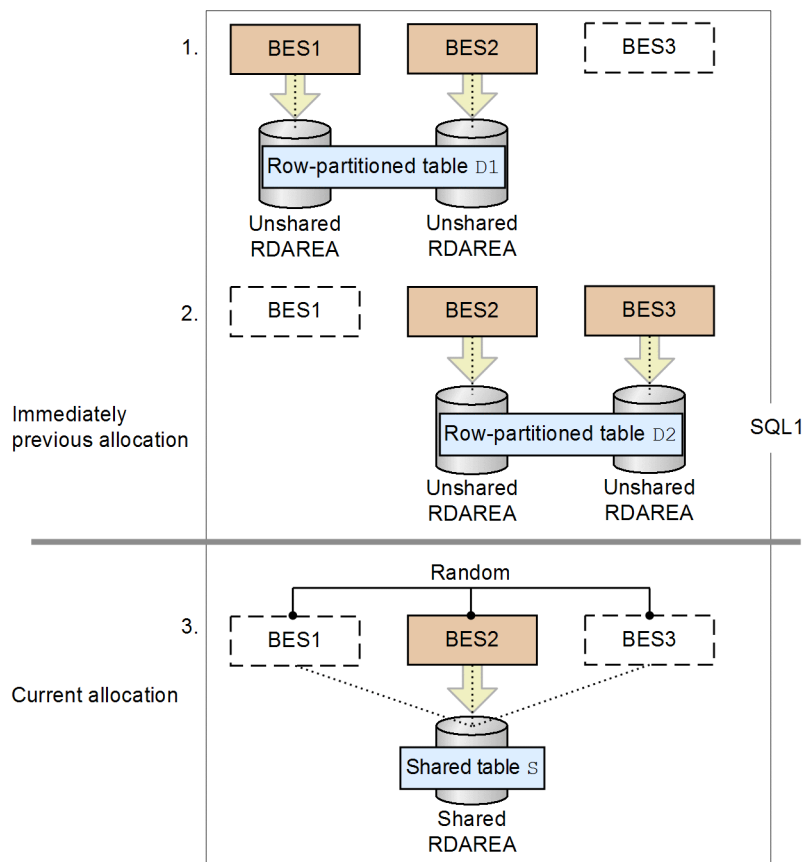


Explanation

In the same SQL statement, unpartitioned table N1, unpartitioned table N2, and shared table S are searched.

1. The back-end server used to access unpartitioned table N1 is BES2.
2. The back-end server used to access unpartitioned table N2 is BES3.
3. When shared table S is accessed immediately after steps 1 and 2, BES2 is randomly selected from among the back-end servers used to access unpartitioned tables N1 and N2, and is allocated (since allocation is random, it might be BES3 in some cases).

(d) Example 10



Explanation

In the same SQL statement, row-partitioned table D1, row-partitioned table D2, and shared table S are searched.

1. The back-end servers used to access row-partitioned table D1 are BES1 and BES2.
2. The back-end servers used to access row-partitioned table D2 are BES2 and BES3.
3. When shared table S is accessed immediately after steps 1 and 2, BES2 is randomly selected from among the back-end servers used to access row-partitioned tables D1 and D2, and is allocated (since allocation is random, it might be BES1 or BES3 in some cases).

12.18.6 Notes about execution of definition SQL statements, utilities, and operation commands

When definition SQL statements, utilities, and operation commands are used to process a shared table or shared index, HiRDB may internally issue the LOCK statement with IN EXCLUSIVE MODE specified and lock the target table and RDAREAs at all back-end servers. If any application accesses a table or index in a corresponding RDAREA, deadlock or server-to-server global deadlock may occur.

HiRDB internally issues the LOCK statement for the following definition SQL statements:

- CREATE TABLE, DROP TABLE, and PURGE TABLE for a shared table
- CREATE INDEX and DROP INDEX for a shared index
- DROP SCHEMA for a schema containing a shared table
- Change to the free space reuse facility for a shared table (ALTER TABLE)
- Add or remove the primary key for a shared table (ALTER TABLE)

HiRDB internally issues the `LOCK` statement for the following utilities:

- Database load utility (`pdload`)
- Database reorganization utility (`pdrorg -k reld, rorg, ixrc, ixmk, ixor`)
- Free page release utility (`pdreclaim`)
- Database definition utility (`pddef`)
- Data dictionary import/export utility (`pdexp`)
- Database structure modification utility (`pdmod -a initialize rdarea`)

HiRDB internally issues the `LOCK` statement for the following operation command:

- `pdorend`

For details about the utilities and operation commands that cannot be executed on shared RDAREAs, see *14.6(8) Notes about using shared RDAREAs*.

12.18.7 Using shared tables with a HiRDB single server configuration

This subsection describes for a HiRDB single server configuration the differences from using shared tables with a HiRDB parallel server configuration.

About notes

The notes about using shared tables with a HiRDB single server configuration (manipulation of shared tables, limitations on shared tables, and notes during execution of definition SQL statements, utilities, and operation commands) are basically the same as for a HiRDB parallel server configuration. The principal difference is that with a HiRDB single server configuration, deadlock between servers does not occur because there is only one server. Also, the notes about execution of operation commands for a HiRDB parallel server configuration do not apply to a HiRDB single server configuration.

About the RDAREAs for storing shared tables and shared indexes

In the case of a HiRDB single server configuration, you store shared tables and shared indexes in normal user RDAREAs because shared RDAREAs cannot be defined. You must provide separate user RDAREAs for storing shared tables and indexes from the user RDAREAs for storing non-shared tables and indexes. If the same user RDAREA contains both shared and non-shared tables or indexes, deadlock may occur (while a shared table is being updated, the RDAREAs containing the shared table and shared index are locked. If any application accesses a table or index in these RDAREAs, the application is placed in lock-release wait status).

About the use of local buffers

If a shared table or shared index is updated using local buffers on a HiRDB single server configuration without issuing a `LOCK` statement and the server process terminates abnormally, HiRDB does not terminate abnormally with abort code `Phb3008`.

About migrating from a HiRDB single server configuration to a HiRDB parallel server configuration

If you are migrating from a HiRDB single server configuration to a HiRDB parallel server configuration, make sure that you do not use the database structure modification utility (`pdmod`) while shared tables and shared indexes are still defined in the HiRDB single server configuration system. The migration procedure is as follows:

1. Check the HiRDB single server configuration for any defined shared tables or shared indexes.

Execute the SQL statement shown below (search the `SQL_TABLES` data dictionary table to check for the names of any defined shared tables in the system). If no table names are displayed, no shared tables are defined. If table names are displayed, those tables are defined.

```
SELECT TABLE NAME
FROM MASTER.SQL_TABLES
WHERE SHARED='S'
WITHOUT LOCK NOWAIT
```

2. Delete all shared tables and shared indexes that are defined in the HiRDB single server configuration.
3. Use the database structure modification utility (`pdmod`) to migrate the HiRDB single server configuration to a HiRDB parallel server configuration.

4. Define shared RDAREAs in the HiRDB parallel server configuration, check and, if necessary, revise the shared tables and shared indexes, then store them in shared RDAREAs.

12.19 Referential constraints

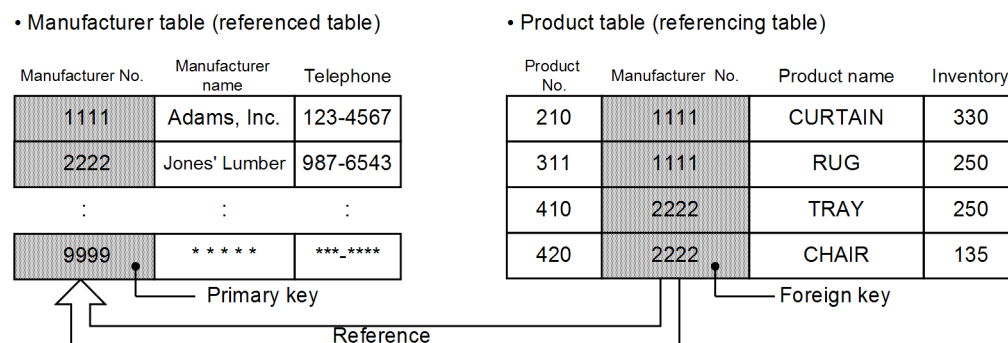
12.19.1 About referential constraints

The tables in a database may not all be independent, because some tables may be related to one another. Some data in a table may serve no purpose if related data does not exist in another table. To maintain referential conformity in data between tables, a *referential constraint* can be defined for a particular column (called a *foreign key*) when the table is defined. A table in which a referential constraint and a foreign key are defined is called a *referencing table*, while a table that is referenced from a referencing table by means of such a foreign key is called a *referenced table*. A *primary key*, which is referenced by one or more foreign keys, must be defined in the referenced table.

Execution of SQL code or utilities may cause loss of guaranteed data integrity between referenced and referencing tables. In such cases, the referencing table is placed in check pending status. For details about check pending status, see 12.19.3 *Check pending status*. For details about operations that cause loss of guaranteed data integrity, see 12.19.4 *Data manipulation and integrity*.

The figure below shows examples of a referenced and a referencing table. In this example, PRODUCT_TABLE is the referencing table and MANUFACTURER_TABLE is the referenced table. The primary key is referenced by a foreign key in the referencing table to obtain the name of a manufacturer.

Figure 12–32: Example of referenced and referencing tables



When you define a referential constraint, defining an index for the foreign key improves throughput. However, if the primary key values in the referenced table are not updated, updating performance may be affected adversely due to the overhead associated with the index updating that results when a foreign key value is updated.

Effects of referential constraints

When you define a referential constraint, the workload associated with UAP creation can be reduced because checking of data integrity on tables and data manipulation can be automated. However, note that the processing time for checking increases when referenced and referencing tables are updated, because data integrity is checked. Processing time increases for checks in the following cases.

- The column to be updated is the primary key of the referenced table.
The more foreign keys there are in the referencing table that references the primary key of the referenced table, the greater the delay.
- The column to be updated is the foreign key of the referencing table.
The more foreign keys there are that have the column to be updated as a constituent column, the greater the delay.

12.19.2 Defining referential constraints

To enable one or more referential constraints, you must first define in the referenced table the primary key that is to be referenced by the foreign key (or keys). To do so, use the CREATE TABLE definition SQL statement to specify PRIMARY KEY in the referenced table. To use check pending status, specify USE in the pd_check_pending operand or do not specify (omit) the operand.

For the referencing table, you specify `FOREIGN KEY` along with the following information in the `FOREIGN KEY` clause:

- Referencing column
- Referenced table
- Referential constraint action

For the referential constraint action, use `CASCADE` or `RESTRICT` to specify the action that is to be taken on the referencing table or referenced table when an operation such as insertion, updating, or deletion is performed.

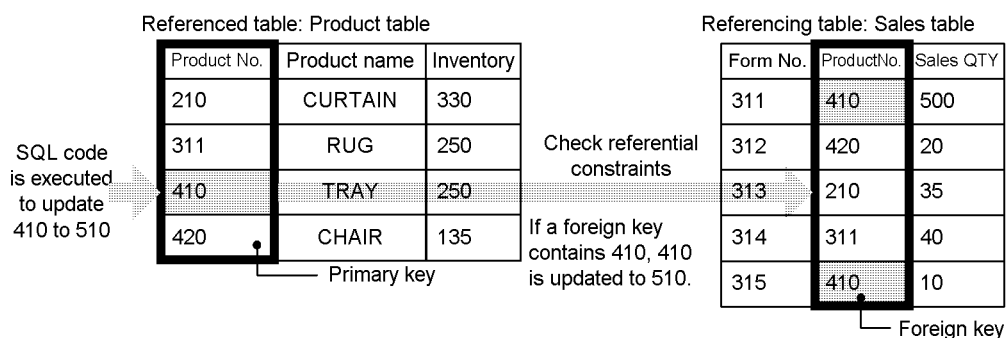
The following subsections explain the actions in the referenced and referencing tables when `CASCADE` or `RESTRICT` is specified.

(1) If `CASCADE` is specified

If `CASCADE` is specified and a change is executed on a primary key value of a referenced table, the referencing foreign key value will also be changed in the same manner. When a foreign key of a referencing table is changed, a check is performed to determine if there is a row containing a primary key whose value is the same as the value of the foreign key after the change; the foreign key is not changed if such a change would result in a referential constraint violation.

Figures 12-33 and 12-34 show examples of the actions that occur if `CASCADE` is specified when SQL code is executed on a referenced table and on a referencing table.

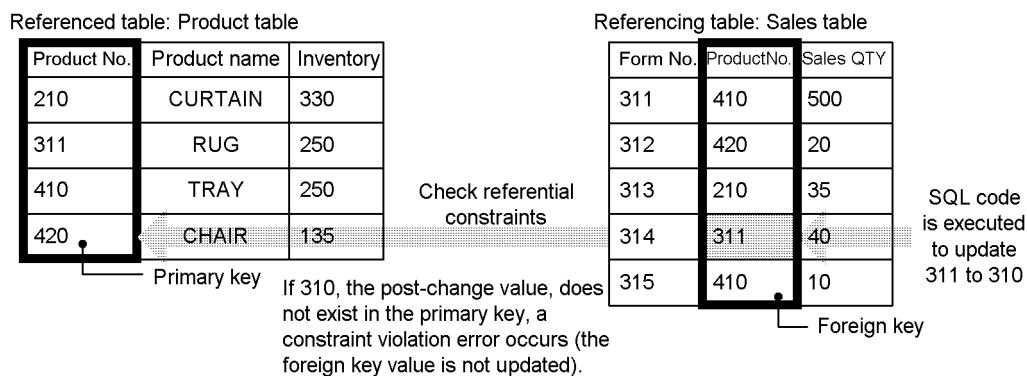
Figure 12-33: Example of the actions that occur when update SQL code is executed on a referenced table (with `CASCADE` specified)



Explanation:

If there is a value in a foreign key that is the same as the value in the primary key, to maintain constraints, the foreign key value is changed in the same way that the primary key value is changed. In the above case, updating of the referenced table is performed. Insertion and deletion are handled in the same manner.

Figure 12-34: Example of the actions that occur when update SQL code is executed on a referencing table (with `CASCADE` specified)



Explanation:

If there is a value in the primary key that is the same as the value in the foreign key after it is updated, updating of the foreign key value is performed. Updating of the foreign key value is also performed if any foreign key in the referencing table contains a null value, even if no value exists in the primary key that is the same as the updated foreign key value. If neither of the above is true, a referential constraint violation results. If this occurs, there is no effect on the referenced table. Insertion and deletion are handled in the same manner.

Table 12-12 lists primary key operations and describes the resulting actions that occur in the referencing table when CASCADE is specified. Table 12-13 lists foreign key operations and describes the resulting actions that occur in the referenced table when CASCADE is specified.

Table 12-12: Primary key operations and the resulting actions that occur in the referencing table (with CASCADE specified)

Primary key manipulation	Relationship between rows in referenced and referencing tables	Result of primary key operation	Action in referencing table
Insert (INSERT statement)	None	Y	None
Update (UPDATE statement), delete (DELETE statement)	The referencing table has a value in a foreign key that is the same as a value in the primary key before the update is performed.	Y	The update is performed with the same value as that in the primary key, or the rows are deleted.
	The referencing table does not have a value in a foreign key that is the same as a value in the primary key before the update is performed.	Y	None

Legend:

Y: Executed normally.

Table 12-13: Foreign key operations and the resulting actions that occur in the referenced table (with CASCADE specified)

Foreign key manipulation	Relationship between rows in referenced and referencing tables		Result of foreign key operation	Action in referenced table
Insertion (INSERT statement)	The referenced table has a value in its primary key that is the same as a value in a foreign key of the rows to be inserted.		Y	None
	The referenced table does not have a value in its primary key that is the same as a value in a foreign key of the rows to be inserted.	A foreign key contains a null value.	Y	
		A foreign key does not contain a null value.	N	
Update (UPDATE statement)	The referenced table has a value in its primary key that is the same as the updated foreign key value.		Y	None
	The referenced table does not have a value in its primary key that is the same as the updated foreign key value.	A foreign key contains a null value.	Y	
		A foreign key does not contain a null value.	N	
Delete (DELETE statement)	None		Y	None

Legend:

Y: Executed normally.

N: A constraint violation error occurs.

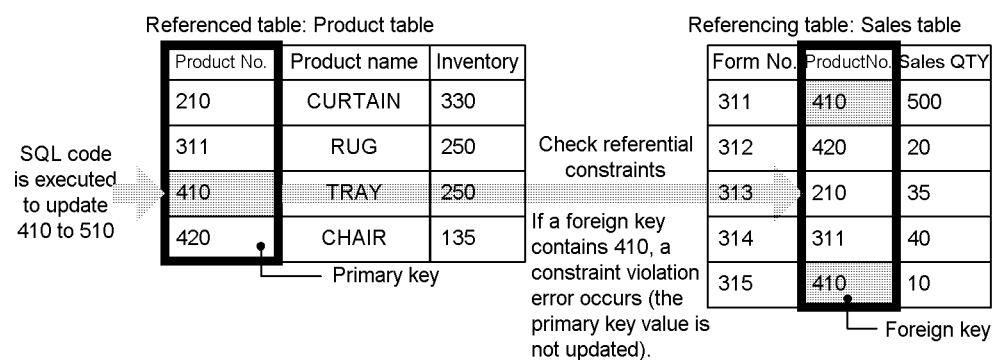
Note that when you specify `CASCADE`, HiRDB internally generates a trigger during table definition to update the foreign key value with the change made in the primary key value. For details about triggers for referential constraint actions and about user-defined triggers, see *12.19.6 Referential constraints and triggers*.

(2) If `RESTRICT` is specified

If `RESTRICT` is specified and a change is executed on a primary key value of a referenced table, a referential constraint violation occurs if there is a value in a foreign key that is the same as the value in the primary key after it has been updated. In this case, the primary key value is not changed. If a change is executed on a foreign key value, a check is performed to determine if there is a value in the primary key that is the same as the updated foreign key value. If a referential restraint violation error occurs, updating is not performed on the foreign key value.

The figure below shows an example of the actions that occur when update SQL code is executed on a referenced table and `RESTRICT` is specified. The actions in a referencing table are the same as those when `CASCADE` is specified (see Figure 12-34).

Figure 12–35: Example of the actions that occur when update SQL code is executed on a referenced table (with `RESTRICT` specified)



Explanation:

If there is a value in a foreign key that is the same as a value in the primary key, a referential constraint violation error occurs, and updating of the primary key value is not performed. If there is no foreign key value that is the same, updating of the referenced table is performed. Insertion and deletion are handled in the same manner.

The table below lists primary key operations when `RESTRICT` is specified, and resulting actions in referenced and referencing tables. Foreign key operations and describes the resulting actions that occur in the referenced table are the same as those when `CASCADE` is specified (see Table 12-13).

Table 12–14: Primary key operations and the resulting actions that occur in referenced and referencing tables

Primary key manipulation	Relationship between rows in referenced and referencing tables	Result of primary key operation	Action in referencing table
Insertion (INSERT statement)	None	Y	None
Update (UPDATE statement), delete (DELETE statement)	The referencing table has a value in a foreign key that is the same as a value in the primary key before the update is performed.	N	None
	The referencing table does not have a value in a foreign key that is the same as a value in the primary key before the update is performed.	Y	

Legend:

Y: Executed normally.

N: A constraint violation error occurs.

(3) Constraint items in defining referenced and referencing tables

The following notes explain constraint items in table definition, table definition change, and table deletion performed on referenced and referencing tables.

(a) Defining tables (CREATE TABLE)

- The same referenced table cannot be referenced from the foreign keys of columns that constitute that foreign key (the line-up need not be the same).
- A foreign key cannot be defined in the following cases:
 - When `WITHOUT ROLLBACK` is specified for the table, or the table is a shared table or falsification prevented table.
 - When a primary key is defined in a table for which `WITHOUT ROLLBACK` is specified, and that primary key is referenced.
- A maximum of 255 foreign keys can be defined in one table.
- A maximum of 255 foreign keys can be defined to reference a single primary key.
- Only tables of the same schema can be referenced when a referencing table is defined.
- When both of the following conditions are met, you can define a referencing table for which `ON UPDATE CASCADE` (referential constraint action during updating is `CASCADE`) that references the same primary key in one table is specified.
 - There are no duplicated foreign key columns.
 - No check constraint or referential constraint related to multiple foreign key columns is defined.
- Use the same character set for the foreign key and for the primary key of the table referenced from that foreign key.

(b) Changing table definitions (ALTER TABLE)

- Table definitions cannot be changed using the `DROP` or `RENAME` clause for a referenced or referencing table.
- When you change the definition in a referenced table for the primary key and foreign key columns, the following restrictions apply:
 - The `CHANGE` clause cannot be used to change data type or data length.
 - The `RENAME` clause cannot be used to change the column name.
- Specification of `WITH PROGRAM` invalidates SQL object functions, procedures and triggers. You need to re-create them using `ALTER ROUTINE`, `ALTER PROCEDURE`, or `ALTER TRIGGER`.

(c) Deleting tables (DROP TABLE)

- A table that is referenced by a foreign key cannot be deleted.

(4) Notes on defining referential constraints

- Deadlock between a referenced and referencing table
 If both of the following conditions are met, deadlock may occur between a referenced and referencing table. These conditions are the same regardless of whether the referential constraint action is `RESTRICT` or `CASCADE`.
 - Two separate transactions occur: one that updates rows in the referencing table and the other that updates the referenced table, and both transactions are executed simultaneously.
 - A value in the primary key of the rows to be updated in the referencing table is the same as a value in a foreign key of the rows to be updated in the referenced table.
 When you manipulate referenced and referencing tables, make sure that at least one of the above conditions is not true. You can also guarantee data integrity by using the `LOCK` statement's lock mode to lock the target table. Doing so, however, may somewhat degrade concurrent execution efficiency.
- Estimating the size of the SQL object buffer length

When you specify a referential constraint action, HiRDB internally generates a trigger to check constraint conditions or execute a referential constraint action. Therefore, you must take these SQL objects into account when specifying the SQL object buffer length. For details about how to estimate the SQL object buffer length (pd_sql_object_cache_size), see the manual *HiRDB Version 9 System Definition*.

- Estimating the size of the data dictionary LOB RDAREA

When you specify *CASCADE* for a referential constraint action, HiRDB generates a trigger to execute the action. The SQL object that defines the trigger action procedure of this trigger is stored in the data dictionary LOB RDAREA. Therefore, when you specify *CASCADE* for a referential constraint action, you need to allocate sufficient space for the data dictionary LOB RDAREA. For details about estimating the size of the data dictionary LOB RDAREA, see *16.5 Determining the size of a data dictionary LOB RDAREA*.

- Backup data

You must back up data at the same time for all RDAREAs in which referenced tables and referencing tables are stored. If you use the inner replica facility, acquire backup for the generation number of all RDAREAs as well. If you use the inner replica facility, back up the generation number of all RDAREAs as well.

The extent of the data that is backed up depends on the check pending status at the time of backup. For details about the backup time and the extent of the data that is backed up, see *RDAREAS to be backed up together* in the *HiRDB Version 9 System Operation Guide*.

(5) Referential constraint definition examples

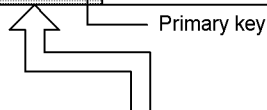
The following section provides examples of how to define referential constraints.

(a) Example of defining a referential constraint with a 1-to-1 correspondence

This example defines a referential constraint where the referenced and referencing tables have a 1-to-1 correspondence.

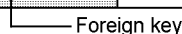
- Manufacturer table (MANUFACTURER)

Manufacturer No. MNO	Manufacturer name MNAME	Telephone No. TELEPHONE
1111	Adams, Inc.	123-4567
2222	Jones' Lumber	987-6543
9999	* * * * *	***_****



- Product table (PRODUCT)

Product No. PNO	Manufacturer No. MNO	Product name PNAME	Inventory QTY
210	1111	CURTAIN	330
311	1111	RUG	250
410	2222	TRAY	250
420	2222	CHAIR	135



Definition example of a referential constraint (1)

```
CREATE TABLE MANUFACTURER
(MNO CHAR(4), MNAME NCHAR(6), TELEPHONE CHAR(12))
PRIMARY KEY(MNO) ...Specification of the primary key
CREATE TABLE PRODUCT
(PNO CHAR(4), MNO CHAR(4), PNAME NCHAR(10), QTY INTEGER)
CONSTRAINT PRODUCT_FK ...Specification of the constraint name
FOREIGN KEY(MNO) ...Specification of the foreign key
REFERENCES MANUFACTURER ...Specification of the referenced table name
```


Details of the referential constraint action

Because this example omits specification of a referential constraint action, RESTRICT is assumed during updating or deletion. If the MANUFACTURER_NO (primary key) of the MANUFACTURER_TABLE is updated or deleted and there is a row corresponding to the MANUFACTURER_NO (foreign key) of the PRODUCT_TABLE, a referential constraint violation error occurs. As a result, updating or deletion of the MANUFACTURER_NO of the MANUFACTURER_TABLE is suppressed.

Definition example of a referential constraint (2)

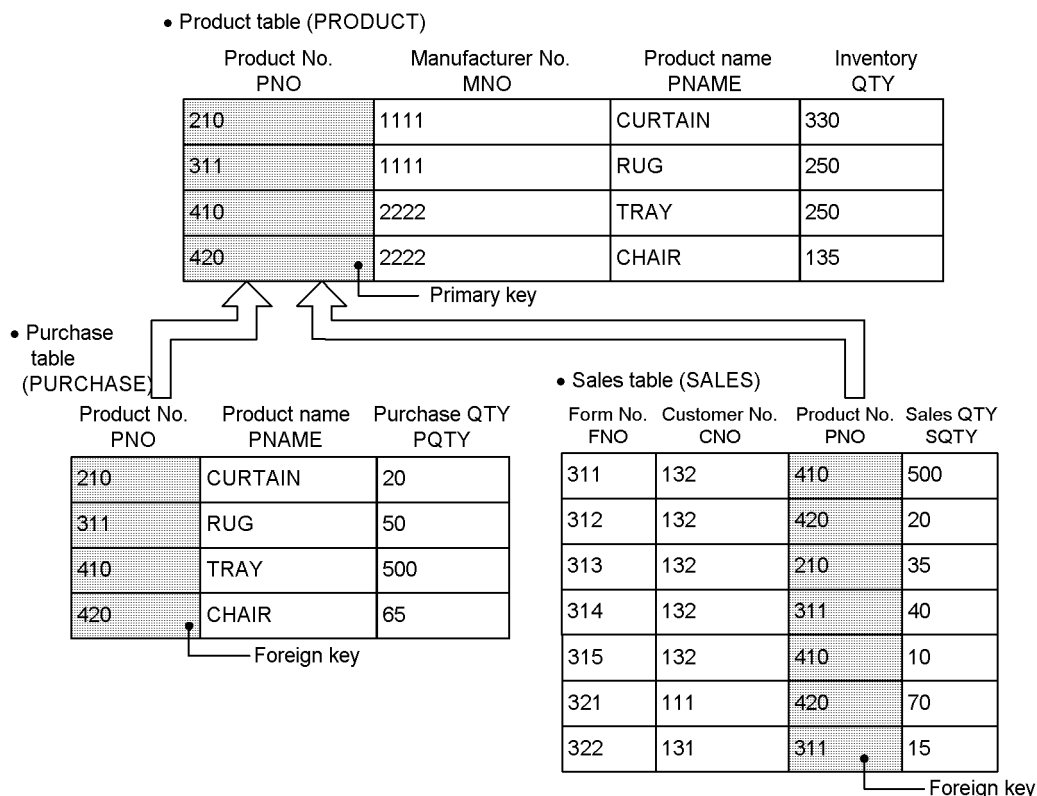
```
CREATE TABLE MANUFACTURER
(MNO CHAR(4), MNAME NCHAR(6), TELEPHONE CHAR(12))
PRIMARY KEY(MNO) ...Specification of the primary key
CREATE TABLE PRODUCT
(PNO CHAR(4), MNO CHAR(4), PNAME NCHAR(10), QTY INTEGER)
CONSTRAINT PRODUCT_FK ...Specification of the constraint name
FOREIGN KEY(MNO) ...Specification of the foreign key
REFERENCES MANUFACTURER ...Specification of the referenced table name
ON UPDATE CASCADE ...Specification of a referential constraint action on
update
ON DELETE CASCADE ...Specification of a referential constraint action on
deletion
```

Details of the referential constraint action

If the MANUFACTURER_NO (primary key) of the MANUFACTURER_TABLE is updated, the MANUFACTURER_NO (foreign key) of the corresponding PRODUCT_TABLE is also updated to the same value as for the primary key. If a row is deleted from the MANUFACTURER_TABLE, the row corresponding to the PRODUCT_TABLE is also deleted.

(b) Example of defining a referential constraint with a 1-to-2 correspondence

This example defines a referential constraint where there is one referenced table and two referencing tables.

**Definition example of a referential constraint**

```
CREATE TABLE PRODUCT
(PNO CHAR(4), MNO CHAR(4), PNAME NCHAR(10), QTY INTEGER)
PRIMARY KEY(PNO) ...Specification of the primary key
```

```

CREATE TABLE PURCHASE
(PNO CHAR(4), PNAME NCHAR(10), PQTY INTEGER)
CONSTRAINT PURCHASE_FK ...Specification of the constraint name
FOREIGN KEY(PNO) ...Specification of the foreign key
REFERENCES PRODUCT ...Specification of the referenced table name
ON UPDATE CASCADE ...Specification of a referential constraint action on
update
ON DELETE CASCADE ...Specification of a referential constraint action
on deletion
CREATE TABLE SALES
(FNO CHAR(4), CNO CHAR(4), PNO CHAR(4), SQTY INTEGER)
CONSTRAINT SALES_FK ...Specification of the constraint name
FOREIGN KEY(PNO) ...Specification of the foreign key

REFERENCES PRODUCT ...Specification of the referenced table name
ON UPDATE RESTRICT Specification of a referential constraint action on
update
ON DELETE RESTRICT ...Specification of a referential constraint action
on deletion

```

Details of the referential constraint action

If the PRODUCT_NO (primary key) of the PRODUCT_TABLE is to be updated and the SALES_TABLE contains a row whose PRODUCT_NO (foreign key) is the same as the primary key before updating, a referential constraint violation error occurs, in which case update processing is suppressed. If the SALES_TABLE contains no row that has the same value as the primary key before updating, the corresponding PRODUCT_NO in the PURCHASE_TABLE is also updated to the same value as the primary key.

If a row is to be deleted from the PRODUCT_TABLE and the SALES_TABLE contains a row that has the same value as the primary key before updating, a referential constraint violation error occurs, in which case the deletion processing is suppressed. If the SALES_TABLE contains no row that has the same value as the primary key, the corresponding row is also deleted from the PURCHASE_TABLE.

(c) Example of defining a referential constraint with a 2-to-1 correspondence

This example defines a referential constraint where there are two referenced tables and one referencing table.

• Product table (PRODUCT)

Product No. PNO	Manufacturer No. MNO	Product name PNAME	Inventory QTY
210	1111	CURTAIN	330
311	1111	RUG	250
410	2222	TRAY	250
420	2222	CHAIR	135

Primary key

• Customer table (CUSTOMER)

Customer No. CNO	Customer name CNAME	Address ADDR
111	ABC, Inc.	157 Main St...
112	Pets, Inc.	246 Seventh St...
113	Carter & Sons	3462 Park Ave...
131	King, Inc.	1234 Oak St...
132	Young, Inc.	847 Fifth St...

Primary key

• Sales table (SALES)

Form No. FNO	Customer No. CNO	Product No. PNO	Sales QTY SQTY
311	132	410	500
312	132	420	20
313	132	210	35
314	132	311	40
315	132	410	10
321	111	420	70
322	131	311	15

Foreign key

Definition example of a referential constraint

```

CREATE TABLE PRODUCT
(PNO CHAR(4),MNO CHAR(4),PNAME NCHAR(10),QTY INTEGER)
PRIMARY KEY(PNO) ...Specification of the primary key
CREATE TABLE CUSTOMER
(CNO CHAR(4),CNAME NCHAR(8),ADDR NCHAR(24))
PRIMARY KEY(CNO) ...Specification of the primary key
CREATE TABLE SALES
(FNO CHAR(4),CNO CHAR(4),PNO CHAR(4),SQTY INTEGER)
CONSTRAINT SALES_PRODUCT_FK ...Specification of the constraint name
FOREIGN KEY(PNO) ...Specification of the foreign key
REFERENCES PRODUCT ...Specification of the referenced table name
ON UPDATE CASCADE Specification of a referential constraint action on
update
ON DELETE CASCADE ...Specification of a referential constraint action on
deletion
CONSTRAINT SALES_CUSTOMER_FK
FOREIGN KEY(CNO) ...Specification of the foreign key

REFERENCES CUSTOMER ...Specification of the referenced table name
ON UPDATE CASCADE ...Specification of a referential constraint action on
update
ON DELETE CASCADE ...Specification of a referential constraint action on
deletion

```

Details of the referential constraint action

If the PRODUCT_NO (primary key) of the PRODUCT_TABLE is updated, the PRODUCT_NO (foreign key) of the SALES_TABLE is also updated to the same value. If a row is deleted from the PRODUCT_TABLE, the corresponding row is also deleted from the SALES_TABLE.

If the CUSTOMER_NO (primary key) of the CUSTOMER_TABLE is updated, the CUSTOMER_NO (foreign key) of the SALES_TABLE is also updated to the same value. If a row is deleted from the CUSTOMER_TABLE, the corresponding row is also deleted from the SALES_TABLE.

12.19.3 Check pending status

If data integrity between tables can no longer be guaranteed due to execution of an SQL statement or of a utility, HiRDB restricts data manipulation in the referencing table. The status in which data manipulation is restricted due to loss of guaranteed data integrity is called *check pending status*. To place a referencing table in check pending status for the purpose of restricting data manipulation, you must either specify `USE` in the `pd_check_pending` operand or do not specify (omit) the operand. You can use the integrity check utility (`pdconstck`) to clear the check pending status of a table. You can also use the integrity check utility to forcibly place a table into check pending status.

If you have specified `NOUSE` in the `pd_check_pending` operand, data manipulation is not restricted even when data integrity between tables cannot be guaranteed. In this case, if you execute an SQL statement or a utility that nullifies the guarantee of data integrity, you can use the integrity check facility to forcibly place the table into check pending status, and then check data integrity.

For details about operations that cause loss of guaranteed data integrity, see *12.19.4 Data manipulation and integrity*. For details about how to check data integrity, see *12.19.5 Procedure for checking table integrity*.

(1) Setting or clearing check pending status

You can also decide whether to set a referencing table to check pending status or clear its check pending status using the following utilities, commands and SQL statements, in addition to the integrity check utility.

- The `constraint` statement of the database load utility (`pdload`)
- The `constraint` statement of the database reorganization utility (`pdreorg`) (reload, reorganization)
- The database structure modification utility (`pdmod`) (re-initialization of RDAREA)
- Reflection processing of updatable online reorganization (`pdorend -p` command)
- The `PURGE TABLE` statement
- The `ALTER TABLE (CHANGE RDAREA)` statement

For details about utilities and commands, see the manual *HiRDB Version 9 Command Reference*. For details about SQL, see the manual *HiRDB Version 9 SQL Reference*.

(2) Managing check pending status

Check pending status is managed based on dictionary tables and on the table information of the RDAREAs in which the tables are stored. In dictionary tables, check pending status is managed for each table and constraint. In table information, check pending status is managed for each RDAREA if the table is a partitioned table, and for each table if the table is not a partitioned table.

The following table lists and describes the storage locations of check pending status information items.

Table 12–15: Storage locations of check pending status information and contents (referential constraint)

Storage location			Stored information
Dictionary table	SQL_TABLES table	CHECK_PEND column	Check pending status of referential constraint for each table
	SQL_REFERENTIAL_CONSTRAINTS table	CHECK_PEND column	Check pending status of referential constraint for each constraint
RDAREA table information		For unpartitioned table	Check pending status of referential constraint or check constraint for each table
		For partitioned table	Check pending status of referential constraint or check constraint for each RDAREA

(3) Operations that are restricted for tables in check pending status

The table below lists operations that are no longer available for tables once they enter check pending status. When a target table is accessed by a trigger action, restricted operations depend on the availability of SQL operations specified

in the triggered SQL statement. If a target table is a view table, the restricted operations depend on the availability of operations on the base table that is the source of the view table.

Table 12–16: Availability of operations on tables in check pending status

Operation on check pending status tables			Availability
Data manipulation SQL	SELECT statement	Searches the target table	Y ^{#1}
		Searches a list created from the target table	
	INSERT statement	Inserts data into the target table	
	UPDATE statement	Updates the target table	
	DELETE statement	Deletes a row from the target table	
	ASSIGN LIST statement	Creates a list from the target table	
Utility	Rebalancing utility (pdrbal)		N
	Database reorganization utility (pdorg)	Reorganizes	Y ^{#2}

Legend:

Y: The operation cannot be performed in certain cases.

N: The operation cannot be performed.

#1

The operation can be performed only when both of the following conditions are met:

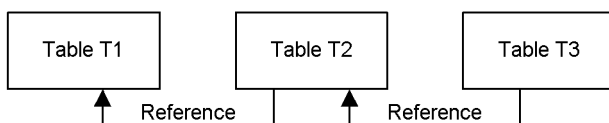
- The target table is a partitioned table, and the partitioning condition is key range partitioning or FIX hash partitioning.
- The target RDAREA is not in check pending status.

#2

Reorganizing a table partitioned using flexible hash partitioning may not be possible. For details, see *Rules and notes* in the *Database Reorganization Utility (pdorg)* chapter of the manual *HiRDB Version 9 Command Reference*.

(4) Operations restricted for tables that are related to a table in check pending status

In this example, tables have the following referential relationship; only tables T2 and T3 are in check pending status.



The following subsections explain operations restricted for each table when either table T2 or T3 or both tables are in check pending status.

(a) When only table T2 is in check pending status

The following table lists operations that are restricted for particular tables when table T2 alone is in check pending status.

Table 12–17: Restricted operations when table T2 is in check pending status

Target table	Restricted operation	Contents
Table T1	UPDATE (updates the target table)	Restrictions depend on the referential constraint action specification defined in table T2.
	DELETE (deletes rows from the target table)	
		<ul style="list-style-type: none"> • If CASCADE is specified:

Target table	Restricted operation	Contents
		<p>These operations cannot be performed if the table information of the RDAREA that is the target of referential constraint action is in check pending status. However, update operations can be performed if the values are the same.</p> <ul style="list-style-type: none"> If RESTRICT is specified: These operations can be performed. Referencing table T2 is referenced to perform data integrity checking.
Table T2	SELECT statement (searches the target table or a list created from the target table)	<p>These operations can be performed only when both of the following conditions are met:</p> <ul style="list-style-type: none"> The target table is a partitioned table and the partitioning condition is key range partitioning or FIX hash partitioning. The target RDAREA is not in check pending status.
	INSERT statement (inserts data into the target table)	
	UPDATE statement (updates the target table)	
	DELETE statement (deletes rows from the target table)	
	ASSIGN LIST statement (creates a list from the target table)	
	Rebalancing utility (pdrbal)	This operation cannot be performed.
	Reorganization by the database reorganization utility (pdrorg)	Reorganization may not be possible for a table partitioned using flexible hash partitioning. For details, see <i>Database Reorganization Utility (pdrorg)</i> in the manual <i>HiRDB Version 9 Command Reference</i> .
Table T3	There is no restricted operation. For INSERT and DELETE , referenced table T2 is referenced to perform data integrity checking.	

(b) When only table T3 is in check pending status

The following table lists operations that are restricted for particular tables when table T3 alone is in check pending status.

Table 12–18: Restricted operations when table T3 is in check pending status

Target table	Restricted operation	Contents
Table T1	UPDATE (updates the target table)	<p>When the referential constraint action defined for table T2 and T3 is CASCADE, these operations cannot be performed if the table information of the RDAREA that is the target of referential constraint action is in check pending status. However, update operations can be performed if the values are the same.</p>
	DELETE (deletes rows from the target table)	
Table T2	UPDATE (updates the target table)	<p>Restrictions depend on the referential constraint action specification defined for tables T2 and T3.</p> <ul style="list-style-type: none"> If CASCADE is specified: These operations cannot be performed if the table information of the RDAREA that is the target of referential constraint action is in check pending status. However, update operations can be performed if the values are the same. If RESTRICT is specified: These operations can be performed. Referencing table T3 is referenced to perform data integrity checking.
	DELETE (deletes rows from the target table)	
Table T3	SELECT statement (searches the target table or a list created from the target table)	<p>These operations can be performed only when both of the following conditions are met:</p> <ul style="list-style-type: none"> The target table is a partitioned table and the partitioning condition is key range partitioning or FIX hash partitioning.

Target table	Restricted operation	Contents
	INSERT statement (inserts data into the target table)	<ul style="list-style-type: none"> The target RDAREA is not in check pending status.
	UPDATE statement (updates the target table)	
	DELETE statement (deletes rows from the target table)	
	ASSIGN LIST statement (creates a list from the target table)	
	Rebalancing utility (pdrbal)	The operation cannot be performed.
	Reorganization by the database reorganization utility (pdrorg)	Reorganization may not be possible for a table partitioned using flexible hash partitioning. For details, see <i>Database Reorganization Utility (pdrorg)</i> in the manual <i>HiRDB Version 9 Command Reference</i> .

(c) When both tables T2 and T3 are in check pending status

The following table lists operations that are restricted for particular tables when both table T2 and table T3 are in check pending status.

Table 12–19: Restricted operations when tables T2 and T3 are in check pending status

Target table	Restricted operation	Contents
Table T1	UPDATE (updates the target table)	<p>If the referential constraint action defined for table T2 and T3 is CASCADE, these operations cannot be performed if the table information of the RDAREA that is the target of referential constraint action is in check pending status. However, update operations can be performed if the value are the same.</p> <p>These operations can be performed if the referential constraint action specification defined for tables T2 and T3 is RESTRICT. Referencing table T2 is referenced to perform data integrity checking.</p>
	DELETE (deletes rows from the target table)	
Table T2	SELECT statement (searches the target table or a list created from the target table)	<p>These operations can be performed only when both of the following conditions are met:</p> <ul style="list-style-type: none"> The target table is a partitioned table and the partitioning condition is key range partitioning or FIX hash partitioning. The target RDAREA is not in check pending status.
	INSERT statement (inserts data into the target table)	
	UPDATE statement (updates the target table)	
	DELETE statement (deletes rows from the target table)	
	ASSIGN LIST statement (creates a list from the target table)	
	Rebalancing utility (pdrbal)	This operation cannot be performed.
	Reorganization by the database reorganization utility (pdrorg)	Reorganization may not be possible for a table partitioned using flexible hash partitioning. For details, see <i>Database Reorganization Utility (pdrorg)</i> in the manual <i>HiRDB Version 9 Command Reference</i> .
Table T3	SELECT statement (searches the target table or a list created from the target table)	<p>These operations can be performed only when both of the following conditions are met:</p> <ul style="list-style-type: none"> The target table is a partitioned table and the partitioning condition is key range partitioning or FIX hash partitioning. The target RDAREA is not in check pending status.
	INSERT statement (inserts data into the target table)	

Target table	Restricted operation	Contents
	UPDATE statement (updates the target table)	
	DELETE statement (deletes rows from the target table)	
	ASSIGN LIST statement (creates a list from the target table)	
	Rebalancing utility (pdrbal)	This operation cannot be performed.
	Reorganization by the database reorganization utility (pdorg)	Reorganization may not be possible for a table partitioned using flexible hash partitioning. For details, see <i>Database Reorganization Utility (pdorg)</i> in the manual <i>HiRDB Version 9 Command Reference</i> .

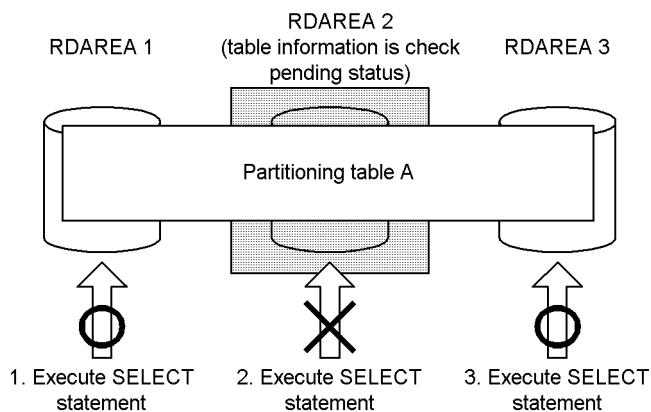
(5) When a partitioned table or the inner replica facility is used

Since the check pending status is managed for each RDAREA, if a partitioned table or the inner replica facility is used, and the table information in the RDAREA actually used is in check pending status, operation on the partitioned table or generation may be restricted. The following subsections explain these cases.

(a) For partitioned tables

The following figure shows an example of when some RDAREAs that store data in a partitioned table are in check pending status.

Figure 12–36: Data manipulation availability when managing check pending status for each RDAREA in a partitioned table



Explanation:

When you execute a `SELECT` statement for partitioned table A, if data actually manipulated is in RDAREA 2 (whose table information is in check pending status), a `SELECT` statement error occurs. When manipulating data in RDAREAs 1 and 3, the `SELECT` statement can be executed normally.

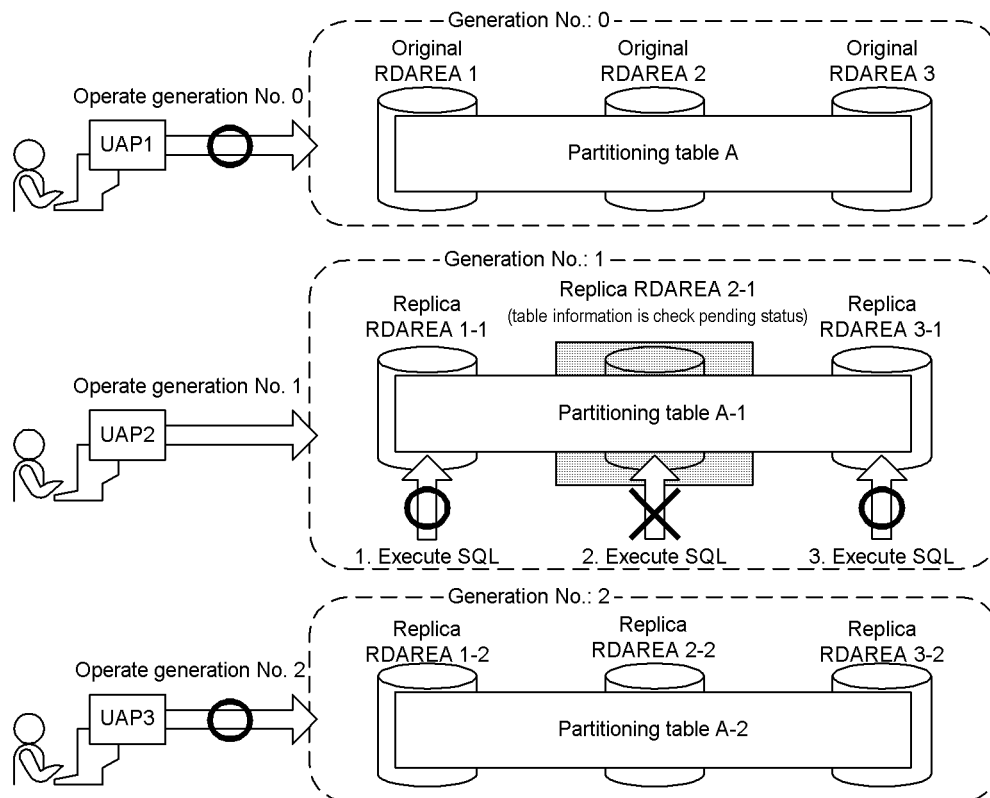
Notes on partitioned table

If you specify `USE` in the `pd_check_pending` operand and re-initialize the RDAREA where referencing table data is partitioned and stored, use the integrity check utility to check the data integrity of each table.

(b) When the inner replica facility is used

The following figure shows an example of when some RDAREAs of a specific generation are placed in check pending status using the inner replica facility.

Figure 12–37: Data manipulation availability when using the inner replica facility and managing check pending status by RDAREA



Explanation:

When manipulating data of generation 1 (the generation that includes the RDAREA whose table information is in check pending status), if the data that is actually being manipulated is in replica RDAREA 2-1, the SQL code results in an error.

(6) Notes on using check pending status

- If you change the value specified in the `pd_check_pending` operand from `NOUSE` to `USE`, you must use the integrity check utility to check the data integrity of the referencing table. For details about how to check data integrity, see *12.19.5 Procedure for checking table integrity*.
- Even if you have specified `USE` in the `pd_check_pending` operand and manipulated a table, causing loss of guaranteed data integrity, depending on the RDAREA status, you may not be able to set check pending status. For that reason, if you change the value specified in the `pd_check_pending` operand from `NOUSE` to `USE`, operations that can normally be performed when check pending status is not used could cause an error. The following explains the status of an RDAREA where the check pending status can be set when executing `PURGE TABLE` statement or `ALTER TABLE (CHANGE RDAREA)`.

When the open trigger attribute is INITIAL:

- RDAREA is not in hold and is in open status
- RDAREA is in updatable backup hold status and also in open status
- RDAREA is in online reorganization hold status and also in open status
- When RDAREA is in synchronous hold status and also open status, check pending status can be set after the hold status is cleared.

When the open trigger attribute is DEFER or SCHEDULE:

- RDAREA is not in hold status
- RDAREA is in updatable backup hold status

- RDAREA is in online reorganization hold status
- When RDAREA is in synchronous hold status, check pending status can be set after the hold status is cleared.

For details about RDAREAs on which check pending status can be set when the utility is executed, see *Whether or not the check pending status can be set of RDAREA Status During Command Execution* in the manual *HiRDB Version 9 Command Reference*.

- If you specify USE in the `pd_check_pending` operand, since lock is applied to referencing tables and RDAREAs that are set to check pending status, locked resources when a utility or SQL code is executed are different from those when check pending status is not used.

12.19.4 Data manipulation and integrity

When a referenced or referencing table is updated, added to, or deleted by a data manipulation SQL statement (excluding the `PURGE TABLE` statement), HiRDB performs checking during execution to guarantee data integrity. However, if the operations described in *Table 12-20* and *Table 12-21* are executed, data integrity may no longer be guaranteed. If you specify USE in the `pd_check_pending` operand and perform these operations, the referencing table is placed in check pending status.

Table 12–20: Operations on referenced tables that nullify the guarantee of data integrity and the conditions under which loss of data integrity occurs

Operation on table or RDAREA		Condition for loss of data integrity
Database load utility (<code>pdload</code>)	Data load of creation mode (<code>-d</code> option)	The loaded primary key column does not contain a value that is the same as a value in a foreign key column of the referencing table.
Database reorganization utility (<code>pdorg</code>)	Reload (<code>-k reld</code>)	The reloaded primary key column does not contain a value that is the same as a value in a foreign key column of the referencing table.
	Reorganization (<code>-k rorg</code>)	UOC was used to delete a row that contains a value that is the same as a value in a foreign key column of the referencing table.
Database structure modification utility (<code>pdmod</code>)	Reinitialization of RDAREA (<code>initialize rdarea</code>)	The referencing table is stored in an RDAREA that is different from the re-initialized RDAREA.
Reflection processing of updatable online re-organization (<code>pdorend</code> command)		<p>You perform any of operations (1) to (4) below on the replica RDAREA and original RDAREA in the current database during operation of updatable online reorganization[#] on tables having a referential relationship (data mismatch occurs after reflection processing):</p> <p>(1) When you perform operations in the following order:</p> <ol style="list-style-type: none"> 1. Insert data into the referencing table in the replica RDAREA. 2. From the referenced table in the original RDAREA, delete a row that has a value that is the same as one of the foreign key values inserted in step 1. <p>(2) When you perform operations in the following order:</p> <ol style="list-style-type: none"> 1. Update foreign key data in the referencing table in the replica RDAREA. 2. From the referenced table in the original RDAREA, delete a row that has a value that is the same as one of the foreign key values updated in step 1. <p>(3) When you perform operations in the following order:</p> <ol style="list-style-type: none"> 1. Delete data from the referenced table in the replica RDAREA. 2. In the referencing table in the original RDAREA, insert a row that has a value that is the same as one of the primary key values deleted in step 1.

Operation on table or RDAREA	Condition for loss of data integrity
	(4) When you perform operations in the following order: <ol style="list-style-type: none"> 1. Update data in the referenced table in the replica RDAREA. 2. In the referencing table in the original RDAREA, insert a row that has a value that is the same as one of the primary key values that existed before it was updated in step 1.
PURGE TABLE statement	Data exists in the referencing table.
Modification of table partition storage conditions by the ALTER TABLE	As a result of partitioning or integration of RDAREAs, a row that contains a value that is the same as a value in the foreign key column of the referencing table is not included.

#

For details about using updatable online reorganization, see the *HiRDB Version 9 Staticizer Option Description and User's Guide*.

Table 12–21: Operations on referencing tables that nullify the guarantee of data integrity and the conditions under which loss of data integrity occurs

Operation on table or RDAREA		Condition for loss of data integrity
Database load utility (pdload)	Data load	The loaded foreign key column does not contain a value that is the same as a value in the primary key column of the referenced table.
Database reorganization utility (pdreorg)	Reload (-k reld)	The reloaded foreign key column does not contain a value that is the same as a value in the primary key column of the referenced table.
Reflection processing of updatable online re-organization (pdorend command)		<p>You perform any of operations (1) to (5) on the replica RDAREA and original RDAREA in the current database during operation of updatable online reorganization[#] on tables having a referential relationship (data mismatch occurs after reflection processing):</p> <p>(1) When you perform operations in the following order:</p> <ol style="list-style-type: none"> 1. Insert data into the referencing table in the replica RDAREA. 2. From the referenced table in the original RDAREA, delete a row that has a value that is the same as one of the foreign key values inserted in step 1. <p>(2) When you perform operations in the following order:</p> <ol style="list-style-type: none"> 1. Update foreign key data in the referencing table in the replica RDAREA. 2. From the referenced table in the original RDAREA, delete a row that has a value that is the same as one of the foreign key values updated in step 1. <p>(3) When you perform operations in the following order:</p> <ol style="list-style-type: none"> 1. Delete data from the referenced table in the replica RDAREA. 2. In the referencing table in the original RDAREA, insert a row that has a value that is the same as one of the primary key values deleted in step 1. <p>(4) When you perform operations in the following order:</p> <ol style="list-style-type: none"> 1. Update data in the referenced table in the replica RDAREA. 2. In the referencing table in the original RDAREA, insert a row that has a value that is the same as one of the primary key values that existed before it was updated in step 1.

Operation on table or RDAREA	Condition for loss of data integrity
	(5) When you perform the following operation: Use the database load utility (<code>pdload</code>) to perform operation on the referencing table in the replica RDAREA, which causes loss of data integrity.

#

For details about using updatable online reorganization, see the *HiRDB Version 9 Staticizer Option Description and User's Guide*.

(1) When the target table is a partitioned table

If the target table is a partitioned table and the table contains mismatched data, execution of a utility may move the RDAREA in which the mismatched data is stored. For example, assume there is mismatched data in RDAREA 1 for a table that is partitioned and stored in RDAREAs 1, 2 and 3. Executing a utility could cause the mismatched data to be moved to RDAREA 3. The following table lists conditions that cause moving of mismatched data in a table between RDAREAs.

Table 12–22: Conditions that cause moving of mismatched data in a table between RDAREAs when the target table is a partitioned table

Operation on table or RDAREA		Conditions that cause moving of mismatched data in a table between RDAREAs
Database reorganization utility (<code>pdorg</code>)	Reorganization (<code>-k rorg</code>)	You perform the following steps in the order listed below on a table partitioned using flexible hash partitioning or a matrix-partitioned table whose second dimension partitioning column is partitioned using flexible hash partitioning: 1. Perform data load for each RDAREA. 2. In a HiRDB single server configuration, execute reorganization for each table. [#] In a HiRDB parallel server configuration, specify the <code>-g</code> option to execute reorganization for each table. [#]
Rebalancing utility (<code>pdrbal</code>)		You add an RDAREA for a table that has mismatched data to execute the rebalancing utility (<code>pdrbal</code>). [#]

#

You cannot execute the utility when you specify `USE` in the `pd_check_pending` operand if the target table is in check pending status.

(2) Other conditions under which loss of data integrity may occur

When all of the following conditions are met, data mismatch may occur; therefore, you need to check data integrity. For details about how to check data integrity, see *12.19.5 Procedure for checking table integrity*. These conditions are the same regardless of whether the referential constraint action is `RESTRICT` or `CASCADE`.

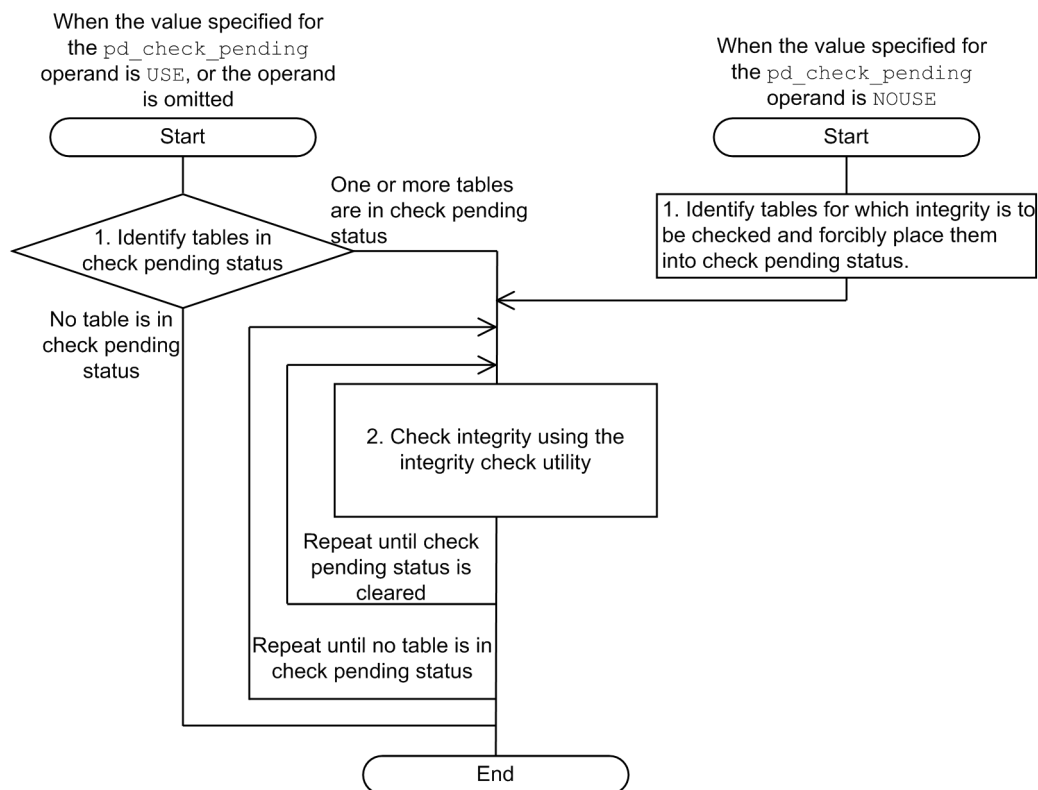
- There are two transactions, one for deleting rows from the referencing table and the other for updating or deleting the referenced table, and these transactions are executed at the same time.
- A value in the primary key column of a row that is to be deleted from the referencing table is the same as a value in a foreign key column of a row that is to be updated or deleted in the referenced table.
- The transaction for updating or deleting rows in the referencing table is committed, and the transaction for deleting rows from the referenced table is rolled back.

When you manipulate referenced tables and referencing tables, make sure that all the above conditions are not true at the same time. You can guarantee data integrity by locking the target table with the `LOCK` statement's shared mode or lock mode. Note that there are some adverse effects on concurrent execution efficiency.

12.19.5 Procedure for checking table integrity

The following figure shows an overview of the procedure for checking data integrity.

Figure 12–38: Overview of procedure for checking data integrity (referential constraint)



When the value specified in the `pd_check_pending` operand is `USE`, or the operand is omitted:

1. Identify tables in check pending status

Search `SQL_TABLES` of the dictionary table to detect the names of tables in check pending status.

```

SELECT TABLE_SCHEMA, TABLE_NAME FROM MASTER.SQL_TABLES
WHERE CHECK_PEND = 'C' OR CHECK_PEND2 = 'C'

```

The owners and names of tables in check pending status are returned in the search result. If no rows are returned in the search result, no tables are in check pending status.

To check whether each generation table is in check pending status when the inner replica facility is used, you can use the condition analysis utility (`pddbst`).

2. Use the integrity check utility to check data integrity.

Use the integrity check utility to check the data integrity of each table and to correct any data that violates constraint conditions. Repeat the procedure until no table remains in check pending status. For details about how to use the integrity check utility to check data integrity, see *12.19.5(1) Procedure for checking data integrity when check pending status is used (referential constraint)*.

When the value specified in the `pd_check_pending` operand is `NOUSE`:

1. Identify the tables for which you want to check data integrity, and forcibly place these tables into check pending status.

To identify tables on which to check data integrity, check the following items:

- Whether a referencing table references a table on which an operation was performed that caused loss of data integrity
- Whether a referential constraint has been defined in the table on which an operation was performed that caused loss of data integrity

The following is an example SQL execution to check these items:

```
SELECT N_PARENTS, N_CHILDREN FROM MASTER.SQL_TABLES
WHERE TABLE_SCHEMA = 'name-of-the-owner-of-the-target-table' AND TABLE_NAME = 'name-of-the-target-table'
```

The following search result is returned:

- The number of foreign keys defined in the target table
- The number of foreign keys that reference the primary key defined in the target table

If `N_PARENTS` is a null value, no referential constraint is defined in the target table.

If `N_CHILDREN` is a null value, no referencing table exists that references the target table as a referenced table.

If the `N_CHILDREN` value is not null, execute the following SQL to check the name of the referencing table that references the target table.

```
SELECT TABLE_SCHEMA, TABLE_NAME, CONSTRAINT_NAME
FROM MASTER.SQL_REFERENTIAL_CONSTRAINTS
WHERE R_OWNER = 'name-of-the-owner-of-the-target-table' AND R_TABLE_NAME = 'name-of-the-target-table'
```

The owners, names, and referential constraint names of referencing tables that reference a target table as a referenced table are returned in the search results. If no rows are returned in the search result, no referencing table that references a target table as a referenced table exists.

When one or more tables are identified, use the integrity check utility to forcibly place the tables into check pending status (the integrity check utility cannot be used to check tables that are not in check pending status).

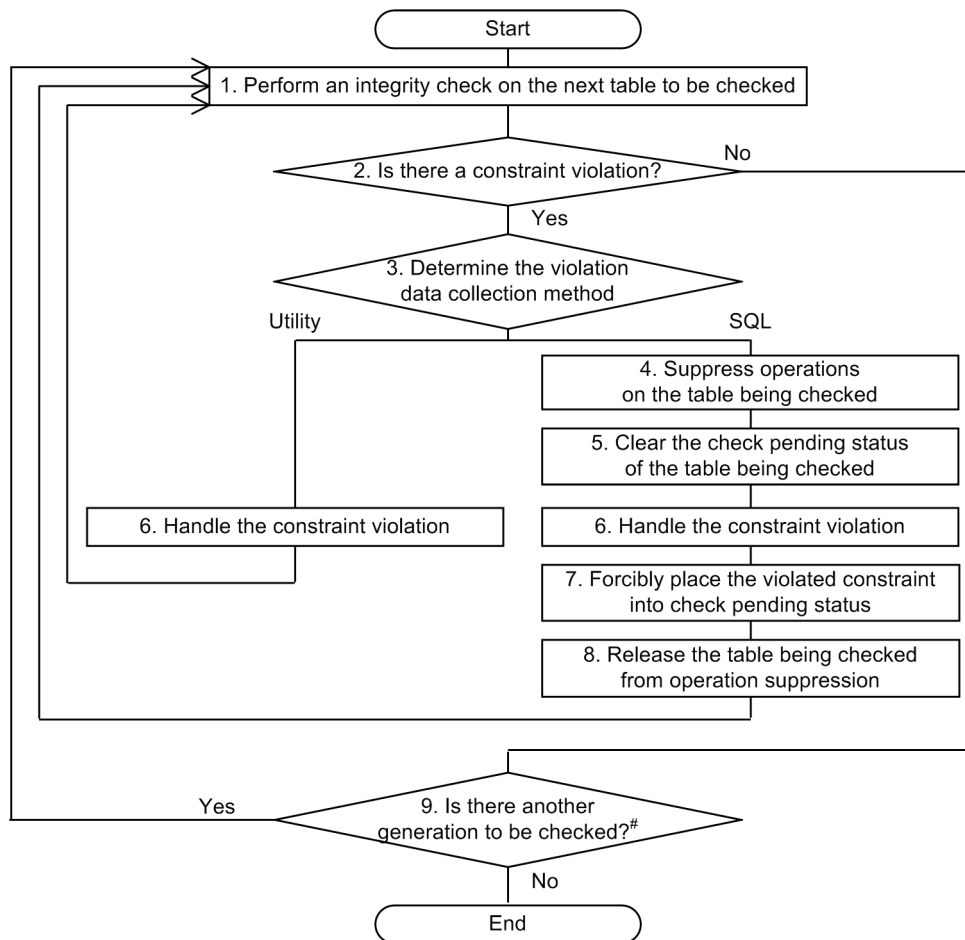
2. Use the integrity check utility to check data integrity.

This step is the same as the step 2 used when the value specified in the `pd_check_pending` operand is `USE`, or the operand is omitted. For details about using the integrity check utility to check data integrity, see *12.19.5(2) Procedure for checking data integrity when check pending status is not used*.

(1) Procedure for checking data integrity when check pending status is used (referential constraint)

The following figure shows the procedure for checking data integrity using the integrity check utility when the value specified for the `pd_check_pending` operand is `USE` or is omitted.

Figure 12–39: Procedure for checking data integrity when check pending status is used (referential constraint)



#: This step is not required in the following cases:

- The inner replica facility is not used.
- The inner replica facility is used and integrity check is performed on all generations.

1. Check the data integrity of the next table to be checked.

Check the data integrity for each table and constraint.

If you use the inner replica facility, specify the generation numbers of the tables to be checked. If you do not use the inner replica facility or if you plan to check the data integrity of all generations, you do not need to specify the generation numbers.

2. Identify constraint violations.

Based on the results of the data integrity check performed in step 1, determine whether any data violates constraint conditions.

3. Correct data that violates constraints.

Decide whether to use the utility or SQL code to correct the violating data. If you choose the utility, proceed to step 6.

4. Stop operations on the table being checked.

Stop performing tasks that use the table for which data integrity cannot be guaranteed.

5. Forcibly cancel the check pending status of the table being checked.

Before taking action to resolve constraint violations, forcibly cancel the check pending status.

6. Take action to resolve constraint violations.

Using the utility:

The following table lists actions. After taking action, return to step 1 to perform data integrity checking, confirm that no violating data remains, and complete the procedure.

Condition	Action
The primary key does not contain the required data	Load correct data using the addition mode of the database load utility (pdload).
The foreign key contains constraint violation data	<ul style="list-style-type: none"> Load correct data using the addition mode of the database load utility (pdload). Use UOC for the database reorganization utility (pdorg) to delete unnecessary data.

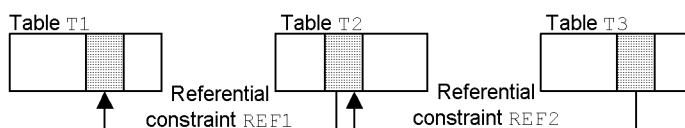
Using SQL code:

The following table lists actions. After taking action, proceed to step 7.

Condition	Action
The primary key does not contain the required data	Use the INSERT statement to insert the required data in the primary key ^{#1} , or use the UPDATE statement to update existing data in the referenced table ^{#2} .
The foreign key contains constraint violation data	Use the DELETE statement to delete the constraint violation data in the foreign key, or use the UPDATE statement to change the data to the correct value ^{#1} .

#1

If a foreign key is also a primary key, and a referencing table has a referenced table for which action is to be taken, you must be careful about the order in which the corrections are performed. For example, assume the following referential relationship exists:



■ Notes when taking action for REF1 constraint violations

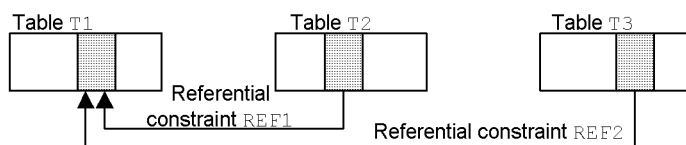
If you use the DELETE statement to correct the data in table T2, if ON DELETE RESTRICT is specified in REF2, first delete the corresponding data in table T3 and then delete the data in table T2. If you use the UPDATE statement to correct the data, if ON UPDATE RESTRICT is specified in REF2, first delete the data in table T3 that corresponds to the pre-update data, and then update the data in table T2.

■ Notes when taking action for REF2 constraint violations

If you use the INSERT statement to correct the data in table T2, check for insertion target data in table T1. If there is no such data, first insert the data into table T1 and then insert the data into table T2. If you use the UPDATE statement to correct the data, check whether post-update data exists in table T1. If there is no such data, first insert the data into table T1 and then update the data in table T2.

#2

About a constraint other than one for which an action is to be taken, if there is a referencing table that references that table as a referenced table, you must be careful about the order of corrections. For example, assume the following referential relationship exists:



■ Notes when taking action for REF1 constraint violations

If you use an UPDATE statement to correct the data in table T1, if ON UPDATE RESTRICT is specified in REF2, first delete the data in table T3 that corresponds to the pre-update data and then update the data in table T2.

7. Forcibly place the violated constraint into check pending status.

Execute the integrity check utility on each constraint, and forcibly place each constraint for which an action was taken into check pending status.

8. Release the stopped operations.

Resume performance of stopped tasks. Return to step 1 to perform data integrity checking and to check for violating data.

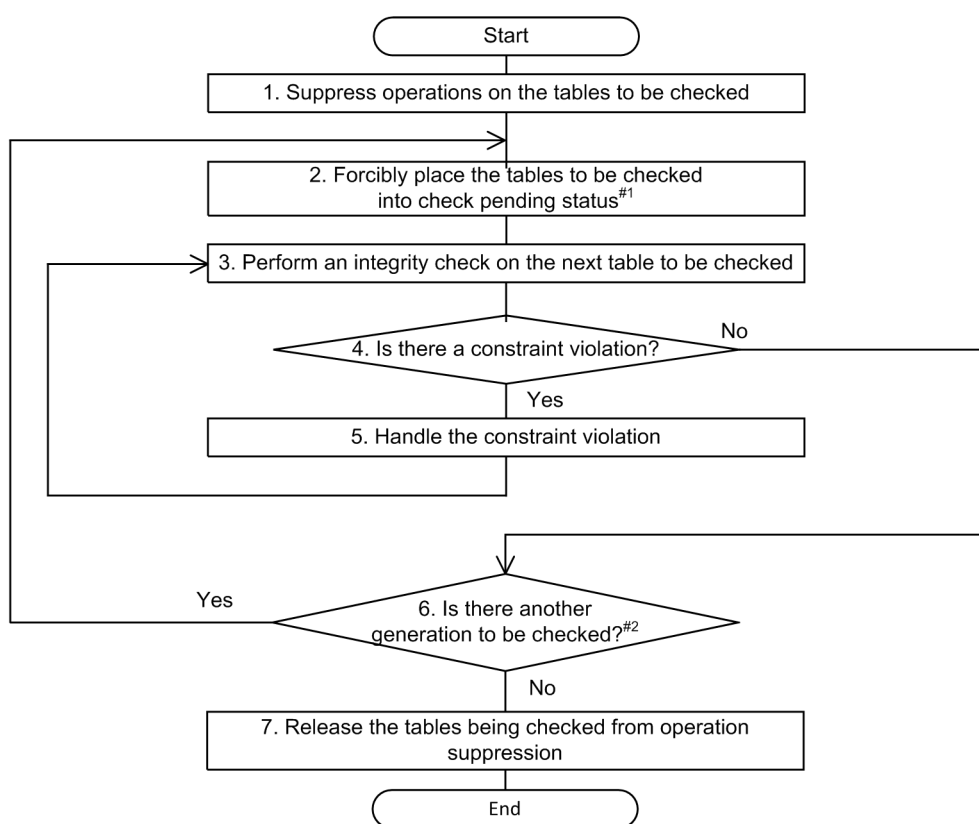
9. Determine if there is another generation to be checked.

When you have created replica RDAREAs of multiple generations or have performed data integrity checking for each generation, return to step 1 to check the data integrity of each generation.

(2) Procedure for checking data integrity when check pending status is not used

The following figure shows the procedure for checking data integrity using the integrity check utility when the value specified for the `pd_check_pending` operand is `NOUSE`.

Figure 12–40: Procedure for checking data integrity when check pending status is not used



#1: This step is not required when a constraint name-basis integrity check is performed.

#2: This step is not required when the inner replica facility is not used. Even when the inner replica facility is used, this step is not required when an integrity check is performed on all generations.

1. Stop operations on the tables to be checked.

Stop performing tasks that use tables for which data integrity cannot be guaranteed.

2. Forcibly place the tables into check pending status.

Forcibly place the tables to be checked into check pending status. If you perform data integrity checking for each constraint in step 3, this step is not necessary.

3. Check the data integrity of the next table to be checked.

Check the data integrity of each table and constraint.

If you use the inner replica facility, specify the generation numbers of the tables to be checked. If you do not use the inner replica facility or if you plan to check the data integrity of all generations, you do not need to specify the generation numbers.

4. Identify constraint violations.

Based on the results of the data integrity check performed in step 3, determine whether any data violates constraint conditions.

5. Correct data that violates constraints.

See step 6 in 12.19.5(1) *Procedure for checking data integrity when check pending status is used (referential constraint)* to correct data that violates constraints.

6. Determine if there is another generation to be checked.

When you have created replica RDAREAs of multiple generations or have performed data integrity checking for each generation, return to step 1 and check the data integrity of each generation.

7. Release the stopped operations.

Resume performance of stopped tasks.

12.19.6 Referential constraints and triggers

(1) Triggers for referential constraint actions

If you specify `CASCADE` for a referential constraint action, HiRDB internally generates a trigger that updates the referencing table for the referenced table. Triggers generated internally by HiRDB become disabled in the following cases. In such a case, you need to re-create the trigger. You may need to create other triggers in addition to those that were generated by HiRDB. Use `ALTER ROUTINE` to re-create all triggers that have been disabled. In addition, if the index definition becomes invalid, or the index information becomes invalid by deletion, re-create the triggers by specifying `ALL` in `ALTER ROUTINE`.

- For update processing
 - The definition of the referencing table was changed.
 - An index was defined for the referencing table.
 - An index of the referencing table was dropped.
 - A trigger whose timing is `UPDATE` was created for the referencing table.
 - For the referencing table, a trigger whose timing is `UPDATE` was deleted.
 - For the table that is referenced by the referencing table, change was made to the table definition of the primary key column.
- For deletion
 - The table definition of the referencing table was changed.
 - An index was defined for the referencing table.
 - An index of the referencing table was dropped.
 - A trigger whose timing is `DELETE` was created for the referencing table.
 - For the referencing table, a trigger whose timing is `DELETE` was deleted.

The triggers internally created by HiRDB are deleted when the referencing table is dropped (by `DROP TABLE` or `DROP SCHEMA`).

(2) Relationship between referential constraints and user-defined triggers

The following explains the order of the operation of triggers, integrity checking for referential constraints, and referential constraint operations (triggers generated internally by HiRDB when a referential constraint is defined) when a trigger and referential constraint are defined for a table, and an update SQL (`INSERT` statement, `UPDATE` statement, or `DELETE` statement) is to be executed. There are two operation order patterns, which depend on the following conditions:

Condition for pattern 1:

The update target is the referenced table and only `RESTRICT` is specified for the referential constraint action, or the update target is the referencing table.

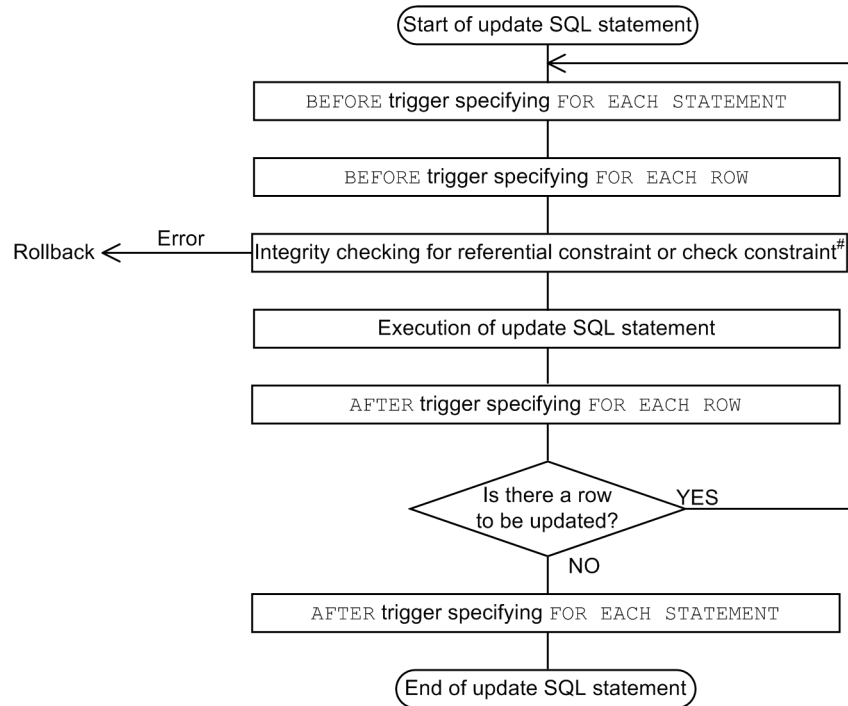
Condition for pattern 2:

The update target is the referenced table and the referential constraint action is not RESTRICT.

If the update target is the referencing table and is also the referenced table, the condition for the referenced table takes precedence.

The order of the actions for each of the patterns is described below.

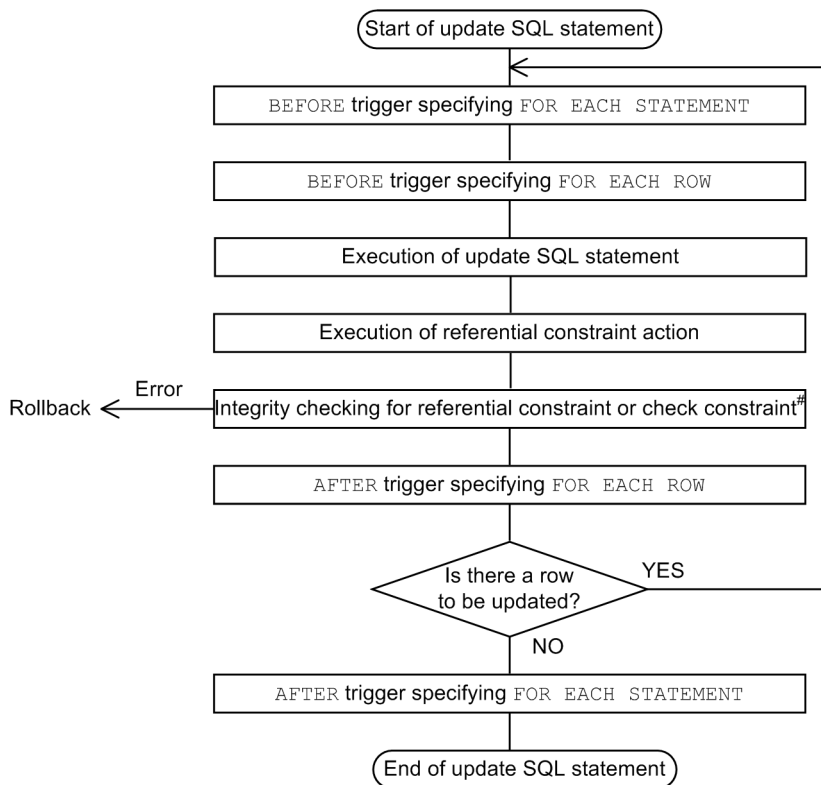
Pattern 1



#: All data integrity checking for the referential constraint takes place at this point. Following are the details of data integrity checking:

1. When the update target is the referencing table
Checking for whether the update (INSERT, UPDATE) data is contained in the referenced table
2. When the update target is the referenced table
Checking for whether the update (UPDATE, DELETE) data is contained in the referencing table
3. When the update target is the referencing table and is also the referenced table
Checking of both 1 and 2 above

Pattern 2



[#]: All data integrity checking for the referential constraint takes place at this point. Details of the integrity checking are the same as for pattern 1.

12.19.7 Notes about linkage with related products

The following notes explain restrictions when linking with related products.

- When the inner replica facility is used
 - When you create an inner replica of an RDAREA in which a referenced or referencing table is stored, use the same generation number for all RDAREAs used to store table data having a referential relationship. If indexes are defined for the referenced or referencing table, use the same generation number for the index storage RDAREA and LOB RDAREA as is used for the RDAREAs that store the tables.
 - If the referencing table in the original RDAREA is in check pending status, do not create an entity of the replica RDAREA. Cancel the check pending status of the referencing table in the original RDAREA and then create an entity for the replica RDAREA.
 - When check pending status is set or canceled for all generations, the generations in command hold and in closed status are handled as not having an entity of the replica RDAREA. Therefore, these areas are excluded as targets for setting or canceling check pending status. If an RDAREA is excluded as a target for setting or canceling check pending status even though it has an entity, first cancel the hold status of the RDAREA and then use the integrity check utility to update the table information in the RDAREA.
 - After executing the following operations, use the integrity check utility, specifying all generations to execute data integrity checking for each table.
 - PURGE TABLE statement
 - Re-initialize RDAREA
 - Delete replica RDAREA
 - Integrate inner replica group
- When performing updatable online reorganization

Data integrity is not guaranteed when updatable online reorganization and database updating are performed in batch mode. This means that, if you have set USE in the `pd_check_pending` operand, the referencing table

might be in check pending status. In this case, use the integrity check utility to cancel check pending status. If `NOUSE` is specified in the `pd_check_pending` operand, use the integrity check utility to forcibly place the table into check pending status and then check data integrity. For details about how to check data integrity, see *12.19.5 Procedure for checking table integrity*.

- Using HiRDB Datareplicator
Make sure that no referential constraint has been defined for the target table.
- Changing partitioning storage conditions
If you change the partition storage conditions for the referenced table or integrate or partition RDAREAs in such a manner that existing data is deleted, data integrity is not guaranteed after the partition storage conditions have been changed; in such a case, the user must check data integrity. For details about how to check data integrity, see *12.19.5 Procedure for checking table integrity*.

12.20 Check constraints

12.20.1 About check constraints

In many cases, there are restrictions on table data in a database, such as with respect to value ranges and conditions. For example, when product information is stored in a database, a price cannot be a negative value. This means that no negative value can exist in such a database and values should be checked for this constraint when data is inserted or updated. The purpose of *check constraints* is to maintain data integrity in the table by checking constraint conditions during data insertion or updating and suppressing the operation if checked data does not satisfy conditions. In this manual, a table for which a check constraint has been defined is called a *check constraint table*.

Execution of a utility or other operation may cause loss of guaranteed data integrity. In such a case, the check constraint table is placed in check pending status. For details about check pending status, see *12.20.3 Check pending status*; for details about operations that cause the loss of guaranteed data integrity, see *12.20.4 Data manipulation and integrity*.

Effects of check constraints

When check constraints are defined, the workload of UAP creation is reduced because checking can be automated during data insertion or updating. However, when a check constraint table is updated, the processing time required for checking increases because data integrity is checked.

12.20.2 Defining check constraints

You can define a check constraint by specifying `CHECK` in the `CREATE TABLE` definition SQL statement and the constraint condition for table values as a search condition. Also, to use the check pending status, specify `USE` in the `pd_check_pending` operand, or do not specify (omit) the operand.

(1) Limitations on tables for which check constraints are defined

This subsection describes limitations that apply to the definition of tables for which check constraints are defined and to modification of the definitions of such tables.

(a) During table definition (`CREATE TABLE`)

- Check constraints cannot be defined for a falsification prevented table.
- You can define a maximum of 254 check constraints per table; you must be careful not to define more than 254 check constraints. The following shows an example of a table definition that is not valid:

```
CREATE TABLE T1(C001 INT CONSTRAINT CHECK_T1_C001 CHECK(C001>0),
                C002 INT CONSTRAINT CHECK_T1_C002 CHECK(C002>0),
                :
                C254 INT CONSTRAINT CHECK_T1_C254 CHECK(C254>0))
                C255 INT CONSTRAINT CHECK_T1_C255 CHECK(C255>0))
```

} 255 check constraints

This definition is invalid because there are more than 254 check constraints. This example would result in an error during table definition.

- For each table, you can define a maximum of 254 check constraints separated by `ANDs` and `ORs`, including the `ANDs` and `ORs` of search conditions in the individual check constraints (this number does not include `ANDs` and `ORs` for search conditions in `CASE` expressions and in those search conditions). The following shows an example of a table definition that is not valid:

```
CREATE TABLE T1(C001 INT CONSTRAINT CHECK_T1_C1 CHECK(C001=0 OR C001=1 OR ~C001=200),
                C002 INT CONSTRAINT CHECK_T1_C2 CHECK(C002=0 OR C002=1 OR ~C002=53))
```

Number of `ANDs` and `ORs`: 200

Number of `ANDs` and `ORs`: 53

This example contains two check constraints, plus there are 200 ANDs in the search conditions in the constraint named `CHECK_T1_C1` and 53 ANDs in the search conditions in the constraint named `CHECK_T1_C2`. The sum of the number of check constraints and the number of ANDs and ORs in the search conditions in the check constraints is 255 ($2 + 200 + 53$), which is greater than 254. Therefore, this definition is invalid and would result in an error during table definition.

The sum of the number of check constraints defined for the table and the number of ANDs and ORs in the search conditions in each check constraint is stored in the `N_CHECK_LIMIT` column of the `SQL_TABLE` data dictionary table.

(b) During table modification (ALTER TABLE)

- You cannot use the `DROP` and `RENAME` clauses in modifying the table definition of a check constraint table.
- You cannot use the `CHANGE` clause to modify a constraint table in the following ways:
 - Changing the data type and data length
 - Changing `SPLIT`
 - Setting and releasing the default value
 - Setting `WITH DEFAULT`
- The `RENAME` clause cannot be used to rename columns of a check constraint table.

(2) Notes when defining a check constraint

- Estimating the size of the SQL object buffer length
When you perform operations on a check constraint table, HiRDB generates triggers to check constraint conditions. Therefore, you must take into account the SQL objects of the constraint conditions generated by HiRDB when you specify the SQL object buffer. For details about how to estimate the SQL object buffer length (`pd_sql_object_cache_size`), see the manual *HiRDB Version 9 System Definition*.
- Backing up data
The extent of data that is backed up differs depending on the check pending status at backup time. For details about the backup time and extent, see *RDAREAs to be backed up together* in the *HiRDB Version 9 System Operation Guide*.
- Reorganizing data dictionary RDAREAs
When you repeat definition and deletion of check constraint tables, storage efficiency of the data dictionary RDAREA decreases. In such a case, use the database condition analysis utility (`pddbst`) to check the storage efficiency of the data dictionary RDAREA and reorganize the area as necessary.

12.20.3 Check pending status

If data integrity can no longer be guaranteed due to execution of a utility or some other operation, HiRDB restricts data manipulation in the check constraint table. The status in which data manipulation is restricted due to loss of guaranteed data integrity is called *check pending status*. To place a check constraint table in check pending status for the purpose of restricting data manipulation, you must either specify `USE` in the `pd_check_pending` operand or do not specify (omit) the operand. You can use the integrity check utility (`pdconstck`) to clear the check pending status of a table. You can also use the integrity check utility to forcibly place a table into check pending status.

If you have specified `NOUSE` in the `pd_check_pending` operand, data manipulation is not restricted even when data integrity between tables cannot be guaranteed. In this case, if you execute an SQL statement or a utility that nullifies the guarantee of data integrity, you can use the integrity check facility to forcibly place the table into check pending status, and then check data integrity.

For details about operations that cause loss of guaranteed data integrity, see *12.20.4 Data manipulation and integrity*. For details of how to check data integrity, see *12.20.5 Procedure for checking table integrity*.

(1) Managing check pending status

Check pending status is managed based on dictionary tables and on the table information of the RDAREAs in which the tables are stored. In dictionary tables, check pending status is managed for each table and constraint. In table

information, check pending status is managed for each RDAREA if the table is a partitioned table and for each table if the table is not a partitioned table.

The following table lists and describes the storage locations of check pending status information.

Table 12–23: Storage locations of check pending status information and their contents (check constraint)

Storage location			Stored information
Dictionary table	SQL_TABLES table	CHECK_PEND2 column	Check pending status of check constraint for each table
	SQL_CHECKS table	CHECK_PEND2 column	Check pending status of check constraint for each constraint
RDAREA table information		For unpartitioned table	Check pending status of check constraint or check constraint for each table
		For partitioned table	Check pending status of referential constraint or check constraint for each RDAREA

(2) Operations that are restricted for tables in check pending status

These restrictions are the same as those for the referential constraint. See *12.19.3(3) Operations that are restricted for tables in check pending status*.

(3) When a partitioned table or the inner replica facility is used

These restrictions are the same as those for the referential constraint. See *12.19.3(5) When a partitioned table or the inner replica facility is used*. However, replace the term *referencing table* with *check constraint table*.

(4) Notes on using check pending status

- If you change the value specified in the `pd_check_pending` operand from `NOUSE` to `USE`, you must use the integrity check utility to check the data integrity of the check constraint table. For details about how to check data integrity, see *12.20.5 Procedure for checking table integrity*.
- If you specify `USE` in the `pd_check_pending` operand, referencing tables and RDAREAs placed in check pending status are locked, and locked resources when a utility or an SQL statement is executed are different from those when check pending status is not used.

12.20.4 Data manipulation and integrity

When a check constraint table is updated, added to, or deleted by a data manipulation SQL statement, HiRDB performs checking during execution to guarantee data integrity. However, if the table is manipulated by the utilities listed in the following table, data integrity may not be guaranteed because HiRDB does not perform integrity checking. If you specify `USE` in the `pd_check_pending` operand and perform these operations, the check constraint table is placed in check pending status.

Table 12–24: Operations on check constraint tables that nullify the guarantee of data integrity and the conditions under which loss of data integrity occurs

Operation on table or RDAREA		Condition for loss of data integrity
Database load utility (<code>pdload</code>)	Data reload	Data that does not satisfy search conditions specified in the check constraint definition is loaded.
Database reorganization utility (<code>pdreorg</code>)	Reload (<code>-k reload</code>)	Data that does not satisfy search conditions specified in the check constraint definition is reloaded.
Updatable online reorganization reflection processing (<code>pdorend</code>)		During operation of updatable online reorganization [#] , in a replica RDAREA in the current database, you used the database load utility (<code>pdload</code>) to perform an operation on a check constraint

Operation on table or RDAREA	Condition for loss of data integrity
	table of the replica RDAREA, which caused a loss of data integrity (loss of data integrity occurs after reflection processing).

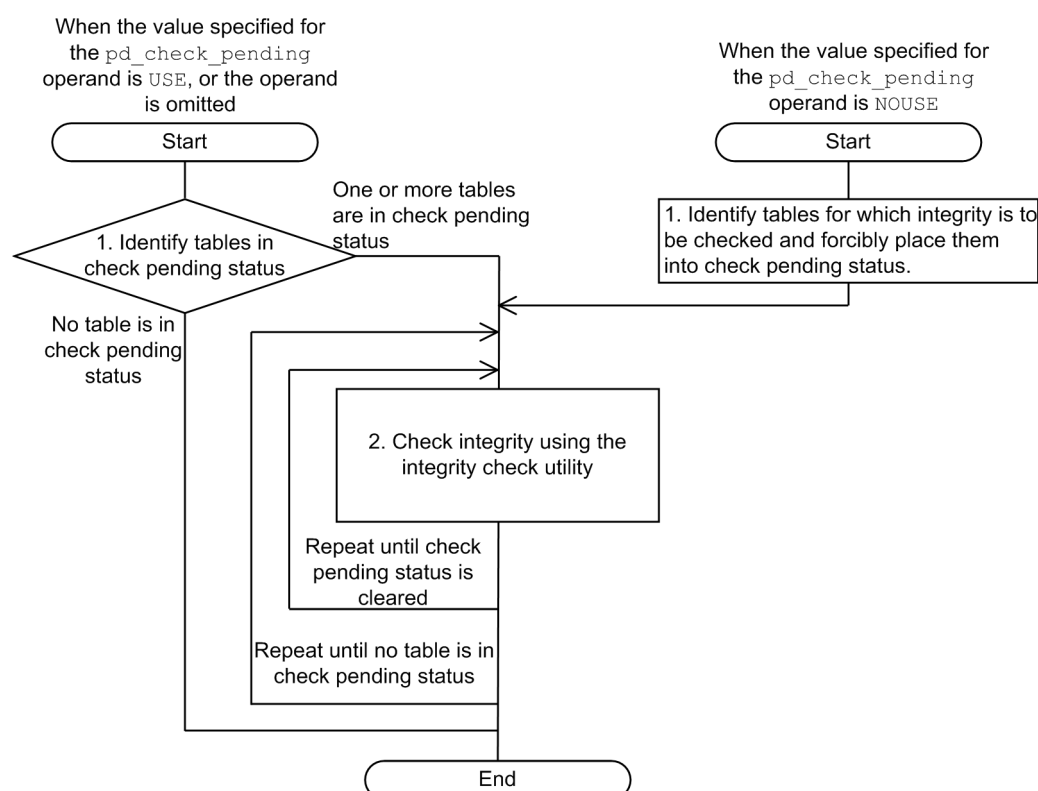
#

For details about operating updatable online reorganization, see the *HiRDB Version 9 Staticizer Option Description and User's Guide*.

12.20.5 Procedure for checking table integrity

The following figure shows an overview of the procedure for checking data integrity.

Figure 12–41: Overview of the procedure for checking data integrity (check constraint)



When the value specified in the `pd_check_pending` operand is `USE`, or the operand is omitted:

1. Identify tables in check pending status.

Search `SQL_TABLES` of the dictionary table to detect the names of tables in check pending status.

```
SELECT TABLE_SCHEMA, TABLE_NAME FROM MASTER.SQL_TABLES
WHERE CHECK_PEND = 'C' OR CHECK_PEND2 = 'C'
```

The owners and names of tables in check pending status are returned in the search result. If no rows are returned in the search result, no tables are in check pending status.

To check whether each generation table is in check pending status when the inner replica facility is used, you can use the condition analysis utility (`pddbst`).

2. Use the integrity check utility to check data integrity.

Use the integrity check utility to check the data integrity of each table and to correct any data that violates constraint conditions. Repeat the procedure until no table remains in check pending status. For details about how to use the integrity check utility to check data integrity, see *12.20.5(1) Procedure for checking data integrity when check pending status is used (check constraint)*.

When the value specified in the `pd_check_pending` operand is `NOUSE`:

1. Identify the tables for which you want to check data integrity, and forcibly place these tables into check pending status.

Check whether a check constraint is defined for a table on which an operation that causes loss of guaranteed data integrity was performed. The following shows an example of SQL code for checking this.

```
SELECT N_CHECK FROM MASTER.SQL TABLES
WHERE TABLE_SCHEMA = 'name-of-the-owner-of-the-target-table' AND TABLE_NAME = 'name-of-
the-target-table'
```

The following search result is returned:

- The number of check constraint definitions

When `N_CHECK` is a null value, no check constraint is defined for the target table.

After identifying the tables, use the integrity check utility to forcibly place the tables into check pending status (you cannot use the integrity check utility to check tables that are not in check pending status).

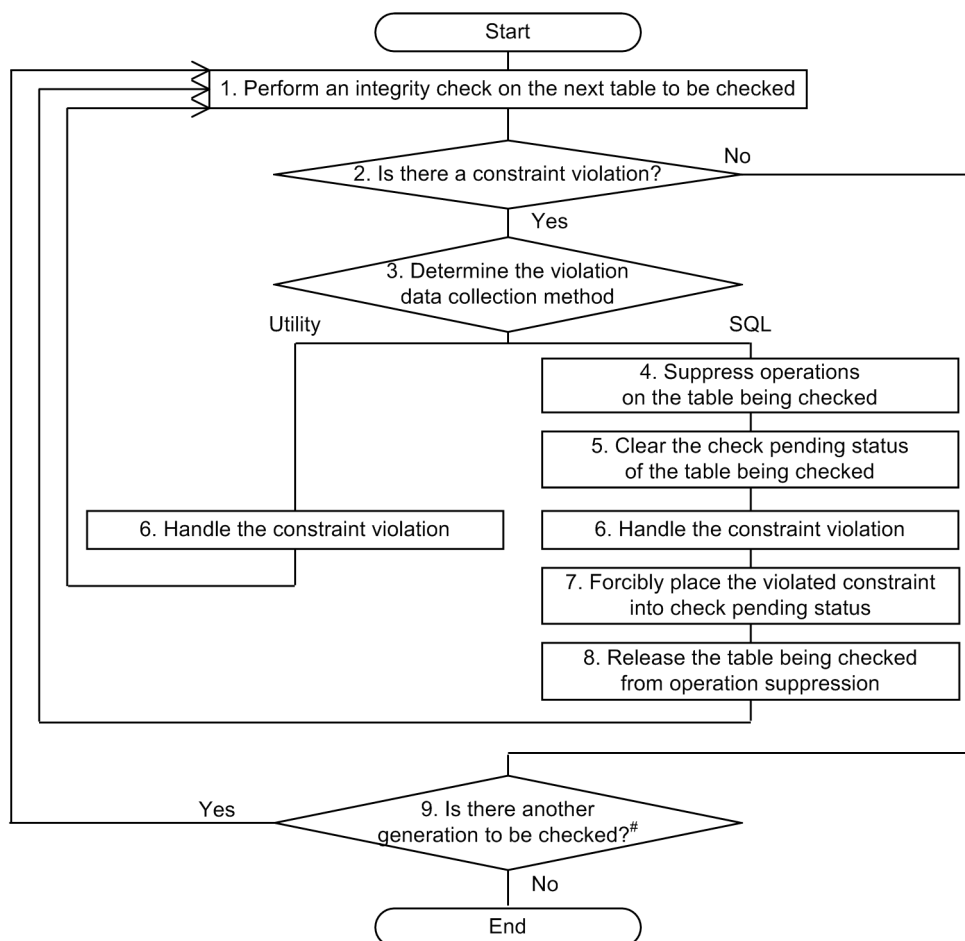
2. Use the integrity check utility to check integrity.

This step is the same as the step 2 used when the value specified in the `pd_check_pending` operand is `USE`, or the operand is omitted. The procedure for checking data integrity is the same as that used for a referential constraint; for details, see *12.19.5(2) Procedure for checking data integrity when check pending status is not used*.

(1) Procedure for checking data integrity when check pending status is used (check constraint)

The following figure shows the procedure for checking integrity using the integrity check utility when the value specified for the `pd_check_pending` operand is `USE` or is omitted.

Figure 12–42: Procedure for checking data integrity when check pending status is used (check constraint)



#: This step is not required in the following cases:

- The inner replica facility is not used.
- The inner replica facility is used and integrity check is performed on all generations.

1. Check the data integrity of the tables to be checked.

Check the data integrity for each table and constraint.

If you use the inner replica facility, specify the generation numbers of the tables to be checked. If you do not use the inner replica facility or if you plan to check the data integrity of all generations, you do not need to specify the generation numbers.

2. Identify constraint violations.

Based on the results of the data integrity check performed in step 1, determine whether any data violates constraint conditions.

3. Correct data that violates constraints.

Decide whether to use the utility or SQL code to correct the violating data. If you choose the utility, proceed to step 6.

4. Stop operations on the tables to be checked.

Stop performing tasks that use tables for which data integrity cannot be guaranteed.

5. Forcibly cancel the check pending status of the tables to be checked.

Before taking action to resolve constraint violations, forcibly cancel the check pending status.

6. Take action to resolve constraint violations.

Using the utility:

The following table lists actions. After taking action, return to step 1 to perform data integrity checking, confirm that no violating data remains, and complete the procedure.

Condition	Action
When correcting search conditions specified in the check constraint	To correct search conditions: <ol style="list-style-type: none"> 1. Unload all data in the table. 2. Use <code>DROP TABLE</code> to delete the table definition. 3. Use <code>CREATE TABLE</code> to redefine the table. At this time, specify the correct check constraint search conditions. 4. Load the data that was unloaded in step 1.
When there is constraint violation data in the table	<ul style="list-style-type: none"> • Use the database load utility (<code>pdload</code>) to load data in creation mode. • Use UOC for the database reorganization utility (<code>pdorg</code>) to delete unnecessary data.

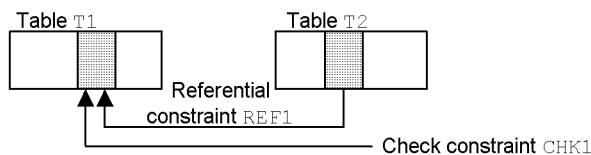
Using SQL code:

The following table lists actions. After taking action, proceed to step 7.

Condition	Action
When correcting search conditions specified in the check constraint	Same as the action when the utility is used.
When there is constraint violation data in the table	Use the <code>DELETE</code> statement to delete the constraint violation data, or use the <code>UPDATE</code> statement to update it to the correct value.#

#

If a referencing table references the table for which an action is to be taken, as a referenced table, you must follow a specific order of corrections. For example, assume the following referential relationship exists:

**■ Notes when taking action for CHK1 constraint violations**

If you use the `DELETE` statement to correct the data in table T1, if `ON DELETE RESTRICT` is specified in REF1, first delete the corresponding data in table T2 and then delete the data in table T1. If you use the `UPDATE` statement to correct the data, if `ON UPDATE RESTRICT` is specified in REF1, first delete the data in table T2 that corresponds to the pre-update data and then update the data in T1.

7. Forcibly place the violated constraint into check pending status.

Execute the integrity check utility on each constraint, and forcibly place each constraint for which an action was taken into check pending status.

8. Release the stopped operations.

Resume performance of stopped jobs. Return to step 1 to perform data integrity checking and to check for violating data.

9. Check for the existence of more generations to be checked.

When you have created replica RDAREAs of multiple generations or have performed data integrity checking for each generation, return to step 1 to check the data integrity of each generation.

12.20.6 Notes about linkage with related products

- When the inner replica facility is used
 - If the referencing table in the original RDAREA is in check pending status, do not create an entity of the replica RDAREA. Cancel the check pending status of the referencing table in the original RDAREA and then create an entity for the replica RDAREA.
 - When check pending status is set or canceled for all generations, the generations in command hold and in closed status are handled as not having an entity of the replica RDAREA. Therefore, these areas are excluded as targets for setting or canceling check pending status. If an RDAREA is excluded as a target for setting or

canceling check pending status even though it has an entity, first cancel the hold status of the RDAREA and then use the integrity check utility to update the table information in the RDAREA.

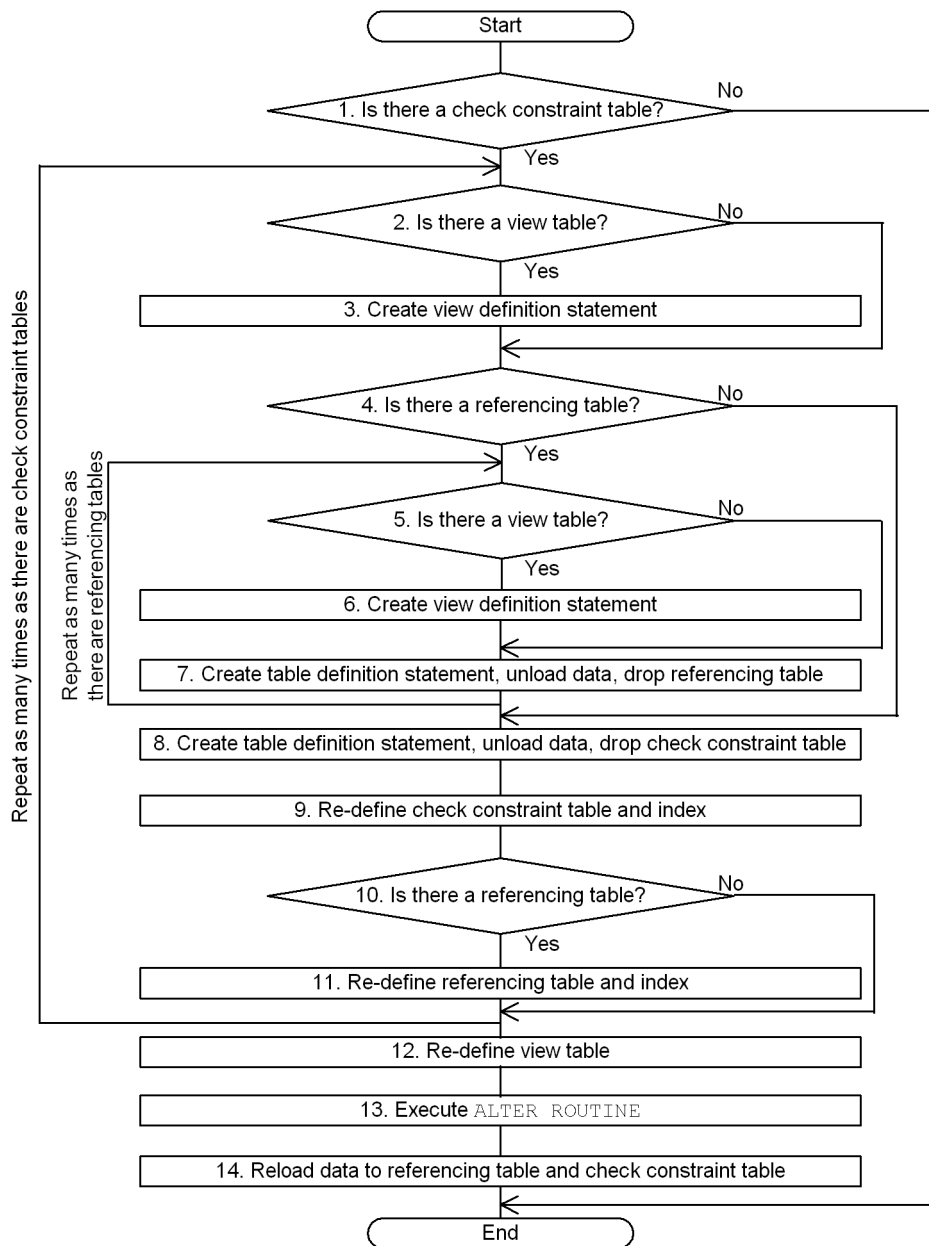
- After executing the following operations, use the integrity check utility, specifying all generations to execute data integrity checking for each table.
 - PURGE TABLE statement
 - Re-initialize RDAREA
 - Delete replica RDAREA
 - Integrate inner replica group
- When performing updatable online reorganization
 - Data integrity is not guaranteed when updatable online reorganization and database updating are performed in batch mode. This means that, if you have set `USE` in the `pd_check_pending` operand, the check constraint table might be in check pending status. In this case, use the integrity check utility to cancel check pending status. If `NOUSE` is specified in the `pd_check_pending` operand, use the integrity check utility to forcibly place the table into check pending status and then check data integrity. For details about how to check data integrity, see *12.20.5 Procedure for checking table integrity*.
- Using HiRDB Datareplicator

When you use HiRDB Datareplicator, there is no need to define check constraints for a target table because only conforming data is applied.

12.20.7 Migrating check constraint tables to 64-bit mode

When you have migrated HiRDB from 32-bit mode to 64-bit mode, an attempt to insert or update data in a check constraint table that was defined in the 32-bit mode will result in an error. To enable insertion and updating of data in such a table in the 64-bit mode, you must restart HiRDB in the 64-bit mode and then re-define the check constraint table. The following figure shows the basic procedure for migrating a check constraint table to 64-bit mode.

Figure 12–43: Basic procedure for migrating a check constraint table to 64-bit mode



To migrate a check constraint table to 64-bit mode:

1. Check for any check constraint tables.

To determine whether there are any check constraint tables, execute the following SQL statement:

```
SELECT TABLE_SCHEMA, TABLE_NAME FROM MASTER.SQL_TABLES WHERE N_CHECK > 0
```

If the number of resulting rows is 1 or greater, there is a check constraint table. In the search results, `TABLE_SCHEMA` indicates the owner of each check constraint table and `TABLE_NAME` indicates the name of each check constraint table.

2. Check for a view table.

If a check constraint table is dropped, the view tables that used the check constraint table are also dropped. Therefore, you must check for any view tables that used a check constraint table. To check for any view tables that used a check constraint table, execute the following SQL statement:

```
SELECT VIEW_SCHEMA, VIEW_NAME FROM MASTER.SQL_VIEW_TABLE_USAGE
WHERE BASE_OWNER=owner-of-check-constraint-table AND TABLE_NAME=name-of-
check-constraint table
```

If the number of resulting rows is 1 or greater, there is a view table that used the check constraint table. In the search results, `VIEW_SCHEMA` indicates the owner of a view table and `VIEW_NAME` indicates the name of a view table.

3. Create a view definition statement.

Use the `pddefrev` command (create a definition SQL statement) to create a view definition statement.

4. Check for a referencing table.

If a primary key has been defined for a check constraint table and a referencing table that references that primary key has been defined, that check constraint table cannot be dropped. The referencing table referencing that primary key must be dropped. To check for any referencing table that references the primary key of a check constraint table, execute the following SQL statement:

```
SELECT CONSTRAINT_SCHEMA, TABLE_NAME
FROM MASTER.SQL_REFERENTIAL_CONSTRAINTS
WHERE R_OWNER= owner-of-check-constraint-table AND R_TABLE_NAME=name-
of-check-constraint-table
```

If the number of resulting rows is 1 or greater, there is an applicable referencing table. In the search results, `CONSTRAINT_SCHEMA` indicates the owner of a referencing table and `TABLE_NAME` indicates the name of a referencing table.

5. Check for a view table.

If the referencing table is dropped, any view tables that used the referencing table are also dropped. Therefore, you must check for any view tables that used the referencing table. To check for any view table that used a referencing table, execute the following SQL statement:

```
SELECT VIEW_SCHEMA, VIEW_NAME FROM MASTER.SQL_VIEW_TABLE_USAGE
WHERE BASE_OWNER=owner-of-referencing-table AND TABLE_NAME=name-of-
referencing-table
```

If the number of resulting rows is 1 or greater, there is a view table that used the referencing table. In the search results, `VIEW_SCHEMA` indicates the owner of a view table and `VIEW_NAME` indicates the name of a view table.

6. Create a view definition statement.

Use the `pddefrev` command (create a definition SQL statement) to create a view definition statement using the referencing table that references the check constraint table.

7. Create a table definition statement, unload data, and drop the referencing table.

Use the `pddefrev` command (create a definition SQL statement) to create a table definition statement for the referencing table. After the table definition statement has been created, unload data from the referencing table that is to be dropped, then drop the referencing table.

8. Create a table definition statement, unload data, and drop the check constraint table.

Use the `pddefrev` command (create a definition SQL statement) to create a table definition statement for the check constraint table. After the table definition statement has been created, unload data from the check constraint table that is to be dropped, then drop the check constraint table.

9. Re-define the check constraint table and index.

Use the table definition statement created in step 8 to re-define the check constraint table and index.

10. Check for a referencing table.

In the same manner as in step 4, check for any referencing table that references the check constraint table re-defined in step 9.

11. Re-define the referencing table and index.

If there is a referencing table that references the check constraint table re-defined in step 9, re-define the referencing table and index using the table definition statement created in step 7.

12. Re-define the view table.

If there is a view table that used the check constraint table or that used the referencing table, re-define the view table using the view table definition statement created in steps 3 and 6.

13. Execute `ALTER ROUTINE`.

Execute the `ALTER ROUTINE` definition SQL statement because the function may have been disabled due to dropping of tables and view tables.

14. Reload data into the referencing table and check constraint table.

Reload data to the re-defined tables.

12.21 Compressed tables

12.21.1 Data compression facility

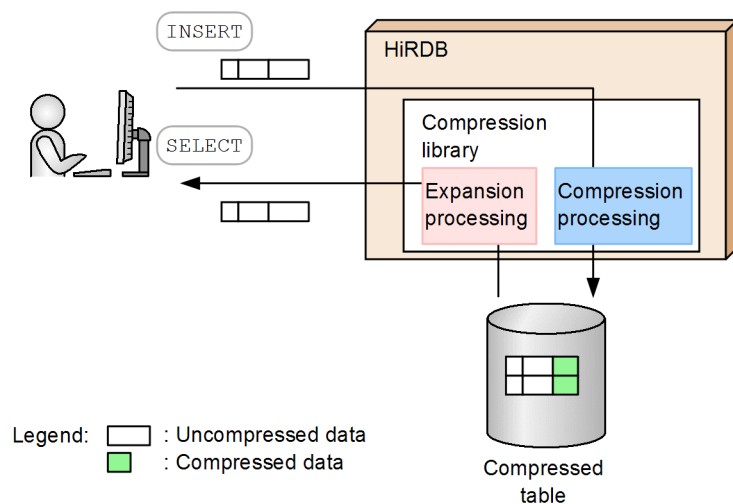
You can compress the data that HiRDB stores in a table. This is called the *data compression facility*. Data compression is specified for individual columns. A column in which compressed data is stored is called a *compressed column*, and a table containing a compressed column is called a *compressed table*.

Compressing data provides the following advantages:

- The database size is reduced.
- There is no need for UAPs to perform data compression processing.

The following figure provides an overview of data compression.

Figure 12–44: Overview of data compression



Explanation:

There is no need for the user to provide instructions for data compression and expansion because HiRDB performs this processing.

(1) Criteria

We recommend that you compress a table that contains large variable-length binary data, such as images and audio data. Because there is overhead for compression and expansion processing that is associated with compressed tables, you should use compressed tables in a system that values storage efficiency over performance.

(2) Guidelines for data compression efficiency

Compression efficiency is a representation of how much storage space can be saved after compression versus before compression. Use the following formula to determine compression efficiency:

Compression efficiency (%) =

$$\{ \text{pre-compression data length} - \text{post-compression data length} \} \div \text{pre-compression data length} \times 100$$

The relationship between the data compression rate and the compression efficiency is as follows:

$$\text{compression rate} + \text{compression efficiency} = 100$$

For details about how to measure the compression rate, see *12.21.7 How to measure the data compression rate*.

The table below provides guidelines for evaluating data compression efficiency. Note that the compression efficiency values shown in the table are only guidelines. The actual data compression rate and compression efficiency depend on the specific data to be compressed.

Table 12–25: Guidelines for compression efficiency

Data type	Compression efficiency (%)
BINARY data consisting of the same characters	98.51
Completely random BINARY data	-0.36 [#]
Text data (.txt)	58.50
Image data (.bmp)	75.42
Audio data (.wav)	9.46

#

This compression efficiency is a negative value because a header area is added during compression processing. For details about the compression processing, see *12.21.2 How data is compressed*.

(3) Files that are output by HiRDB when compressed tables are manipulated

The following table shows the data status in files that are output by HiRDB when compressed tables are manipulated.

Table 12–26: Data status in files that are output by HiRDB

Processing	File	Data status
Execution of SQL statements that require a work table file	Work table file	Expanded data
Database update processing	System log file	Compressed data
System log unload processing	Unload log file	
Database backup processing (pdcopy command)	Backup file	
Table reorganization processing (pdrorg -k rorg command)	Unload data file	Compressed data [#]
Table unload processing (pdrorg -k unld command)		Expanded data

#: If reorganization is performed by using UOC, the expanded data is stored in a table.

(4) Error handling

If an error occurs in an RDAREA containing the data and indexes of a compressed table, you can recover the RDAREA by using the database recovery utility (pdrstr) in the same manner as for normal database recovery.

12.21.2 How data is compressed

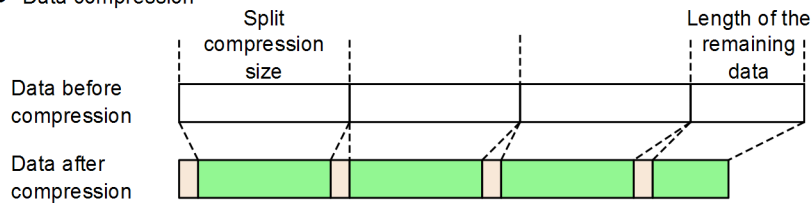
The compression library used during data compression is `zlib`. HiRDB uses `zlib` to compress data into segments of the *split compression size* specified when the table is defined (default: MIN (32,000 bytes, *definition length for the compressed column*)). For purposes of managing information about the data before and after compression, HiRDB adds a header area (8 bytes) to each segment of the split compression size (this is separate from the header area added to the compressed data by `zlib`).

If the data length is the same before and after compression, or the data length is greater after compression than before compression, HiRDB stores the data without compressing it. Because of the header areas that are added, the data size

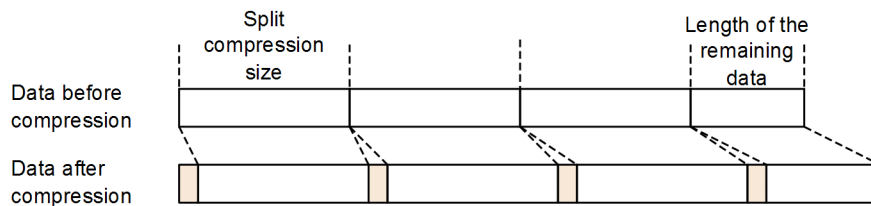
after compression might be greater than the data size before compression. The following figure shows data before and after compression.

Figure 12–45: Data before and after compression

- Data compression



- When the data length is the same before and after compression, or when the data length is greater after compression than before, the data is not compressed



Legend: : Header area for managing information about data before and after compression
 : Uncompressed data
 : Compressed data

12.21.3 How to define a compressed table

To define a compressed table, use the compression specification (`COMPRESSED`) in the column definition in the `CREATE TABLE` definition SQL statement. If necessary, also specify a split compression size. Note that the following conditions apply to the compression specification:

- A compression specification can only be specified for columns (it cannot be specified for tables).
- A compression specification can only be specified for columns of the following data types:
 - `BINARY` type whose definition length is 256 bytes or greater
 - Abstract data type (XML type)[#]

#

To compress data in a column with the abstract data type (XML type), HiRDB XML Extension version 09-03 or later is required.

12.21.4 How to convert an existing table to a compressed table

(1) Converting columns in an existing table to compressed columns

The column attribute change definition (`CHANGE column-name`) of the `ALTER TABLE` definition SQL statement cannot be used to change a column to a compressed column. Instead, you must use the procedure below to convert columns in an existing table to compressed columns.

To convert columns to compressed columns:

1. Unload the existing table.
Unload the existing table.
2. Delete the existing table.

Use `DROP TABLE` to delete the existing table.

3. Redefine the table.

Use `CREATE TABLE` to redefine the table with the compression specification made for each column that is to be changed to a compressed column. Do not make any changes other than to add the compression specification.

4. Reload the table.

Reload the unload data file that was unloaded in step 1 to the table that was redefined in step 3.

For details about unloading and reloading data, see the manual *HiRDB Version 9 Command Reference*.

Reference note

When the column to be changed to a compressed column is the last column of a table, steps 2 and 3 above can be replaced with the following step:

- Delete the existing column and add a compressed column
Use `PURGE TABLE` to remove all data from the table and then the column deletion definition (`DROP column-name`) of `ALTER TABLE` to delete the column that is to be changed to a compressed column. Next, use the column addition definition (`ADD column-name`) of `ALTER TABLE` to redefine (add) the table using the compression specification for the column that was deleted.
-

(2) Adding a compressed column at the end of an existing table

Use the column addition definition (`ADD column-name`) of the `ALTER TABLE` definition SQL statement to add a compressed column with the compression specification. Then load data into the compressed column. The data is compressed and stored in the column.

Reference note

The column addition definition of `ALTER TABLE` adds a column at the end of a table. Therefore, a compressed column can be added only at the end of a table.

12.21.5 How to change the definition of a compressed column (removing the compression specification for a column)

The column attribute change definition (`CHANGE column-name`) of the `ALTER TABLE` definition SQL statement cannot be used to change the definition of a compressed column. To change the definition of a compressed column, such as removing the compression specification or changing the split compression size, you must use the procedure below.

To change the definition of a compressed column:

1. Unload the compressed table.

Unload the compressed table whose definition is to be changed.

2. Redefine the table.

Use one of the following methods to redefine a table whose compression specification has been changed or removed:

- Redefining the table
Use `DROP TABLE` to delete the compressed table and then use `CREATE TABLE` to redefine the table whose compression specification was changed or removed.
- Deleting and adding columns
Use `PURGE TABLE` to remove all data from the table and then use the column deletion definition (`DROP column-name`) of `ALTER TABLE` to delete the compressed column. Next, use the column addition definition (`ADD column-name`) of `ALTER TABLE` to add the column whose compression specification was changed or removed.
Note that the column addition definition of `ALTER TABLE` adds a column at the end of a table. Therefore, the compression specification can be changed or removed only for the last column.

3. Reload the table

Reload the unload data file that was unloaded in step 1 to the table that was defined in step 2.

For details about unloading and reloading data, see the manual *HiRDB Version 9 Command Reference*.

12.21.6 Notes about using compressed tables

- When data in compressed columns is manipulated by SQL statements and utilities, overhead is required for compression and expansion processing. You can check the statistical information about a UAP provided by the statistics analysis utility (`pdstedit`) to determine the time required for compressing and expanding `BINARY` type data. Note that the processing time cannot be checked for an abstract data type (XML type).
- The data compression efficiency becomes higher as the split compression size is increased, although this also depends on the nature of the data to be compressed. If the split compression size is large, more process private area is required to execute some SQL statements[#] that involve storing and extracting data in compressed columns. In order to avoid a memory shortage, specify an appropriate split compression size taking into account the available memory in the system and the value of the `pd_max_access_tables` operand. For details about the increase in the process-specific memory requirement, see *15.1.6(9) Determining the size of the memory required to execute data manipulation SQL statements on compressed columns* for a HiRDB single server configuration and *15.2.6(9) Determining the size of the memory required to execute data manipulation SQL statements on compressed columns* for a HiRDB parallel server configuration.
- If an error occurred during table reorganization processing and an unload data file (file unloaded by `pdorg -k rorg`) that contains compressed data is to be reloaded during the course of responding to the error, the table's compression specifications (whether compression specifications are used and the specified split compression size) must be the same for the unload source and the reload target. If the table contains a column whose compression specifications differ, `pdorg` will terminate with an error.
- If a compressed table is to be rebalanced in shared mode, the time required for the processing might be longer when the table contains compressed columns than when the table does not contain any compressed columns because of the time required for data compression and expansion processing. If you want to reduce the execution time, use exclusive mode.

[#]: This applies to the following SQL statements:

- SQL statements using the `SUBSTR` function
- SQL statements using the `POSITION` function
- SQL statements involving backward deletion/updating

12.21.7 How to measure the data compression rate

This subsection explains how to measure the data compression rate. Use the methods described here to see how much data will actually be compressed before you compress and store data, or to see how much data was compressed after you have stored data. Note that for columns of the abstract data type (XML type), the data compression rate cannot be measured after columns have been compressed and stored because such columns cannot be checked by `pdbst`.

(1) How to measure the data compression rate before data is compressed and stored

The data compression rate depends greatly on the nature of the data to be compressed. An exact compression rate cannot be obtained until after data has actually been compressed and stored. However, you can obtain an estimate of the compression rate by using the approximate length of the data compressed by `gzip`.[#] The formula is shown below.

[#]

`gzip` uses a compression algorithm that is equivalent to the one used by HiRDB (`Deflate`).

Formula:

compression rate (%) =

{(data length after compression obtained by `gzip` × 1.05[#]) ÷ data length before compression} × 100

#

An extra 5% is added to the compressed data size because of differences between `zlib` and `gzip` in the format of the headers that are added during compression processing for managing compression information.

(2) How to measure the data compression rate after data has been compressed and stored

The following shows the formula for obtaining an approximate compression rate after data has actually been compressed and stored.

Formula:

compression rate (%) =

(sum of data lengths after compression^{#1} ÷ sum of data lengths before compression^{#2})^{#3} × 100

#1: The following shows the calculation procedure:

1. Execute the database condition analysis utility (`pddbstat`) with the `-d` option specified for each RDAREA or table.
2. Based on the `<BINARY segment>` information in the output results, use the following formula to obtain the length of the compressed data:

10

$$\sum_{i=1} n_i \times a \times b$$

$i=1$

n_i : Maximum value of each ratio indicated as `Used Page Ratio` (number of used pages for each ratio) for binary-only segments (for example, if `Used Page Ratio` is 1 to 10%, then the maximum value is 0.1 (10%); if it is 11 to 20%, the maximum value is 0.2 (20%)).

a : Page value corresponding to n_i

b : Page size of binary-only segment

3. If the RDAREA or table processed in step 1 contains both compressed and uncompressed columns of the `BINARY` type, the data lengths of the uncompressed columns are subtracted from the results obtained in step 2. The data lengths of uncompressed columns are obtained by executing the following SQL statement:

```
select sum (length(uncompressed-column-name)) from table-identifier [in RDAREA-name]
```

#2

The length of the uncompressed data is obtained by executing the same SQL statement as is used for obtaining the data length of an uncompressed column (see step 3 in footnote #1).

#3

A value of 1.0 or greater means that the data length has increased after compression processing or that the effects of compression are small. In such a case, we recommend that you remove the compression specifications by changing the definitions of the columns to be compressed. For details about removing the compression specification, see *12.21.5 How to change the definition of a compressed column (removing the compression specification for a column)*.

12.22 Temporary tables

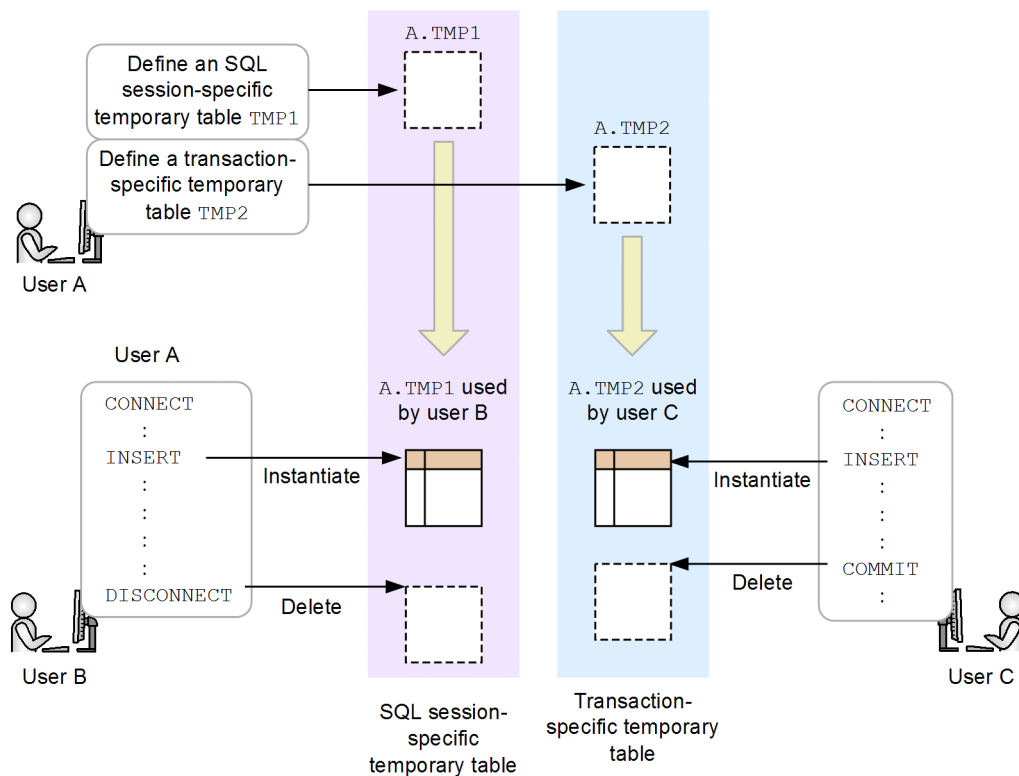
A *temporary table* is a base table that exists only during a transaction or an SQL session. A temporary table that exists only during a transaction is called a *transaction-specific temporary table*, and a table that exists only during an SQL session is called a *SQL session-specific temporary table*.

Temporary tables are not created when the table is defined. A temporary table is created when the first `INSERT` statement for the table is executed. This is called *instantiating* a temporary table.

A temporary table is created for each connection that is established for a single table definition (by executing the `CONNECT` statement). Therefore, a temporary table is not affected by data operations (`SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements) by another user even when multiple users use temporary tables at the same time. A temporary table and the indexes defined for the temporary table (*temporary table index*) are stored in a temporary table RDAREA and are deleted automatically when the transaction is completed or the SQL session is terminated. For details about temporary table RDAREAs, see *14.7 Temporary table RDAREAs*.

The following figure provides an overview of temporary tables.

Figure 12–46: Overview of temporary tables



Effects of using temporary tables

- If you perform complex processing during a transaction or SQL session, you can use a temporary table as a work table to store intermediate processing results and then obtain final results after further processing.
- If a part of a table containing many data items is accessed frequently during a transaction or SQL session, you can reduce the number of input and output operations by storing the corresponding data in a temporary table, thereby improving performance.
- Because temporary tables are deleted automatically when transactions are completed or SQL statements are terminated, no postprocessing is required by UAPs, thereby reducing the workload for UAP creation.

Criteria for temporary tables

We recommend that you use temporary tables for transactions that frequently access only part of a table containing many data items and for batch jobs that perform complex processing for which intermediate processing results need to be stored temporarily.

12.22.1 Valid period of data in temporary tables

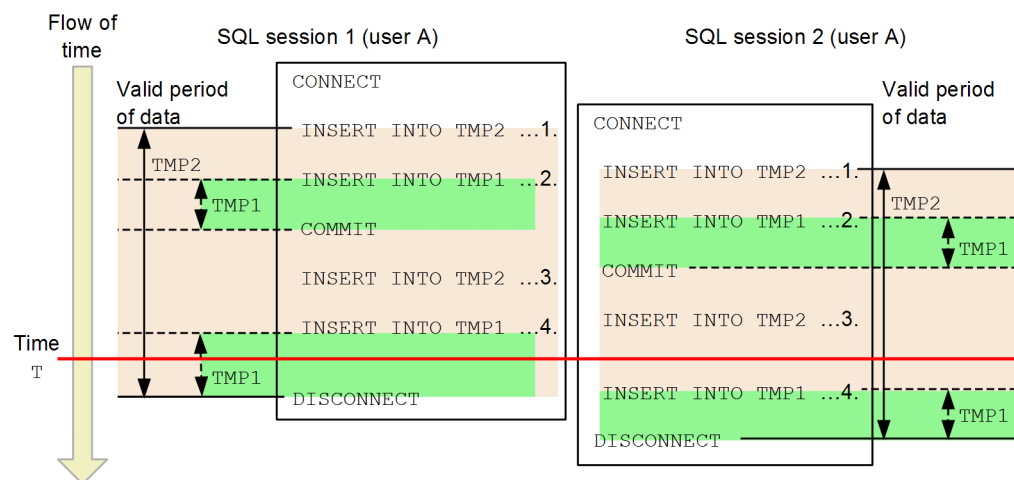
The valid period of data (period in which entities exist) in an instantiated temporary table depends on whether the temporary table is a transaction-specific temporary table or an SQL session-specific temporary table. The table below describes when the valid period of data begins and ends for a temporary table. Figure 12-47 and Figure 12-48 show examples of a valid period of data and the data stored at a given point in time.

Table 12–27: Start and end of a valid period of data for temporary tables

Type of temporary table	Start timing	End timing
Transaction-specific temporary table	When the first <code>INSERT</code> statement is executed on the temporary table during a transaction	When the transaction is completed
SQL session-specific temporary table	When the first <code>INSERT</code> statement is executed on the temporary table during an SQL session	<ul style="list-style-type: none"> When the SQL session is completed When the back-end server that instantiated the temporary table is terminated When the unit containing the back-end server that instantiated the temporary table is terminated When a system switchover occurs on the back-end server that instantiated the temporary table or on the unit containing that back-end server

Figure 12–47: Example of a valid period of data and the data stored at a given point in time (1)

- Manipulation of the transaction-specific temporary table `TMP1` and the SQL session-specific temporary table `TMP2`



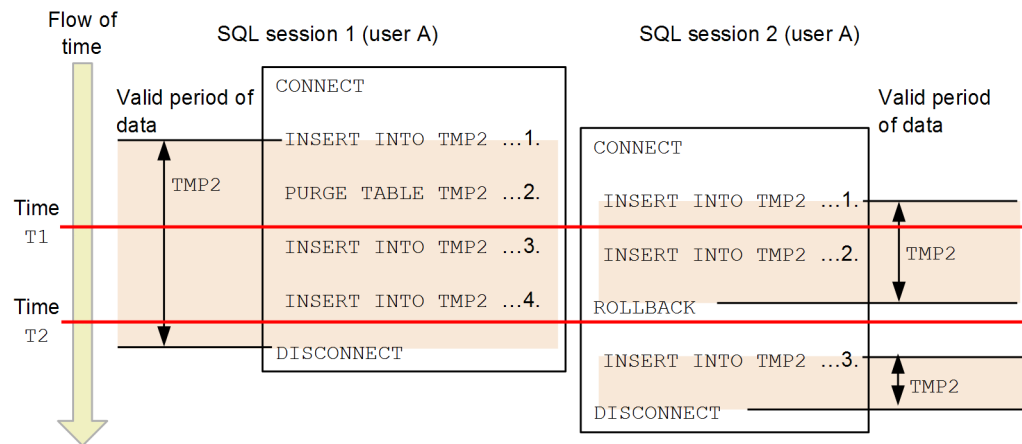
Explanation:

The following table shows at time `T` the data contained in temporary tables `TMP1` and `TMP2` which are used by SQL sessions 1 and 2:

SQL session	Temporary table	Data contained
SQL session 1	<code>TMP1</code>	Data inserted in 4
	<code>TMP2</code>	Data inserted in 1 and 3
SQL session 2	<code>TMP1</code>	No data at this point in time
	<code>TMP2</code>	Data inserted in 1 and 3

Figure 12–48: Example of a valid period of data and the data stored at a given point in time (2)

- Manipulation of the SQL session-specific temporary table `TMP2`



Explanation:

The following table shows at times T1 and T2 the data contained in temporary table `TMP2` which is used by the SQL sessions 1 and 2:

Time	SQL session	Data contained
T1	SQL session 1	There is no data at this point in time, because the table data was deleted in 2. However, there is an instantiation of <code>TMP2</code> .
	SQL session 2	Data inserted in 1
T2	SQL session 1	Data inserted in 3 and 4
	SQL session 2	There is no data at this point in time. There is no instantiation of <code>TMP2</code> , because processing has rolled back to the synchronization point before <code>TMP2</code> was instantiated.

! Important note

- Performing search, update, or deletion processing on a temporary table whose valid period of data has expired has the same result as when an SQL statement is executed on an empty table.
- For a HiRDB parallel server configuration, if a back-end server on which a SQL session-specific temporary table has been instantiated (or the unit containing such a back-end server) terminates abnormally or results in a system switchover, the valid period of data ends. Therefore, if data manipulation is attempted on the corresponding temporary table before the SQL session terminates, an SQL error results.

12.22.2 How to define temporary tables and temporary table indexes

(1) How to define temporary tables

Specify `GLOBAL TEMPORARY` in the `CREATE TABLE` definition SQL statement. To define a transaction-specific temporary table, specify `ON COMMIT DELETE ROWS`. To define an SQL session-specific temporary table, specify `ON COMMIT PRESERVE ROWS`. Note that for temporary tables, some operands are not allowed or are ignored if specified. For details, see `CREATE TABLE` in the manual *HiRDB Version 9 SQL Reference*.

(2) How to define temporary table indexes

The method is basically the same as when normal indexes are defined. For temporary table indexes, some operands are not allowed or are ignored if specified (as is the case with temporary tables). For details, see `CREATE INDEX` in the manual *HiRDB Version 9 SQL Reference*.

(3) How to specify a temporary table RDAREA for storing data

Specify in `PDTMPTBLRDAREA` in the client environment definition the name of a temporary table RDAREA that can be used. If you specify multiple RDAREAs or have omitted this environment definition, HiRDB determines the temporary table RDAREA to use for storing data according to the following rules:

- When multiple RDAREAs are specified in `PDTMPTBLRDAREA`
If the specified RDAREAs include both a temporary table RDAREA with the SQL session-specific attribute and a temporary table RDAREA with the SQL session shared attribute, HiRDB uses the temporary table RDAREA with the SQL session-specific attribute.
- When `PDTMPTBLRDAREA` is omitted
HiRDB uses a temporary table RDAREA with the SQL session shared attribute.

Note that in an XDS client, the specification is assumed to be omitted because `PDTMPTBLRDAREA` is ignored.

12.22.3 Rules for choosing an RDAREA for storage

When there are multiple temporary table RDAREAs, or when `PDTMPTBLRDAREA` is omitted from the client environment definition, HiRDB chooses a temporary table RDAREA for storing data. This subsection explains how HiRDB chooses the target RDAREA to use for storage.

(1) Choosing a target back-end server for storage (applicable to HiRDB parallel server configurations only)

For a HiRDB parallel server configuration, HiRDB first chooses a back-end server for storing data. HiRDB narrows down the candidate back-end servers based on the rules described below and then chooses a back-end server that accesses a base table that is not a temporary table among all the base tables specified in the `INSERT` statement.

- When RDAREAs are specified in `PDTMPTBLRDAREA` in the client environment definition
HiRDB chooses as the storage candidate the back-end server containing the specified RDAREAs.
If the specified RDAREAs include both temporary table RDAREAs with the SQL session-specific attribute and with the SQL session shared attribute, HiRDB uses the back-end server that contains the temporary table RDAREA with the SQL session-specific attribute.
- When no RDAREAs are specified in `PDTMPTBLRDAREA` in the client environment definition
HiRDB chooses as the storage candidate the back-end server containing a temporary table RDAREA with the SQL session shared attribute.

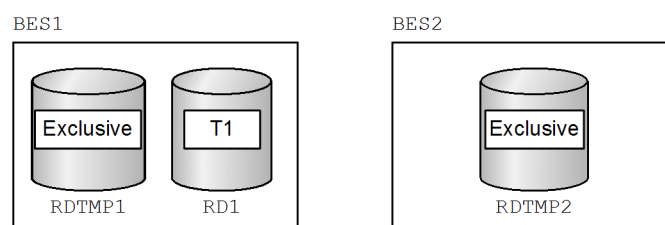
Tip

When HiRDB chooses a target back-end server for storage, it preferentially chooses a back-end server that accesses a base table that is not a temporary table among all the base tables specified in the `INSERT` statement. Therefore, you can avoid data transfer between back-end servers if you use `INSERT SELECT` as shown in the following example.

Example: `INSERT INTO TMP1 SELECT C1, C2, C3 FROM T1`

This SQL statement inserts into temporary table `TMP1` columns `C1`, `C2`, and `C3` from table `T1`.

The figure below show the configuration for executing this SQL statement. In this example, `RDTMP1` and `RDTMP2` are specified in `PDTMPTBLRDAREA`.



Legend:

Exclusive: Temporary table RDAREA with the SQL session-specific attribute

In this example, HiRDB chooses BES1 that contains table T1 as the target back-end server for storage. As a result, SQL statements can be executed without having to transfer data between BES1 and BES2.

(2) Choosing storage candidate RDAREAs

HiRDB chooses the storage candidate RDAREAs based on the specification of PDTMPBLRDAREA in the client environment definition. For details about the specification of PDTMPBLRDAREA, see *12.22.2(3) How to specify a temporary table RDAREA for storing data*.

(3) Choosing the RDAREAs that satisfy the conditions

From the storage candidate RDAREAs, HiRDB chooses RDAREAs that satisfy all the following conditions:

- RDAREAs that allow locks for temporary table operations to be acquired.
For details about acquiring locks for temporary table operations, see *12.22.5 Locking for temporary tables*.
- RDAREAs that are accessible from the UAP.
- If the temporary table to be stored is a FIX table, RDAREAs that can accommodate the temporary table's row length.
For details, see rule 3 for the FIX operand in *CREATE TABLE* in the manual *HiRDB Version 9 SQL Reference*.
- RDAREAs that can accommodate the total length of the columns that comprise any temporary table indexes that are to be stored.
For details, see common rule 5 in *CREATE INDEX* in the manual *HiRDB Version 9 SQL Reference*.
- RDAREAs for which the usage count for temporary tables is less than 500.
- RDAREAs for which the usage count for temporary table indexes is less than 500.
- RDAREAs that have unused segments.
- RDAREAs for which the total number of temporary tables and temporary table indexes does not exceed the specified `pd_max_temporary_object_no` operand value.
- If ACCESS is specified in the `pd_tmp_table_initialize_timing` operand, RDAREAs that are uninitialized temporary table RDAREAs.
For details about initialization of temporary table RDAREAs, see *14.7(4) Initializing the temporary table RDAREAs*.

(4) Choosing the temporary table RDAREA in which to store data

Among the RDAREAs satisfying the conditions, HiRDB preferentially uses the following temporary table RDAREAs:

- The temporary table RDAREA that contains the largest number of unused segments
- An uninitialized temporary table RDAREA if ACCESS is specified in the `pd_tmp_table_initialize_timing` operand.
For details about initialization of temporary table RDAREAs, see *14.7(4) Initializing the temporary table RDAREAs*.

12.22.4 Processing when there are no available temporary table RDAREAs

If no temporary table RDAREAs exist when HiRDB attempts to store data in a temporary table, HiRDB issues the KFPAL9704-E message and ignores the transaction. In such a case, the cause of the error indicated in the KFPAL9704-E message applies only to the first storage candidate RDAREA. If you have taken appropriate action for the RDAREA displayed in the message but the same message is issued again, check the status of other temporary table RDAREAs and take appropriate action for them also.

Action to be taken:

Execute the `pddbls -T` command.

Check the execution results to see if there is an RDAREA for which `OCCUPIED` or `SHARED` is displayed for `RDAREA_FOR_TEMPORARY_TABLE`.

- There is no RDAREA for which `OCCUPIED` or `SHARED` is displayed

If there are no available temporary table RDAREAs, you must add a temporary table RDAREA with the SQL session-specific attribute. If necessary, specify that RDAREA in `PDTMPTBLRDAREA` in the client environment definition.

- There are RDAREAs for which `OCCUPIED` or `SHARED` is displayed

If there are temporary table RDAREAs, then none of them satisfies the storage conditions (for details about the storage conditions, see *12.22.3(3) Choosing the RDAREAs that satisfy the conditions*). In order to determine which conditions are not satisfied, execute `pddbls -a -T` and `pddbst -k -phys` on the temporary table RDAREAs. Check the execution results for the items described in the table below and take appropriate action for the conditions that are not satisfied.

Table 12–28: Check items and actions

No.	Command	Check item	Description	Action
1	<code>pddbls -a -T</code>	STATUS	RDAREA's status	<p>If the RDAREA is in any of the following statuses, you must take the appropriate action explained below:</p> <ul style="list-style-type: none"> • Closed • Shut down • Status in which the <code>pdhold</code> command has been accepted <p>Change the RDAREA status by opening it or releasing its shutdown status so that the UAP can access the RDAREA. If the RDAREA is in error shutdown status, use the <code>pdmod</code> command to re-initialize the temporary table RDAREA (<code>initialize rdarea</code> statement).</p>
2		SEGMENT	Number of unused segments in the RDAREA	<p>If there are no unused segments, use the <code>pdmod</code> command to take one of the following actions:</p> <ul style="list-style-type: none"> • Add a temporary table RDAREA (<code>create rdarea</code> statement) • Re-initialize the existing temporary table RDAREA (<code>initialize rdarea</code> statement) • Expand the existing temporary table RDAREA (<code>expand rdarea</code> statement) • Change the attribute of the existing temporary table RDAREA (<code>alter rdarea</code>) to apply automatic extension.
3	<code>pddbst -k -phys</code>	Page Size	RDAREA's page length	<p>If the condition for page length is not satisfied, take one of the following actions:</p> <ul style="list-style-type: none"> • Add a temporary table RDAREA that satisfies the condition for page length.

No.	Command	Check item	Description	Action
				<ul style="list-style-type: none"> Change the page length of the existing temporary table RDAREA so that it satisfies the condition.
4		Unused Segment	Number of unused segments in the RDAREA	Same as 2 above

12.22.5 Locking for temporary tables

A temporary table contains specific data for a particular transaction or SQL session and is not available to be accessed by other users. In general, therefore, HiRDB does not acquire locks for table operations other than when a temporary table is instantiated. This subsection explains locking for temporary tables.

(1) Locks acquired when temporary tables are instantiated

When a temporary table is instantiated, HiRDB acquires the locks shown in the following table.

Table 12–29: Locks acquired when a temporary table is instantiated

Resource type name	Description	Unlock timing	
		Transaction-specific temporary table	SQL session-specific temporary table
RDAR	RDAREA (table and index ^{#1} storage RDAREA)	When the transaction is completed	When the DISCONNECT statement is completed
TABL	Table		
RATM	User directory table information		
RAIM	User directory index information ^{#1}		
SBMB	User directory segment information		
TEMP	Temporary table instantiation control information		After the temporary table has been instantiated ^{#2}
TPID	User-specific ID management (temporary table)		When the DISCONNECT statement is completed
RDAS	RDAREA status management		After the temporary table has been instantiated ^{#2}
RRAM, TRAL, TRAI	RDAREA management information		
PTBL	Table for preprocessing		When the DISCONNECT statement is completed

#1

Applicable only if there is a temporary table index.

#2

A lock for this resource is acquired temporarily during instantiation.

(2) Locks acquired for operations on temporary tables

When the following SQL statements are executed, a lock is acquired only for the preprocessing table (PTBL):

- LOCK statement

- CREATE INDEX statement
- DROP INDEX statement
- DROP TABLE statement

Note that executing the following SQL statements on a temporary table will not lock pages, rows, or key values:

- SELECT statement
- INSERT statement
- UPDATE statement
- DELETE statement
- PURGE TABLE statement

(3) Operations that might result in a lock-release wait status or an execution error

If you are performing an operation on a temporary table and, at the same time, you execute any of the following operations, a lock-release wait status or an execution error might result:

- `pdclose` (close a temporary table RDAREA)
- `pdhold` (shut down a temporary table RDAREA)
- `pdopen` (open a temporary table RDAREA)
- `pdrels` (release a temporary table RDAREA from shutdown status)
- `create rdarea` statement in the `pdmod` command (add a temporary table RDAREA)
- `initialize rdarea` statement in the `pdmod` command (re-initialize a temporary table RDAREA)
- `remove rdarea` statement in the `pdmod` command (delete a temporary table RDAREA)
- CREATE INDEX statement (define a temporary table index)
- DROP INDEX statement (delete a temporary table index)
- DROP SCHEMA statement (delete a schema)
- DROP TABLE statement (delete a temporary table)

12.22.6 Limitations on the use of temporary tables

(1) Operation commands and utilities

The operation command and the utilities listed below cannot be executed on temporary tables. For details, see the manual *HiRDB Version 9 Command Reference*.

- `pdorbegin` command
- Optimizing information collection utility (`pdgetcst`)
- Database load utility (`pdload`)
- Global buffer residence utility (`pdpgbfon`)
- Free page release utility (`pdreclaim`)
- Database reorganization utility (`pdrorg`)

(2) SQL statements

Limitations (such as that temporary tables and temporary table indexes cannot be specified) apply to the SQL statements listed below. For details, see the manual *HiRDB Version 9 SQL Reference*.

- Definition SQL statements
`ALLOCATE MEMORY TABLE`

ALTER INDEX
ALTER TABLE
CREATE INDEX
CREATE TABLE
CREATE TRIGGER
DROP INDEX
DROP SCHEMA
DROP TABLE
GRANT
REVOKE

- Data manipulation SQL statements
ALLOCATE CURSOR statement
ASSIGN LIST statement
DECLARE CURSOR statement
Dynamic SELECT statement
SELECT statement (table reference, query expression format 2)
DELETE statement
UPDATE statement
- Control SQL statements
COMMIT statement
DISCONNECT statement
ROLLBACK statement
LOCK TABLE statement
SET SESSION AUTHORIZATION statement

13

Designing Indexes

This chapter explains items that should be examined during design of an index with a B-tree structure or a plug-in index.

13.1 Items to be examined during index design

An index is created to improve table processing performance. However, a poorly designed index can have an adverse effect on performance. You should examine the methodology for creating effective indexes. Also, table processing performance and operability vary depending on the method used to store indexes in user RDAREAs. You should take these points into account when designing an index.

The following table lists items to consider when you are designing an index.

Table 13–1: Items to consider when you are designing an index

Design task and items to be examined	Advantages	Disadvantages	Section
Index creation	Table search performance is improved.	As the number of indexes created increases, overhead for index update processing also increases.	13.2
Index row partitioning	Table storage RDAREAs and index storage RDAREAs can be managed on a one-by-one basis, thereby improving utilities' operability.	If a non-partitioning key index is partitioned, the performance of a search using an index is reduced.	13.3
Creation of plug-in index	If a plug-in index is created in a column defined as an abstract data type using the index type provided by a plug-in, table search performance is improved.	As the number of indexes created increases, overhead for index update processing also increases.	13.4
Plug-in index row partitioning	User LOB RDAREAs can be handled independently during batch index creation.	Row partitioning results in an increase in RDAREAs. When backing up a database with RDAREAs specified or when reorganizing the database, note that the table and index have a one-to-one correspondence.	13.5

13.2 Index

This section describes the design of an index that has a B-tree structure.

13.2.1 Creating an index

(1) Effects of indexes

Improved performance

Table retrieval performance improves when an index is created for a column that is used as the key for table retrieval.

(2) Criteria

An index should be created for the following columns:

- Column used as a condition for narrowing the data to be retrieved
- Column used as a condition for table join processing
- Column used as a condition for data sorting or grouping
- Component column for which a referential constraint has been defined (foreign key)

An index should not be created for the following columns (if an index is created for such a column, retrieval performance will be degraded):

- Column that is updated frequently
- Column that contains many duplicated values

(3) Creation procedure

The `CREATE INDEX` definition SQL is used to create an index for a table.

(4) Common rules

1. A maximum of 255 indexes can be defined per table.
2. Indexes can be defined for columns with null values or columns with no rows.
3. Indexes cannot be created for view tables.
4. When optimizing indexes based on cost, use the optimizing information collection utility (`pdgetcost` command) to collect optimizing information as necessary to improve the accuracy of optimization. For details about the necessity of executing this utility, see *Optimizing information collection levels* in the manual *HiRDB Version 9 Command Reference*.

(5) Data types for which indexes cannot be defined

Indexes cannot be specified for columns of the following types:

- BLOB
- BINARY
- Abstract data types

(6) Maximum index key length

The length of an index key must satisfy the following condition; if this condition is not satisfied, the index cannot be defined:

Index key length (bytes)

$$\leq \text{MIN}\{(\text{index-storage-RDAREA-page-size} \div 2) - 1,242, 4,036\}$$

If the page size of the index storage RDAREA is 4,096 bytes, the maximum key length that can be specified for an index is 806 bytes. For details about index key length, see Table 16-5 *List of index key lengths*.

For a multicolumn index, the total index key length is the total of the key lengths of the columns that make up the index.

(7) Notes

The same index cannot be created more than once for the same table. The following examples show how indexes can be regarded as being the same index in spite of having different index names.

■ Single-column index

```
CREATE INDEX index-1 ON table-1 (column-1 ASC)
CREATE INDEX index-2 ON table-1 (column-1 DESC)
```

In this case, *index-2* is treated as the same index as *index-1*. Therefore, *index-1*, which was defined first, is the valid one.

■ Multicolumn index

```
CREATE INDEX index-1 ON table-1 (column-1 ASC, column-2 ASC)
CREATE INDEX index-2 ON table-1 (column-1 DESC, column-2 DESC)
```

or

```
CREATE INDEX index-1 ON table-1 (column-1 ASC, column-2 DESC)
CREATE INDEX index-2 ON table-1 (column-1 DESC, column-2 ASC)
```

In this case, *index-1* and *index-2* are treated as the same index. Therefore, *index-1*, which was defined first, is the valid one. In the following case, on the other hand, the indexes are treated as different indexes:

```
CREATE INDEX index-1 ON table-1 (column-1 DESC, column-2 DESC)
CREATE INDEX index-2 ON table-1 (column-1 ASC, column-2 DESC)
```

13.2.2 Index creation taking into account optimizing based on cost

If a table has multiple indexes, HiRDB selects for use the index with the lowest access cost based on the search conditions specified for the table retrieval. This index selection process is called optimizing based on cost.

HiRDB takes into account the following factors in estimating access cost:

- Hit rate based on the specified search conditions
- Number of input/output operations required for SQL processing
- CPU workload required for SQL processing

HiRDB provides better table retrieval performance because it optimizes processing based on cost. Table retrieval performance will not be reduced even when an SQL statement that specifies complicated search conditions is executed.

(1) Index creation criteria taking into account optimizing based on cost

Because HiRDB optimizes processing based on cost, the user can create a UAP without having to prioritize the indexes to be used by HiRDB. However, the user should examine beforehand how an index should be created for a table that is to be accessed by UAPs.

To take advantage of optimizing based on cost, an index that is to be used by HiRDB should be created taking into account its priority. Consideration should also be given to the difference between a single-column index and a multicolumn index, the use of multiple indexes, and performance depending on the number of indexes.

The following table lists the order of priority for index usage by HiRDB.

Table 13–2: Order of priority for index usage by HiRDB

Priority	Index used by HiRDB	Example of condition specification for index column (C1)
1 Always used ^{#1}	Plug-in index specified for the column in the first argument of an index type plug-in function whose condition is IS TRUE.	contains (C1, '...') IS TRUE
	Index that contains as its index component columns all the columns in the search condition in a structured repetition predicate.	ARRAY (C1, C2) [ANY] (C1='ABC' and C2=10) C1 and C2 define a multi-column index.
2	Plug-in index specified for the column in the first argument of a plug-in-provided function whose condition is IS TRUE	within (C1, '...') IS TRUE
3	Index with UNIQUE specified for a column that is subject to the = limitation condition.	C1=100
4	Index for a column subject to the = limitation condition.	C1=100
5	Index for a column subject to the IS NULL limitation condition ^{#2} .	C1 IS NULL
6	Index for a column specified for a prefix search using a literal (%) in the LIKE or SIMILAR predicate pattern character string.	C1 LIKE 'ABC%' C1 SIMILAR TO 'ABC%'
7	Index for a column specified for a prefix search other than the above using a literal in the LIKE or SIMILAR predicate pattern character string.	C1 LIKE 'ABC_' C1 SIMILAR TO 'ABC_'
8	Index for a column subject to a limitation condition in the IN predicate.	C1 IN (10, 20, 30)
9	Index for a column subject to a limitation condition in the BETWEEN predicate.	C1 BETWEEN 20 AND 40
	Index for a column for which a range condition is specified.	20<=C1 AND C1<=40
10	Single-column index for a column subject to a limitation condition in the IN predicate using a subquery that has no external reference.	C1 IN (SELECT C1 FROM T2)
	Single-column index for a column subject to a limitation condition in the =ANY or =SOME quantified predicate using a subquery that has no external reference.	C1=ANY (SELECT C1 FROM T2) C1=SOME (SELECT C1 FROM T2)
11	Index for a column subject to the >, >=, <, or <= limitation condition.	C1>50 C1<=200
12 ^{#3}	Index for a column specified for a scalar operation (system-defined scalar function, other than IS_USER_CONTAINED_IN_HDS_GROUP) ^{#2} .	length (C1)=10
13	Index for a column subject to a limitation condition in the NOT BETWEEN predicate.	C1 NOT BETWEEN 10 AND 30
14	Index for a column subject to a limitation condition in the XLIKE predicate, or in LIKE or SIMILAR predicates other than the above.	C1 XLIKE '%ABC%' C1 LIKE '%ABC%' C1 SIMILAR TO '%ABC%'

Priority	Index used by HiRDB	Example of condition specification for index column (C1)
15	Index for a column specified in an argument of the set function (MIN or MAX) ^{#4} .	MIN (C1) MAX (C1)
16	Index for a join condition column or in a column subject to grouping or sorting.	ORDER BY C1
--	Index for a column subject to a negation limitation condition (except NOT BETWEEN).	C1 NOT LIKE '%ABC%' C1 IS NOT NULL
	Index for a column subject to a limitation condition in the quantified predicate ANY or SOME other than the above.	C1>=ANY (SELECT C1 FROM T2) C1>SOME (SELECT C1 FROM T2)
	Index for a column subject to a limitation condition in the quantified predicate ALL.	C1>ALL (SELECT C1 FROM T2)
	Plug-in index specified for the column in the first argument of a plug-in-provided function whose condition is IS FALSE or IS UNKNOWN	within(C1, '...') IS FALSE

Legend:

--: Indexes that are not used.

Notes

1. The `contains` function call is a function provided by the HiRDB Text Search Plug-in.
2. The `within` function call is a function provided by the HiRDB Spatial Search Plug-in.
3. An index cannot be used if it is for a column subject to a limitation condition that contains a subquery involving external referencing.
4. If indexes can be used in the conditional expressions on both the terms of the OR operator, the priority depends on the predicate used in the conditional expressions.
5. A limitation condition refers to a search condition other than the join condition.
6. HiRDB may not use a defined index if it determines that the index cannot be used effectively.

#1: The index indicated as Always used in the Priority column must be defined; otherwise, an error results.

#2: For the following types of columns, do not create an index whose exception key is the null value:

- Column for which the `IS NULL` limitation condition is specified.
- Column for which a limitation condition includes `VALUE` and `CASE` expressions.
- Column with the `BIT_AND_TEST` limitation condition for which `IS UNKNOWN`, `IS NOT TRUE`, or `IS NOT FALSE` is specified.

You can create indexes with limitation conditions other than as indicated above. Table 13-3 shows whether HiRDB uses an index whose exception key is the null value.

#3: Only when **Key conditions that include a scalar operation** is selected as an SQL optimization option does an index have this usage priority. For details about SQL optimization options, see the *HiRDB Version 9 UAP Development Guide*. Depending on the predicate, an index may have a better priority. If negation is not included, the priority order is in the range of 13-15; if negation is included, the priority order is 13 or up.

#4: In the case of an SQL statement specifying one table without specifying `GROUP BY`, the index for the column specified in the argument is used if only one set function (MIN or MAX) is specified and one of the following conditions is satisfied:

- The component column of a single-column index is specified in the set function's argument.
- The column specified in the set function's argument is component column *n* of a multicolumn index without an exception key value and `=` or `IS NULL` is specified in component columns 1 through *n*-1.
- The column specified in the set function's argument is component column *n* of a multicolumn index with an exception key value and `=` is specified in component columns 1 through *n*-1.

Table 13–3: Whether HiRDB uses an index whose exception key value is the null value

Limitation condition specified in the component column		Whether index is used
IS NULL, VALUE, CASE expression, and BIT_AND_TEST	Other than IS NULL, VALUE, CASE expression, and BIT_AND_TEST ^{#1}	
Specified	Specified	Used
Specified	Not specified	Not used
Not specified	Specified	Used ^{#2}
Not specified	Not specified	Not used ^{#3}

#1: Applicable to the limitation conditions for priority levels 4-15 shown in Table 13-2.

#2: HiRDB may not use the index if it determines that the index cannot be used effectively.

#3: The index is used for retrieval if all the following conditions are satisfied:

- The selection expression consists of only set functions that use the index component column as the argument.
- Only one table is specified in the FROM clause.
- The WHERE clause is not specified.

If indexes are created consistent with the priorities shown in Table 13-2, favorable results can be obtained in narrowing the search conditions specified in the SQL statement. However, an index with a high priority may not be used if HiRDB determines as a result of cost-based optimization that its use would not be effective.

13.2.3 Single-column index vs. multicolumn index

The two types of indexes are single-column indexes and multicolumn indexes. A single-column index is an index based on the values in one column of a table. A multicolumn index is an index based on the values in multiple columns of a table.

(1) Creating a single-column index

A single-column index should be created when retrieval will be executed using one column only as the key.

(2) Creating a multicolumn index

A multicolumn index should be created in the cases discussed below.

(a) Retrieval of data that satisfies multiple conditions

A multicolumn index should be created when data satisfying multiple conditions is to be retrieved, such as when a complex-condition retrieval using the AND operator with multiple columns as the key is executed.

For example, suppose that a complex-condition search is to be executed using table columns C1, C2, and C3 as the key items:

```
SELECT retrieval-column FROM retrieval-table WHERE C1=10 AND C2=20 AND C3=30
```

In this case, a multicolumn index consisting of the three columns C1, C2, and C3 should be created instead of creating three separate single-column indexes. In this way, overhead for index and row accesses can be reduced.

When a complex-condition retrieval is to be executed, it is important that the column for which the equals (=) condition is specified be defined as the first component column of the multicolumn index. Then the column that is next most likely to have the equals condition should be specified, followed by the third column, and so on. As a result, the retrieval range can be reduced within the index, thereby reducing the retrieval time. If the equals condition is not specified for the first component column of a multicolumn index, appropriate retrieval results may not be obtained from the index. In this case, better results may be achieved by using a single-column index.

(b) Grouping or sorting data after narrowing the data with a search condition

A multicolumn index should be created using the columns specified as the search condition then the columns to be grouped or sorted, in this order.

Suppose that a complex condition retrieval is executed using table columns C1 and C2 as the key, and then the retrieval results are sorted in descending order of C3 and ascending order of C4, as shown as follows:

```
SELECT retrieval-column FROM retrieval-table WHERE C1=10 AND C2=20
ORDER BY C3 DESC,C4 ASC
```

In this case also, a multicolumn index consisting of columns C1, C2, C3, and C4 should be created, instead of creating two single-column indexes in columns C1 and C2. The data in column C3 should be sorted in descending order, and the data in column C4 should be sorted in ascending order, so that overhead for index and row accesses is reduced.

(c) Duplicated multicolumn indexes created for one table

If a multicolumn index consisting of columns C1 and C2 is created for a table together with another multicolumn index consisting of columns C1 and C3, overhead increases when the duplicated column, C1, is updated. To reduce this overhead, one multicolumn index consisting of C1, C2, and C3 should be created.

Note that if the table is retrieved using columns C1 and C3 as the search conditions, retrieval performance may be reduced.

(d) Priority between single-column and multicolumn indexes

If both single-column and multicolumn indexes are created for the same table, HiRDB uses the indexes in the priority order shown in the following table. This table assumes that search condition C1=10 AND C2=20 is specified for table retrieval.

Table 13–4: Priority among single-column and multicolumn indexes

Columns constituting the index			Priority
Component column 1	Component column 2	Component column 3	
C1	C2	None	1
C1	C3	C2	2
C1	None	None	3
C1	C3	None	4
C3	C2	None	5

13.2.4 Using multiple indexes

More than one index can be created for a table. It is more effective for purposes of narrowing the rows to be retrieved to use multiple indexes than to use a single index (single-column or multicolumn index).

13.2.5 Using an index with an exceptional key value set

When an index is defined for a column, all the data in the column is loaded into the index as the index values. Sometimes an index will contain unused values, such as the null value. In this case, the null value can be specified as an exceptional key value so that its occurrences will be excluded from the index. This is appropriate for an index that contains many occurrences of the null value in all its component columns.

(1) Effects of setting an exceptional key value for an index

The following are the effects of setting an exceptional key value for an index:

1. The size of the index is reduced because the null value key is not created in the index.

2. Overhead for index maintenance during row insertion, deletion, and update processing (CPU time, number of input/output operations, number of lock requests, and frequency of deadlock) is reduced, in addition to the amount of log information being reduced.
3. When the null value is specified as the exceptional key value and `IS NULL`, `VALUE`, or `CASE` expression is specified as the search condition for the index component column, then the index is not used for the retrieval processing. As a result, the retrieval performance is improved in the following case:
 - Input/output operations occur on the same page because the index contains many occurrences of the null value and the data page is accessed at random.

(2) Setting procedure

An exception value is set by specifying `EXCEPT VALUES` in the `CREATE INDEX` definition SQL.

(3) Notes

- The only key value that can be specified as an exceptional key value is the null value.
- An exceptional key value cannot be specified for an index that contains a column with the `NOT NULL` constraint.
- An exceptional key value cannot be specified for an index for which a cluster key is specified.
- An index with an exceptional key value cannot be specified for unloading in index order.

13.2.6 Effects on performance of the number of indexes

When rows are added to or deleted from a table, all indexes created for the table are updated. Therefore, as the number of indexes increases, the overhead for index updating increases. Thus, the following considerations should be taken into account when indexes are created:

- Do not define an index for a column that is updated frequently.
- Create multicolumn indexes to reduce the number of indexes.
- In the case of a HiRDB parallel server configuration, create the minimum number of indexes required in order to improve the effects of parallel processing, especially when retrieving all entries.

13.3 Index row partitioning

If you partition a table, you can also partition and store its index in multiple user RDAREAs.

(1) Partitioning key index and non-partitioning key index

Before designing a row-partitioned index, you need to understand a partitioning key index and a non-partitioning key index.

An index that satisfies a specified condition is a partitioning key index, while an index that does not satisfy a specified condition is a non-partitioning key index. This condition depends on whether the table is a single-column partitioning or multicolumn partitioning table.

Note

A table partitioning condition based on only one column corresponds to single-column partitioning, and a table partitioning condition based on multiple columns corresponds to multicolumn partitioning.

(a) Single-column partitioning

An index that satisfies one of the following conditions is a partitioning key index:

Conditions:

- Single-column index defined for a column for which storage conditions were specified when partitioning the table (partitioning key)
- Multicolumn index with a component column 1 for which storage conditions were specified when partitioning the table (partitioning key)

The following figure shows an index that qualifies as a partitioning key index, using the inventory chart below as an example.

Figure 13–1: Partitioning key index (single-column partitioning)

STOCK

PCODE	PNAME	COLOR	PRICE	QUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29

Example setting for the partitioning conditions (partitioning key)

Explanation:

```
CREATE INDEX A12 ON STOCK (PCODE ASC)           ...1
CREATE INDEX A12 ON STOCK (PCODE ASC, PRICE DESC) ...2
CREATE INDEX A12 ON STOCK (PRICE DESC, PCODE ASC) ...3
```

1. If the partitioning key column PCODE is specified as an index, it becomes a partitioning key index. If any other column is specified as an index, it becomes the non-partitioning key index.
2. If the partitioning key column PCODE is specified as component column 1 of a multicolumn index, the multicolumn index becomes a partitioning key index.
3. If the partitioning key column PCODE is specified as a component column other than component column 1, the multicolumn index becomes a non-partitioning key index.

(b) Multicolumn partitioning

An index that satisfies the following condition is a partitioning key index:

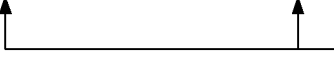
Condition:

- Index created on multiple columns that includes all columns specified for partitioning in the same order, beginning with the partitioning key

The following figure shows an index that qualifies as a partitioning key index, using the inventory chart below as an example.

Figure 13–2: Partitioning key index (multicolumn partitioning)

STOCK				
PCODE	PNAME	COLOR	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29


 Example settings for the partitioning conditions
(partitioning keys)

CREATE TABLE STOCK ~
 HASH HASH1 BY PCODE, PRICE ~

Explanation:

```

CREATE INDEX A12 ON STOCK (PCODE ASC, PRICE DESC)           ... 1
CREATE INDEX A12 ON STOCK (PCODE ASC, PRICE DESC,          ... 2
                        SQUANTITY ASC)
CREATE INDEX A12 ON STOCK (PRICE DESC, PCODE ASC)           ... 3
CREATE INDEX A12 ON STOCK (PCODE ASC, SQUANTITY           ... 4
                        DESC, PRICE ASC)
  
```

1. This multicolumn index becomes a partitioning key index because it specifies all partitioning keys (columns PCODE and PRICE), and the order of these partitioning keys is the same as when the table was defined.
2. This multicolumn index becomes a partitioning key index because it specifies all partitioning keys (columns PCODE and PRICE), and the order of these partitioning keys is the same as when the table was defined.
3. This multicolumn index becomes a non-partitioning key index because it specifies all partitioning keys (columns PCODE and PRICE), but the order of these partitioning keys is not the same as when the table was defined.
4. This multicolumn index becomes a non-partitioning key index because it specifies all partitioning keys (columns PCODE and PRICE), but the order of these partitioning keys is not the same as when the table was defined.

(2) Index partitioning guidelines

Guidelines for index partitioning depend on whether the index is a partitioning key index or a non-partitioning key index, as shown in the following table.

Table 13–5: Index partitioning guidelines

Type of index	HiRDB single server configuration	HiRDB parallel server configuration	
		Table partitioned by rows within one server	Table partitioned by rows among multiple servers
Partitioning key index	Index is also row-partitioned according to its row-partitioned table.	Index is also row-partitioned according to its row-partitioned table.	Index is also row-partitioned according to its row-partitioned table.
Non-partitioning key index	Index should not be row-partitioned. Row-partitioning the index may result in poor performance during a search using the index. [#]	Index should not be row-partitioned. Row-partitioning the index may result in poor performance during a search using the index. [#]	

[#]: You should not row-partition a non-partitioning key index. Row-partitioning the index may result in poor performance during a search using the index. Specifically, a search using any of the following paths is disabled, adversely affecting the search performance:

- KEY SCAN MERGE JOIN

- LIST SCAN MERGE JOIN
- L-KEY R-LIST MERGE JOIN
- L-KEY R-SORT MERGE JOIN
- L-LIST R-KEY MERGE JOIN
- L-LIST R-SORT MERGE JOIN
- L-SORT R-KEY MERGE JOIN
- L-SORT R-LIST MERGE JOIN

For details about these access paths, see the access path display utility (`pdvwopt` command) in the manual *HiRDB Version 9 Command Reference*.

However, if there is a large amount of table data, you should consider index row partitioning. Row-partitioning an index enables table storage RDAREAs and index storage RDAREAs to be managed on a one-by-one basis, thereby improving utilities' operability. For example, when the index is not row-partitioned, if you load data in units of RDAREAs or reorganize each RDAREA, you need to create an index in batch mode after data loading or reorganization is completed. If you row-partition the index, there is no need to execute such batch index creation after loading data in units of RDAREAs or reorganizing each RDAREA.

If an index is defined for a matrix-partitioned table, row partitioning is required just as with partitioning keys even if there is a non-partitioning key index.

(3) Design considerations

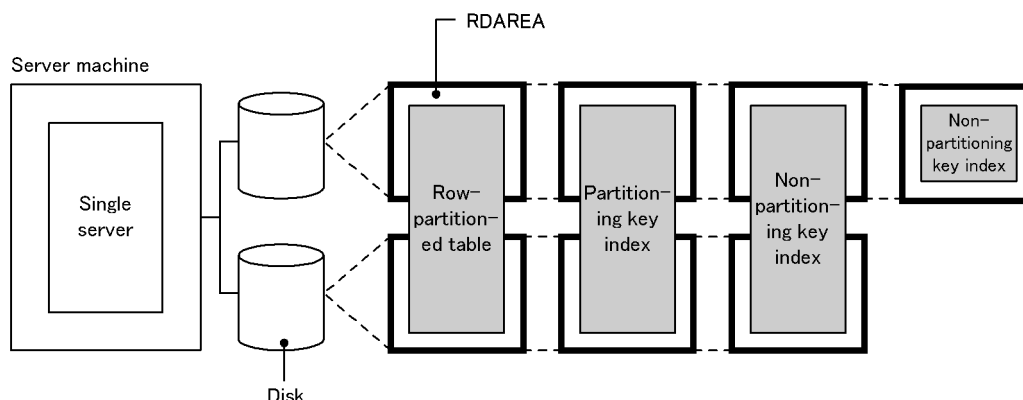
- You should use separate user RDAREAs for a row-partitioned table and for its index. This improves the utilization efficiency of the user RDAREAs.
- If a table contains a key that is to be made unique, you should define a partitioning key index with `UNIQUE` specified for this key, or you should specify a cluster key for the partitioning key. `UNIQUE` can be specified for either of the following, even if there is a non-partitioning key index.
 - Non-partitioning index
 - Partitioning index with the partitioning key included in a random constituent column

However, `UNIQUE` cannot be specified for the index if the table is flexible hash partitioned. For details, see *Specifiability of UNIQUE in conjunction with row-partitioning of a table under CREATE INDEX* in the manual *HiRDB Version 9 SQL Reference*.

(4) Example of index row partitioning (HiRDB single server configuration)

The following figure shows an example of index row partitioning (for a HiRDB single server configuration).

Figure 13–3: Example of index row partitioning (HiRDB single server configuration)



Explanation:

- To avoid disk access contention, place the RDAREAs storing the partitioned table on a disk separate from the RDAREAs storing its index.

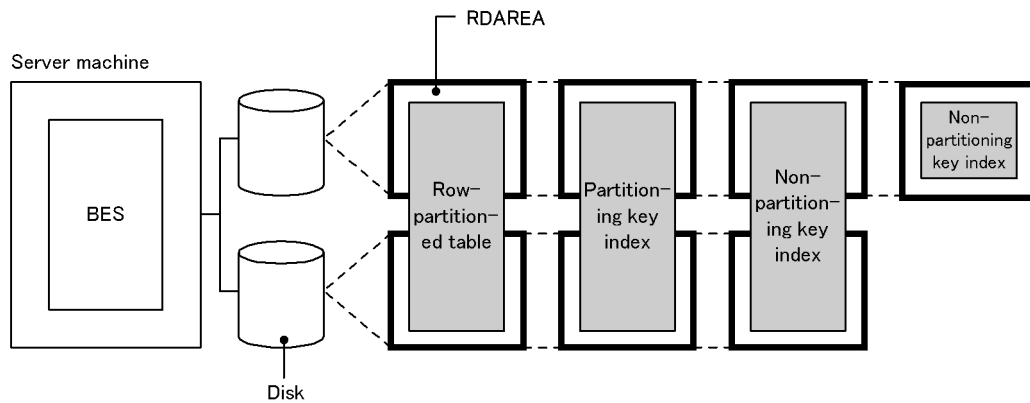
- Row-partition the partitioning key index.
- If performance is more important than operability, do not row-partition the non-partitioning key index.
- If operability is more important than performance, row-partition the non-partitioning key index.

(5) Example of index row partitioning (HiRDB parallel server configuration)

(a) Partitioning a table within one server

The following figure shows an example of index row partitioning (in one server).

Figure 13–4: Example of index row partitioning (within one server)



BES: Back-end server

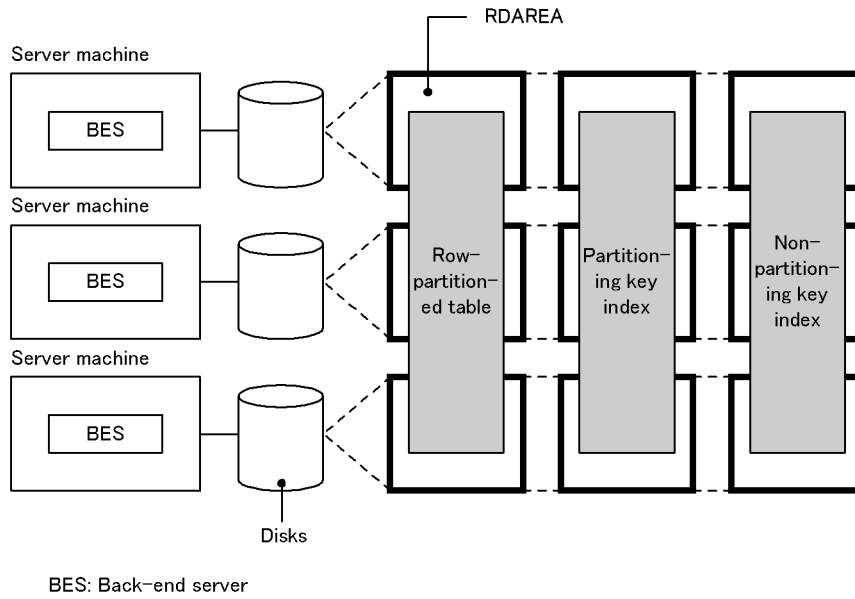
Explanation:

- To avoid disk access contention, place the RDAREAs storing the partitioned table on a disk separate from the RDAREAs storing its index.
- Row-partition the partitioning key index.
- If performance is more important than operability, do not row-partition the non-partitioning key index.
- If operability is more important than performance, row-partition the non-partitioning key index.

(b) Partitioning a table among multiple servers

The following figure shows an example of index row partitioning (among multiple servers).

Figure 13–5: Example of index row partitioning (among multiple servers)



Explanation:

- To avoid disk access contention, place the RDAREAs storing the partitioned table on a disk separate from the RDAREAs storing its index.
- Row-partition the partitioning key index as well as the non-partitioning key index.

13.4 Plug-in index

This section describes plug-in indexes.

(1) Effects of plug-in indexes

Improved performance

When a plug-in is used, table retrieval performance can be improved by creating a plug-in index. The user can execute complicated retrieval processing at high speed by using the index types provided by plug-ins.

(2) Creation procedure

The `CREATE INDEX` definition SQL is used to create a plug-in index for a table.

(3) Notes

Some plug-ins require definition of a plug-in index. If a function that uses a plug-in index is specified without the plug-in index having been defined, an error may result during execution.

(4) Batch creation of plug-in index

You can use the database load utility (`pdload`) to create a plug-in index in the batch mode. For details about batch creation of a plug-in index, see *6.4 Creating a table containing a plug-in-provided abstract data type*.

13.5 Plug-in index row partitioning

When you partition a table, you also need to partition its plug-in index and store it in multiple user LOB RDAREAs.

(1) Effects of plug-in row partitioning

Improved operability

When a plug-in index is created in the batch mode, the portion of the plug-in index in each user LOB RDAREA is processed independently.

(2) Definition procedure

For details about how to define plug-in index row partitioning, see 6.4 *Creating a table containing a plug-in-provided abstract data type*.

(3) Forms of plug-in index row partitioning

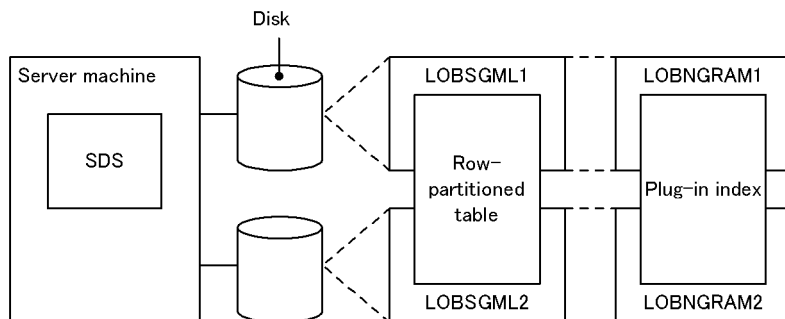
The forms of plug-in index row partitioning are described below for a HiRDB single server configuration and for a HiRDB parallel server configuration.

(a) HiRDB single server configuration

For a HiRDB single server configuration, a plug-in index can be partitioned and stored in multiple user LOB RDAREAs on multiple disks on the same basis as the row-partitioned table.

Figure 13-6 shows a form of plug-in index row partitioning. Figure 13-7 shows an example of plug-in index row partitioning based on the form shown in Figure 13-6.

Figure 13-6: Form of plug-in index row partitioning (HiRDB single server configuration)



SDS: Single server

LOBSGML1, LOBSGML2, LOBNGRAM1, and LOBNGRAM2: user LOB RDAREAs

Note There is a one-to-one correspondence between the table and the plug-in index.

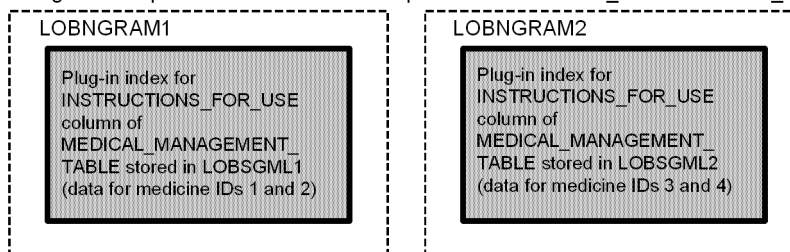
Figure 13–7: Example of plug-in index row partitioning (key range partitioning) (HiRDB single server configuration)

- MEDICAL_MANAGEMENT_TABLE partitioned and stored in user LOB RDAREAs LOBSGML1 and LOBSGML2

LOBSGML1 MEDICAL_MANAGEMENT_TABLE	
MEDICINE_ID	INSTRUCTIONS FOR USE
Medicine 1	<data> <indications> Diarrhea, food poisoning, contaminated water ingestion,... </indications> <directions and dosage> Adults (20 years and up): 10 tablets,... after supper. </directions and dosage> : <warnings> Keep out of the reach of children... Keep refrigerated after opening... </warnings> </data>
Medicine 2	<data> <indications> Headache, toothache, arthritis, back ache, ... </indications> <directions and dosage> Adults (20 years and up): 5 packets,... maximum of 5 packets in a 24-hour period. </directions and dosage> : <warnings> Keep out of the reach of children... For other than headache, ... </warnings> </data>
:	:

LOBSGML2 MEDICAL_MANAGEMENT_TABLE	
MEDICINE_ID	INSTRUCTIONS FOR USE
Medicine 3	<data> <indications> Common cold, cold-like symptoms, ... </indications> <directions and dosage> 5 times daily... within 5 minutes after a meal... </directions and dosage> : <warnings> Keep out of direct sunlight... Do not operate automobiles, etc., after taking this medicine... </warnings> </data>
Medicine 4	<data> <indications> Eye strain, tight shoulder, ... </indications> <directions and dosage> Adults (20 years and up): 10 tablets daily... </directions and dosage> : <warnings> Keep out of the reach of children... </warnings> </data>
:	:

· Plug-in index partitioned on basis of row-partitioned MEDICAL_MANAGEMENT_TABLE



Explanation:

The example assumes that a plug-in index is defined for the INSTRUCTIONS_FOR_USE column.

The MEDICAL_MANAGEMENT_TABLE is partitioned and stored in user LOB RDAREAs LOBSGML1 and LOBSGML2 using the MEDICINE_ID column as the condition. The plug-in index is stored in LOBNGRAM1 and LOBNGRAM2.

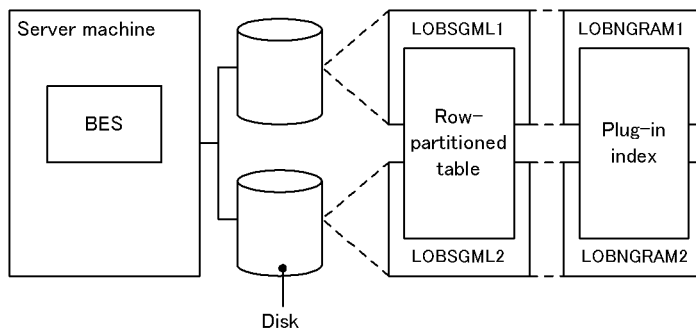
(b) HiRDB parallel server configuration

For a HiRDB parallel server configuration, a plug-in index can be partitioned and stored in multiple user LOB RDAREAs located in multiple server machines or back-end servers, on the same basis as its row-partitioned table.

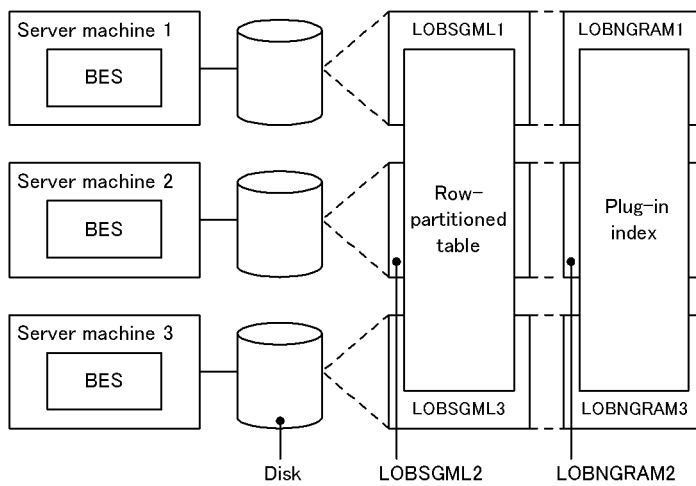
Figure 13-8 shows a form of plug-in index row partitioning. Figure 13-9 shows an example of plug-in index row partitioning based on the form shown in Figure 13-8.

Figure 13–8: Form of plug-in index row partitioning (HiRDB parallel server configuration)

- Row partitioning in a back-end server



- Row partitioning in multiple back-end servers



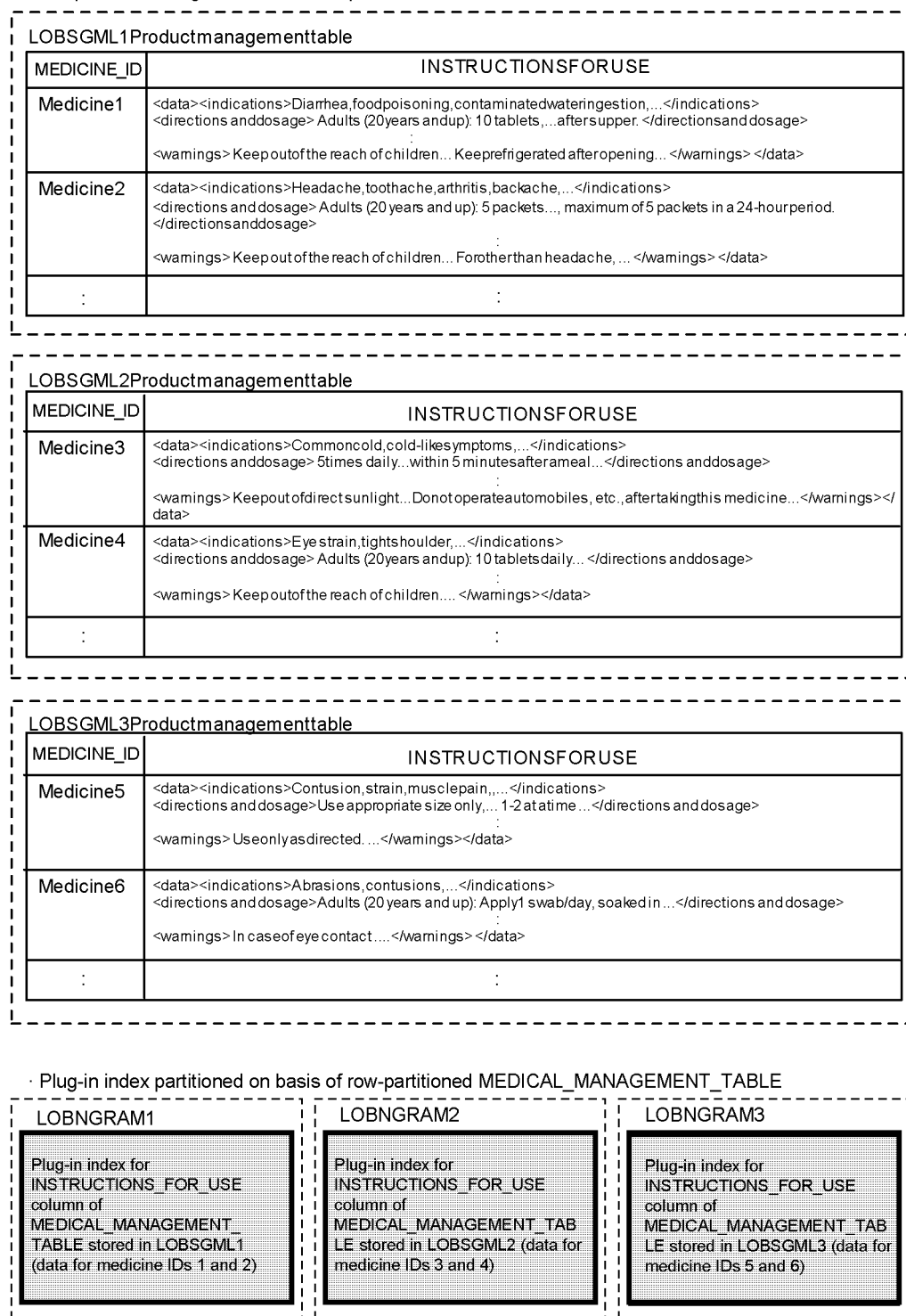
BES : Back-end server


LOBSGML1 through LOBSGML3 and LOBNGRAM1 through LOBNGRAM3:
user LOBRDAREAs.

Note: There is a one-to-one correspondence between the table and the plug-in index.

Figure 13–9: Example of plug-in index row partitioning (key range partitioning) (HiRDB parallel server configuration)

- The product management table is row partitioned onto user LOBRDAREAs LOBSGML1-LOBSGML3.



 : Plug-in index

Explanation:

The example assumes that a plug-in index is defined for the OPERATION_MANUAL column.

`MEDICAL_MANAGEMENT_TABLE` is partitioned and stored in user LOB RDAREAs `LOBSGML1-LOBSGML3` using the `MEDICINE_ID` column as the condition. The plug-in index is stored in `LOBNGRAM1`, `LOBNGRAM2`, and `LOBSGML3`.

(4) Design considerations

Separate user LOB RDAREAs should be used for a row-partitioned table and for its plug-in index.

(5) Notes

Row partitioning results in an increase in the number of RDAREAs; therefore, when the database is backed up with RDAREA specified, the table and its index will have a one-to-one correspondence.

14

Designing RDAREAs

This chapter explains items that should be examined while designing the segments and pages that constitute RDAREAs.

14.1 Items to be examined during RDAREA design

The amount of disk space required depends on the sizes of segments and pages that constitute RDAREAs. You should take this point into account when designing RDAREAs. *Table 14-1 Items to be examined during RDAREA design* lists the items to be examined during RDAREA design, and *Table 14-2 Maximum and minimum values for RDAREAs* lists the maximum and minimum values for RDAREAs.

Table 14–1: Items to be examined during RDAREA design

Design task and items to be examined		Advantages	Disadvantages	Section
Segment size	Size increased	If the length of a row changes as a result of update processing or if a row is added to a table for which a cluster key is specified, unused pages can be allocated that are adjacent to the page containing the specified row, thereby reducing the data input/output time.	Because the number of segments is reduced, the number of tables and indexes that can be stored per user RDAREA is also reduced.	14.2.1
	Size reduced	If many tables, each of which contains a small amount of data, are stored in one user RDAREA, wasted space caused by unused pages can be minimized.	<ul style="list-style-type: none"> • If a large amount of data is added to a user RDAREA, the number of segment allocations increases, resulting in an increase in overhead. • Because the number of segments increases, the amount of locked resources also increases when a table is deleted or all rows are deleted from a table. 	
Per-cent-age of free space in segment	Specified	When data is added to a table for which a cluster key is specified, data can be stored in the page close to the cluster key value, thereby reducing the number of data input/output operations.	As the value becomes larger, more disk space is required.	14.2.2
	Set to 0	The disk space required can be reduced.	When data is added to a table for which a cluster key is specified, data cannot be stored in the page close to the cluster key value, resulting in poor storage status; therefore, reduction in the number of data input/output operations is no longer beneficial.	
Page length	Percent-age of unused space in page specified	<ul style="list-style-type: none"> • If a row becomes longer as a result of UPDATE statement processing, and the contiguous free space is longer than the updated row, the corresponding line fits in the page. • When rows are added repeatedly by the INSERT statement, rows can be added until the page located close to the cluster key value becomes full. 	For a table with the <code>FIX</code> attribute, storage efficiency is poor.	14.3.2
	Percent-age of unused space in page set to 0	For a table with the <code>FIX</code> attribute, storage efficiency is improved if the data is placed in ascending order.	If a row becomes longer than before as a result of update processing, the row spans multiple pages, resulting in overhead in row accesses.	
Free space reuseage	Used	<ul style="list-style-type: none"> • Free space in the used segments can be used effectively. • The performance of free space search after the RDAREA is full is improved. 	If there is insufficient free space for reuse, the overhead for free space search increases.	14.5

Design task and items to be examined		Advantages	Disadvantages	Section
	Not used	If there is adequate free space, rapid insertion processing is possible.	RDAREA storage efficiency is reduced. Performance of free space search after the RDAREA is full is degraded.	
Shared RDAREA	Used	If a heavily accessed table that is difficult to partition is stored in a shared RDAREA, the efficiency of parallel processing improves because the table can be referenced by all back-end servers.	When a shared table is updated, the shared RDAREA containing the table is locked, and deadlock may occur if an application accesses another table in the shared RDAREA.	14.6
	Not used	Deadlock and server-to-server global deadlock, which sometimes result from use of a shared RDAREA, are avoided.	For complex search processing, such as join processing, overhead associated with connection establishment between multiple back-end servers and with data transfer increases.	
Temporary table RDAREA	Used	Temporary tables can be used to perform complex data processing, as well as for executing transactions and SQL sessions without being affected by other users. Temporary tables do not require postprocessing.	There is overhead for initializing a temporary table RDAREA when HiRDB starts or when the first INSERT statement is executed on a temporary table.	14.7
	Not used	There is no overhead for initializing temporary table RDAREAs when HiRDB starts.	When intermediate processing results are stored in a table during complex processing, postprocessing is required (such as deleting data after completion of processing).	

Table 14–2: Maximum and minimum values for RDAREAs

Item	Maximum and minimum values
Total number of RDAREAs	3 to 8,388,592
Number of master directory RDAREAs	1
Number of data directory RDAREAs	1
Number of data dictionary RDAREAs	1 to 41
Number of user RDAREAs	1 to 8,388,589
Number of data dictionary LOB RDAREAs	1 to 2
Number of user LOB RDAREAs	0 to 8,388,325
Number of registry RDAREAs	0 to 1
Number of registry LOB RDAREAs	0 to 1
Number of list RDAREAs	0 to 8,388,588
Number of HiRDB files per RDAREA	1 to 16
Number of base tables per RDAREA	0 to 500
Number of indexes per RDAREA	0 to 500
Number of lists per RDAREA	0 to 50,000
Total number of HiRDB files	1 to 134,217,728

- Estimating the size of index storage RDAREAs

For details about how to estimate the size of index storage RDAREAs, see *16.1 Determining the size of a user RDAREA*. The following lists notes about size estimation:

1. Data is stored orderly immediately after an index is created in the batch mode using the database load utility or database reorganization utility. The size of the index continues to increase thereafter due to index page splitting unless all keys are inserted in ascending order during data insertion.
2. In general, index pages do not reuse used free pages. Therefore, if there is an update or deletion that changes a key value, the page where the key was stored before the update or deletion cannot be reused. For this reason, there are used free pages that are wasted and are not reused. However, there are operations that can reuse used free pages. For details see the *HiRDB Version 9 System Operation Guide*.
3. The structure of an index depends on whether there are duplicated key values. You need an accurate number of duplicate values to estimate the accurate size of an index. The ratio of this error to the size of the index becomes greater as the number of index records decreases.

14.2 Segments

The following table lists the statuses that a segment can have.

Table 14–3: Segment statuses

Segment status	Explanation
Used segment [#]	<p>This is a segment that stores table or index data.</p> <p>A segment that is full, such that no more data can be added, is called a <i>full segment</i>. A used segment that is not full is called a <i>non-full segment</i>.</p> <p>A used segment from which data has been deleted so that only free pages remain (used free pages or unused pages) is called a <i>used free segment</i>.</p>
Unused segment	This is a segment that has never been used. Such a segment can be used by all tables (or indexes) in the RDAREA.
Free segment	This is a segment that does not store any data. Used free segments and unused segments are <i>free segments</i> .

[#]: Used segments can be used only by tables or indexes that have data stored in them. Other tables or indexes cannot use such segments.

14.2.1 Determining the segment size

We normally recommend an RDAREA segment size of about one-tenth the RDAREA storage page count. The segment size will be less than one-tenth for large RDAREAs, however, since the maximum segment size is 16,000 pages. This subsection describes the implications of segment size and considerations that should be taken into account when the segment size is selected.

(1) Selecting a large segment size

Improved performance

- If the length of a row changes as a result of update processing or if rows are added to a table for which a cluster key is specified, the data input/output time can be reduced because unused pages can be acquired adjacent to the particular page that contains the rows.
- The effects of batch input can be achieved by the prefetch facility because the data in a table is stored in consecutive pages. When the prefetch facility is used, the segment size should be the same as the maximum number of pages for batch input that is specified with the `-p` option of the `pdbuffer` operand in the system common definition.

Note:

- The number of tables and indexes that can be stored per user RDAREA is reduced because the number of segments per RDAREA is reduced.

(2) Selecting a small segment size

Reduction in required disk space

- The number of unused pages can be reduced because many tables, each containing a small amount of data, can be stored in one user RDAREA.

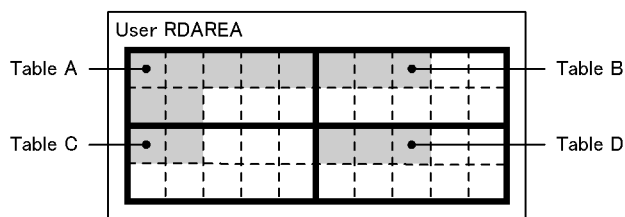
Notes

- If a large amount of data is added to a user RDAREA that is based on a small segment size, the segment allocations count increases, thereby increasing overhead.
- Because the number of segments increases, the amount of locked resources also increases when a table is deleted or all rows are deleted from a table.

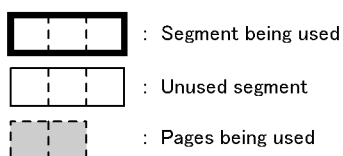
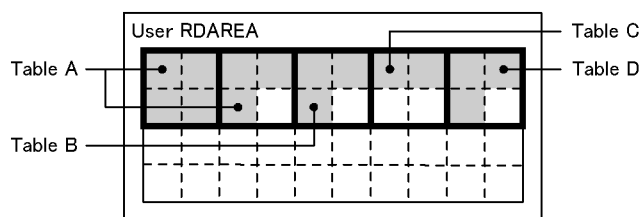
The following figure provides an overview of user RDAREAs depending on segment size.

Figure 14–1: Overview of RDAREAs depending on segment size

- Large segment size



- Small segment size



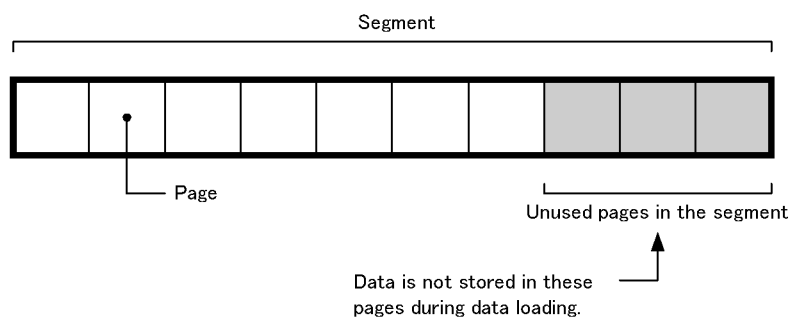
(3) Setting procedure

The `create rdarea` statement of the database initialization utility (`pdinit`) or the database structure modification utility (`pdmod`) is used to set the segment size.

14.2.2 Setting the percentage of free pages in a segment

The percentage of unused pages allocated in a segment when a table is defined is called the *percentage of free pages in a segment*. Free pages here refers to unused pages. The following figure provides an overview of the free pages in a segment.

Figure 14–2: Overview of percentage of free pages in a segment



(1) Effects of specifying a percentage of free pages in a segment

Improved performance:

When data is added to a table for which a cluster key is specified, the data is stored in a page close to the cluster key value, which means that the number of data input/output operations is reduced.

(2) Criteria

- A percentage of free pages in a segment should be set if a large amount of data will be added to a table for which a cluster key is specified after data has been stored by the database load utility (`pdload`).
- The percentage of free pages in a segment should be set at 0 if data addition or update processing on the table will occur rarely.

(3) Specification

To specify the percentage of free pages in a segment, use the `PCTFREE` operand of the `CREATE TABLE` definition SQL statement.

(4) Notes

If the percentage of free pages in a segment is set to 0 for a table for which a cluster key is specified, it will not be possible to store added data close to the cluster key values. As a result, the data storage condition becomes poor and reduction in the number of data input/output operations can no longer be expected.

14.2.3 Allocating and releasing segments

When a table is defined, segments are not allocated. Segments are allocated as needed when data is to be stored in the table. Once a segment has been allocated (once a segment has been used), no other table or index can use that segment until the segment has been released. With repeated additions and deletions of data, an RDAREA may run out of space even though the data volume has not increased. To avoid this, you should perform the following operations periodically in order to release segments:

- Use the database reorganization utility (`pdreorg` command) to reorganize the tables or indexes
- Release used free segments with the free page release utility (`pdreclaim` command)

For details about table reorganization, index reorganization, and releasing used free segments, see the *HiRDB Version 9 System Operation Guide*.

In addition to the above methods, you can release segments by performing any of the following types of operations:

- Deleting the table definition
- Deleting all rows of the table
- Deleting the table owner (schema)
- Deleting the index definition
- Deleting the primary key of the table
- Reinitializing RDAREAs
- Executing data loading in creation mode (`-d` option specified)
- Freeing segments as a result of table reorganization

Furthermore, in the following cases, the segments allocated by the transaction are released:

- If a rollback occurs during load processing of the database load utility or batch index creation
- If a rollback occurs during reload processing of the database reorganization utility

14.3 Pages

The following table lists the statuses that a page can have.

Table 14–4: Page statuses

Page status	Explanation
Unused page	An unallocated page.
Used free page	A page that stores no data because its data has been deleted. [#]
Used page	A page that stores data but has free space where data can be added. For tables that are using the free space reuse facility, this includes pages to which data cannot be added because the free space created by deleting data [#] from the page cannot be used.
Used full page	A page that stores data and has no free space where data can be added. For tables and indexes that are not using the free space reuse facility, this includes pages to which data cannot be added because the free space created by deleting data from the page [#] cannot be used.

#

Free space created by data deletion cannot be used until the transaction that executed the data deletion is committed.

14.3.1 Determining the page length

(1) Considerations in determining the page length

The considerations that should be taken into account in determining the page length are discussed as follows.

1. A large page size should be used for an RDAREA when a table or index satisfying the following conditions is to be stored by an application that retrieves or updates all entries or a large amount of data:
 - RDAREA stores tables that do not have indexes
 - RDAREA stores tables with a cluster key specified and their indexes
 - RDAREA stores indexes used for range condition retrieval or updating of a large amount of data
2. The page length should be set on the basis of the row length of the tables stored in the RDAREA so that invalid space can be eliminated as much as possible:

$$\text{invalid space} = \text{MAX}(\text{mod}((\text{page length} - 48), (\text{row length} + 2)), \text{page length} - 48 - (\text{row length} + 2) \times 255)$$
3. The following formula should be used as a guideline to setting the percentage of unused space in a page:

$$(\text{Page length} \times \text{percentage of unused space in a page}) \div 100 - \text{row length} \times \text{number of rows that can be stored in unused space in a page}.$$

A meaningless value that does not allow even one row to be stored in the unused space in a page should not be specified.
4. For a page used to store an index, approximately 4,096 to 8,192 bytes is an appropriate size in terms of input/output efficiency.
5. If a column's data type is VARCHAR, NVARCHAR, or MVARCCHAR and its definition length is at least 256 bytes, its data will be branched onto another page. If there is variable-length character string data with a length of at least 256 bytes, the page length should be set to the smallest value that is at least the average length of the data.
6. In the case of a column whose data type is VARCHAR, NVARCHAR, or MVARCCHAR, if a row of null values is inserted by the INSERT statement, the column's data may be branched onto another page depending on the length of the updated data when the UPDATE statement is used subsequently to update the null-value data to real data. If character string data is often set initially to the null value and then updated later to real data, the page length should be determined taking into account the length of the updated rows.

7. HiRDB allows locking control in units of pages or rows. If row-level locking control is to be used, the page length should be set on the basis of the row length so that as many rows as possible can be stored per page. The following should be taken into account in this case:

- Minimize the percentage of unused space in a page.
- Define the page length to minimize the global buffer lock-release waits count for page input/output requests. In the case of a frequently updated table, small pages should be used; otherwise, the lock-release waits count may increase.
- Define the page length to lower the page input/output waits count becomes low for the number of page input/output requests. If the application uses mainly random accesses, the page length should be small; otherwise, the actual input/output units become too large for the row length, which is the access unit, resulting in unneeded data transfers.

If the `UPDATE` statement is used frequently to update data in a column whose data type is `VARCHAR`, `NVARCHAR`, or `MVARCHAR` and this updating results in a change in the row length, the percentage of unused space in a page should be set to a slightly higher value when the table is defined. For details about how to set the percentage of unused space in a page, see *14.3.2 Setting the percentage of unused space in a page*.

(2) Specification

To specify a page length, use the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`).

(3) Notes on determining the page length

An error results when a row is added to a table and as a result the actual row length exceeds the page length (except in the case of columns whose data type is `VARCHAR`, `NVARCHAR`, or `MVARCHAR`). The actual row length is obtained using the formula for required disk space that is provided in *Chapter 16. Determining RDAREA Size*. If the obtained row length is greater than the page length of the user RDAREA to be used, the user RDAREA must be reinitialized and then the page size must be redefined. The database structure modification utility (`pdmod`) is used to reinitialize RDAREAs. For details about how to reinitialize RDAREAs, see the *HiRDB Version 9 System Operation Guide*.

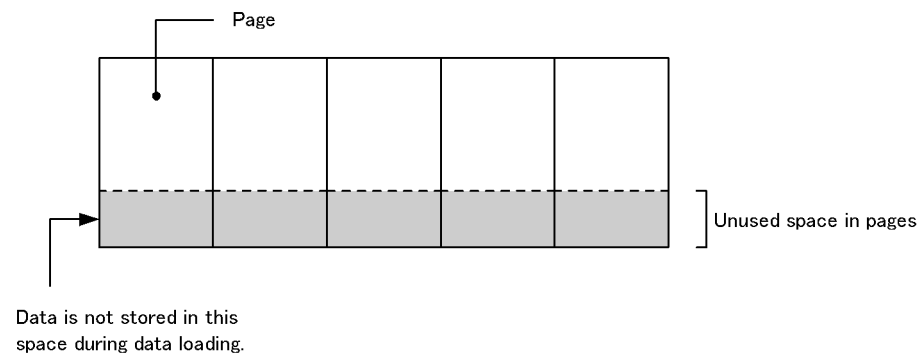
14.3.2 Setting the percentage of unused space in a page

The percentage of unused space allocated in a page when a table or index is defined is called the *percentage of unused space in a page*. When an unused space value is set, the database load utility (`pdload`) and database reorganization utility (`pdreorg`) will not normally store data in the specified amount of space.

However, if the database load utility is executed with the `-y` option specified and no new page can be allocated, it will store data in the specified unused space.

The following figure provides an overview of the unused space in pages.

Figure 14–3: Overview of unused space in pages



(1) Effects of setting a percentage of unused space in a page

- If the length of contiguous free space is longer than the row length after update processing, the corresponding row can fit in the page even if it has become longer than its original length as a result of UPDATE statement processing.
- When the INSERT statement is used to add rows repeatedly, the pages close to the cluster key value can become filled with rows.

(2) Criteria

1. You should specify a percentage of unused space in a page if rows will be added to a table for which a cluster key is specified.
2. For a table with the `FIX` attribute, if data will be sorted in ascending order, you can improve the storage efficiency by setting the percentage of unused space in a page to 0.
3. You should specify a percentage of unused space in a page if rows will become longer as a result of update processing.
4. Rows become longer when the following types of update processing are executed:
 - The null value is updated to real data.
 - A column with the `VARCHAR`, `NVARCHAR`, `MVARCHAR` or `BINARY` data type is updated so that the value becomes longer.

(3) Specification

To specify the percentage of unused space in a page, use the `PCTFREE` option of the definition SQL statement `CREATE TABLE`, `CREATE INDEX`, or `ALTER TABLE`.

(4) Notes

If the set amount of unused space is too small and a row becomes longer as a result of update processing, the number of input/output operations increases because a single row spans multiple pages.

(5) Obtaining the percentage of unused space in a page

- Generally, the value obtained from the following formula is used as the percentage of unused space (where the length of the first row stored is $L1$ and becomes $L2$ after processing):

$$\text{Percentage of unused space in a page} = ((L2 - L1) \div L2) \times 100 (\%)$$
- The following procedure should be used when a cluster key is specified for a table:
 1. Obtain the number of data items per page that are stored in the table by the database load utility (`pdload`); assume that this value is m .
 2. Obtain the number of data items that will be stored later; assume that this value is n .
 3. Use the following formula to obtain the percentage of unused space in a page from m and n obtained in steps 1 and 2:

$$\text{Percentage of unused space in a page} = (n \div (m + n)) \times 100 (\%)$$

14.3.3 Allocating and releasing pages

(1) Allocating pages

When a table is defined, pages are not allocated. Pages are allocated as needed when data is to be stored in the table. Once a page has been allocated (once a page has been used), the page cannot be reused until it has been released.

If an index is defined, the system allocates pages according to the number of data items. If there is no data item, the system allocates only one page (root page). If you specify the `EMPTY` option in the `CREATE INDEX` statement (so as not to create the index entity), the system does not allocate any page.

Notes

1. If you update data in such a manner that the row length of a non-FIX table changes, the space created by the reduced row length cannot be reused.
2. An index page cannot be reused until a key value that is identical to a key value that was stored in the deleted page is added.
3. Reusing a page freed up by deletion of data is subject to the following restrictions:
 - The page cannot be used for rows that contain a repetition column or a column whose type is `VARCHAR` of at least 256 bytes, `BINARY` type, or abstract data type.
 - Until a segment's usage reaches 100%, the page cannot be used for insertion of data.
 - Until a transaction that issued a `DELETE` has been committed, the free space generated by the deletion cannot be used.

(2) Releasing pages

- When a segment is released, the pages in the segment are also released.
- When a table has been locked with the `LOCK` statement with `EXCLUSIVE` specified, pages will be released when the UAP deletes all rows on the pages. The index pages are not released.
- When the `PURGE TABLE` statement is executed, the pages and segments of the tables and indexes are released. However, the root pages of the indexes remain.
- You release used free pages with the free page release utility (`pdreclaim` command). For details about releasing used free pages, see the *HiRDB Version 9 System Operation Guide*.

14.4 Designing list RDAREAs

(1) Number of required list RDAREAs

You can use the following operands to specify the maximum number of lists that can be created per list RDAREA:

- `create rdarea` statement's `max entries` operand in the database initialization utility (`pdinit`)
- `create rdarea` statement's `max entries` operand in the database structure modification utility (`pdmod`)
- `initialize rdarea` statement's `max entries` operand in the database structure modification utility (`pdmod`)

The permitted range of maximum values is 500 to 50,000.

(2) How to obtain a page length and a segment size

A list contains its base table's row identifiers. Unlike in tables, no data is stored directly in the list; therefore, a comparatively large number of rows can be stored in one page. Note that if the specified page length and segment size are too large for the actual number of rows to be stored in the list, unneeded free space is created in the RDAREA.

To determine the page length and segment size for a list RDAREA, estimate the average number of rows in the list that may be created within the server, then specify the appropriate page length and segment size based on one of the following cases:

Condition	Page length	Segment size
Average number of rows in a list created within the server is less than 3,000	4,096	1
Average number of rows in a list created within the server is 3,000 to 6,000	4,096	2
Average number of rows in a list created within the server is more than 6,000	See (a)	See (b)

(a) Obtaining the page length when the average number of rows in a list is more than 6,000

Specify the page length in the range of 4,096 to 8,192. If you want to reduce the list input/output time by reducing the number of list input/output operations, a larger page size may be specified. If the page length is large, the required size of the global buffer also increases, thereby requiring a large amount of shared memory.

Specify the page length that satisfies the following condition:

Condition:

$$\text{Number of rows that can be stored in one list page} \leq \text{average number of rows in the list created within the server} \div 2$$

To obtain the number of rows that can be stored in one list page, use the following formula:

$$\text{Number of rows that can be stored in one list page} = \downarrow \{ \text{page length} - 70 - (a \times 8) \div 4 \} \downarrow$$

a: Total number of HiRDB files in the RDAREAs that contain the list's base table within the server

(b) Obtaining the segment size when the average number of rows in a list is more than 6,000

The segment size is a unit size of space in an RDAREA that can be allocated to a single list. This means that one segment is the smallest size that can be allocated to a list. Following are the guidelines for the segment size:

- To reduce the overhead of segment allocation, increase the segment size.
- If you use the prefetch facility with the global buffer for a list RDAREA, specify a value of at least 2 for the segment size. Otherwise, the prefetch facility will not function.
- If the segment size increases, the possibility of creating unneeded unused pages in a segment also increases. To reduce such unneeded unused pages, specify a small segment size.
- Specify the segment size that satisfies the following condition:

$$\text{Number of rows that can be stored per list segment} \leq \text{average number of rows in list within the server} \div 2$$

To obtain the number of rows that can be stored in one list segment, use the following formula:
 Number of rows that can be stored per list segment = *number of rows that can be stored in one list page* \times
segment size

(3) How to obtain the number of segments

You can use the following formula to obtain the number of segments required for a list RDAREA:

Formula

$$\text{Number of segments required for a list RDAREA} = \lceil \lceil a \div b \rceil \times (c + 0.5) \rceil$$

a: Number of lists within the server

b: Number of list RDAREAs within the server

c: Average number of segments that are used per list

Obtain this value using the following formula:

$$\lceil \text{average number of rows in list within the server} \div \text{number of rows that can be stored per list segment} \rceil$$

If a segment shortage occurs, the system can no longer create a list. Therefore, specify a sufficient value based on the number of segments obtained from the previous formula.

14.5 Free space reusage facility

The free space reusage facility makes free space reusable once the data it stores has been deleted. This section explains the following items:

- Data storage search modes
- Free space reusage facility
- Effects and applicability
- Considerations
- Environment settings
- Checking execution status
- Notes

14.5.1 Data storage search modes

Once data has been stored in a table, either of the following two page search modes can be used to search the storage area:

- New page allocate mode
When the final page of a used segment becomes full, a new unused segment is allocated. If no unused pages remain in the RDAREA, free space to store the data is searched for in used pages from the beginning of the used segments.
When there are unused segments, storage efficiency will not improve, but processing can be performed at high speed. However, performance will drop significantly when there are no more unused segments.
- Free page reuse mode
When the final page of a used segment becomes full, free space is searched for in the used pages of the used segments before any unused segments are allocated. The search start position is then remembered for next search, and the subsequent search for free space begins from that point.
Storage efficiency will improve even when there are unused segments because free space is searched for and data is then stored, but this also entails overhead.

14.5.2 Free space reusage facility

The *free space reusage facility* uses the free space on used pages by switching the page search mode to the free page reuse mode once the number of a table's used segments reaches the number of user-specified segments and those segments have all become full. If there is no free space in any of the specified number of segments, it switches to the new page allocate mode for allocation of a new unused segment.

When no segment count has been specified, the free page reuse mode is used when there are no more unused pages in the RDAREA.

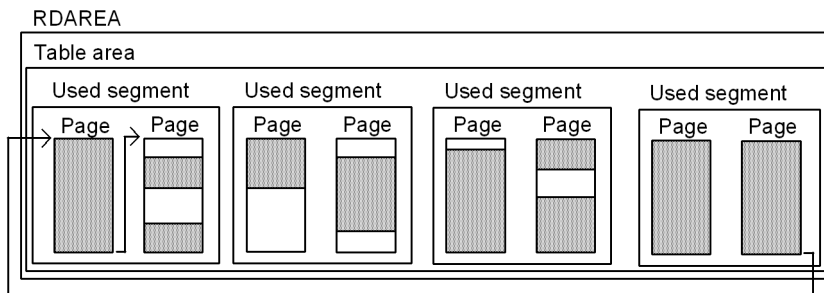
When there are no more unused pages in the RDAREA, search efficiency is better in the free page reuse mode when the free space reusage facility is used than when it is not used. In free page reuse mode, the next search position is remembered and subsequent searches are performed from there. If this facility is not used, searches always start from the beginning.

Note that if the free space reusage facility is not used, operations always use the new page allocate mode. In this case, performance will drop significantly when there are no more unused segments. To prevent this, increase the number of unused segments, such as by reorganizing the table or by releasing used free pages and used free segments.

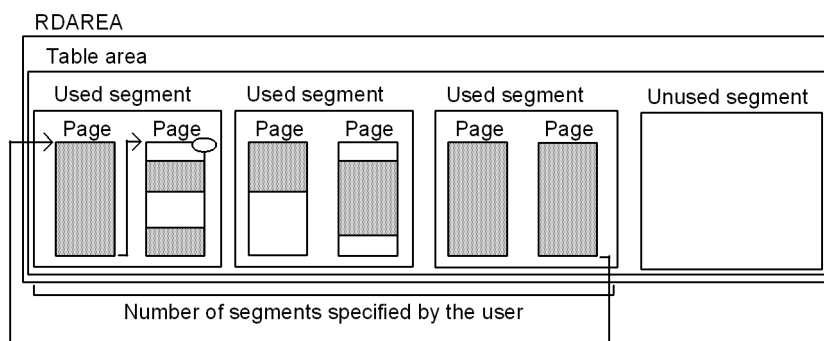
The following figure provides an overview of the free space reusage facility.

Figure 14–4: Overview of the free space reuse facility

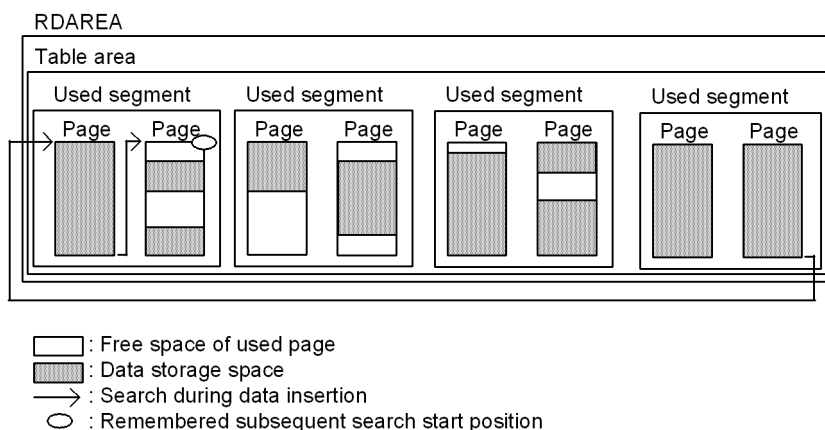
- Free space reuse facility not used



- Free space reuse facility used (number of segments specified)



- Free space reuse facility used (number of segments not specified)



Explanation:

- When the free space reuse facility is not used
When there are no more unused pages in the RDAREA, free space to store the data each subsequent time that data is inserted is searched for on used pages from the beginning of the used segments.
- When the free space reuse facility is used and the number of segments is specified
If there is an attempt to insert data into a table once the specified number of segments has been reached, an unused segment is not allocated, but instead free space to store the data is searched for in used pages from the beginning of the used segments. The search start position is then remembered for the subsequent search, and searching begins at that position the next time.
- When the free space reuse facility is used and the number of segments is not specified
When there is an attempt to insert data when there are no unused pages in the RDAREA, free space to store the data is searched for in used pages from the beginning of the used segments. The search start position is then remembered for the subsequent search, and searching begins at that position the next time.

14.5.3 Effects and applicability

(1) Effects

The following effects can be expected with the use of this facility:

- **Effective reuse of free space**
By reusing the free space of used pages, operations can be performed using a minimum amount of RDAREA space, thereby minimizing the frequency of database reorganization. If multiple tables and indexes are stored in the same RDAREA, the insertions and deletions for some tables can be combined, such that occupied area can be recovered.
- **Recovery from an insufficient pages error for variable-length columns and BINARY type columns**
Normally, if the no-split option is not specified, unused pages are allocated whenever a variable-length character column of at least 256 bytes is inserted or a BINARY type column that does not fit on one page is inserted. Even if used free pages are available, an error will result if an unused page cannot be allocated. If the free space reuse facility is being used, however, errors can be avoided because used free pages will be allocated if no unused free pages are available.
- **Reduction in overhead during a search for free space on used pages**
In the free page reuse mode, high-speed processing is possible due to the reduced overhead because the search start position is remembered and is used for the subsequent search.

(2) Applicability

- Since processing to reuse free space involves overhead, use the free space reuse facility when storage efficiency matters more than performance.
- If you have an application that performs frequent deletions or insertions, such that the amount of data results in use of a large number of segments and frequent need for reorganization, and you would like to minimize the number of reorganizations, you should use the free space reuse facility. This subsection describes the application characteristics and the circumstances under which this facility is recommended.
 - **When there is no increase in data volume, including deletions (updates) and insertions**
If the maximum size of the data to be stored is specified with the free space reuse facility, the area from deleted data will later have priority for reuse. The application can then continue without having to add new area, so reorganization will not be necessary.

Example: Electronic administrative window

An application that receives data from an electronic window must be a 24-hour system. When an application is received, the data is inserted and later, once the storage period has passed, it is deleted. If the maximum segment size for the data to be received within the storage period is specified, then the space from deleted data can be reused. The application can then continue without having to add additional space. Reorganization then becomes unnecessary, the application will never need to stop, and it can provide 24-hour service.

- **When there is a steady increase in data volume, including deletions (updates) and insertions**
Steadily increasing data is stored not only in new space, but also in deleted space, increasing storage efficiency.

Example: Customer management

This application requires inserting new customer data, and deleting old customer data as it becomes no longer needed. Once the initial customer data has been entered in full, if the segment size is specified before starting a transaction to add or delete a customer, customer data added later will reuse the space from any deleted customer data.

- For insertion processing, performance is best if data is stored in unused pages and unused segments. Therefore, if the database reorganization utility (`pdorg`) can be executed quickly, the free space reuse facility would not be appropriate, and database reorganization would provide better performance.

14.5.4 Considerations

1. The free space reusage facility is effective when deletion processing ensures that there is always sufficient free space. If there is a search for space when there is not sufficient free space or when there is none at all, the search for free space will constitute a waste of time and resources. It is then necessary to specify more pages per segment, and the facility will have to be stopped. Because a change in the specification of the number of pages per segment requires re-creation (deletion or addition) of the RDAREA, you should consider carefully the number of segments and the segment size when you make your initial design.
 - In the following case, the number of segments can be omitted from the `SEGMENT REUSE` option:
There is one table in the RDAREA, no indexes are mixed in, and automatic extension is not specified
 - In the following cases, the number of segments cannot be omitted from the `SEGMENT REUSE` option:
 - There is one table in the RDAREA, no indexes are mixed in, and automatic extension is specified
 - There is one table in the RDAREA, with indexes mixed in
 - There are multiple tables in the RDAREA
 - If the amount of data will increase, specify the number of segments and specify the segment size to be large enough so that deletion will take place in each segment until it becomes full. If the amount of data will not increase, specify the number of segments by estimating the total number of segments that the table will need; there is no need to consider the segment size. However, within the same RDAREA, keep the total number of segments to be reused (if indexes are mixed in, the number of segments to be reused by the tables and the number of segments estimated for the indexes in the same RDAREA) to less than the total number of segments in the RDAREA.

If the free space reusage facility is used on tables for which automatic extension is specified, space extension has priority, and free space reusage is executed once the extended space has reached the specified number of segments.
2. If the free space reusage facility is enabled, it searches for free areas within used segments, which reduces the effectiveness of the following functions:
 - Storing data ordered by the cluster key in the cluster table
 - Storing data close to the original data when the row length is long during execution of an `UPDATE` statement (specifying `PCTFREE` in the `CREATE TABLE` statement, or specifying `tblfree` during data loading or table reorganization)

Therefore, we do not recommend that you use these functions when you apply the free space reusage facility.

14.5.5 Environment settings

The environment settings for use of the free space reusage facility are explained in this section.

1. Use the `pd_assurance_table_no` operand to specify the number of tables that will use the free space reusage facility.
For partitioning tables, calculate one table per partition. If the inner replica facility is being used, also calculate the table stored in the replica RDAREA as one table. For a HiRDB parallel server configuration, make the calculation separately for each back-end server, and specify the highest number in this operand.
The free space reusage facility can be used for tables defined by `CREATE TABLE` or modified by `ALTER TABLE` up to the number of times (number reserved) specified in the `pd_assurance_table_no` operand. If an insert is executed on a table for which the number reserved has been reached, the `KFPH22030-W` message is output, and the free space reusage facility is not applied. In such a case, the free space reusage facility will be applied for all defined tables if you increase the value of the `pd_assurance_table_no` operand. If the `ALTER TABLE` statement is specified with `ADD RDAREA` to add table storage RDAREAs such that the defined number exceeds the reserved number or the number defined for the HiRDB parallel server configuration exceeds the reserved number, free space reusage may or may not be applied to each RDAREA by partitioning tables for which free space reusage is defined.
2. Estimate the number of segments to be used for free space reusage (estimate the total number of segments from the total amount of data in the tables; see *16.1 Determining the size of a user RDAREA*), and specify the estimated number of segments in the `CREATE TABLE` definition SQL statement with the `SEGMENT REUSE` option specified. For tables already created, use the `ALTER TABLE` statement with the `SEGMENT REUSE` option specified. The number of segments specified here is applicable to all RDAREAs.

In addition, if you want to further increase storage efficiency, specify a reuse option value for `OPTION` in `SEGMENT REUSE`. By specifying a reuse option value, you can use the following features:

Reuse option value for each feature	Feature	Cases in which storage efficiency is expected to improve
1	UPDATE operation support	<ul style="list-style-type: none"> • UPDATE operations on fixed length data or ADT columns that contain NULL values • UPDATE operations on BINARY columns • UPDATE operations on VARCHAR, NVARCHAR, and MVARCHAR columns (with NO SPLIT specified) • The number of search mode switchovers is small compared to the number of segments already allocated (checkable using the <code>pddbst</code> command).
2	Improved storage efficiency of tables with many branch rows	Operations on tables that create branch rows
3	UPDATE operation support and improved storage efficiency for tables with many branch rows	<ul style="list-style-type: none"> • UPDATE operations on fixed length data or ADT columns that contain NULL values • UPDATE operations on BINARY columns • UPDATE operations on VARCHAR, NVARCHAR, and MVARCHAR columns (with NO SPLIT specified) • The number of search mode switchovers is small compared to the number of segments already allocated (checkable using the <code>pddbst</code> command). • Operations on tables that create branch rows

Note that if you apply `OPTION 1` (UPDATE support), page allocation at the time of the UPDATE operation proceeds from the last segment (in new page allocation mode), or from the segment at the start position of the previous search (in free page reuse mode). In this case, free space created during data loading or reorganization is no longer used preferentially. We therefore recommend, for tables in which `OPTION 1` is specified, that you specify `(0, 0)` for `PCTFREE` in the `CREATE TABLE` statement.

3. To change the number of segments once it has been specified, you can specify the number of segments again using `ALTER TABLE` with the `SEGMENT REUSE` option specified. HiRDB will process as follows, depending on the page search mode and the value specified for the number of segments:

- When in the new page allocate mode
If the specified number of segments is fewer than the number of used segments, free space reuse will be executed once all free space has disappeared from the last allocated segment.
- When in the free page reuse mode
If the number of segments specified is not greater than the number of used segments, nothing changes. If the number of segments specified is greater than the number of used segments, then once all free space has been used free space reuse will stop briefly, at which point new unused pages will be allocated.

4. If there is temporarily a large amount of addition due to such as batch processing, and you want to temporarily stop the free space reuse facility, specify `ALTER TABLE` with `SEGMENT REUSE NO` specified. When this is done, the free space reuse facility will stop immediately, and unused segments will be newly allocated.

5. To suppress the RDAREA segment usage notification messages (`KFPH00211-I` or `KFPA12300-I`) that are output when a table that uses the free space reuse facility secures segments, specify `N` in the `pd_rdarea_warning_point_msgout` operand.

If deletion (updating) and insertion are included and there is no increase in the amount of data, using the free space reuse facility removes the need to reorganize the table or expand RDAREAs. For this reason, the user also does not need to monitor the output of RDAREA segment usage notification messages. If deletion (updating) and insertion are included, there is no increase in the amount of data, and all the following conditions are met, output of RDAREA segment usage notification messages can be suppressed.

- Only tables that use the free space reuse facility are defined in the storage RDAREA.
- The table is a `FIX` attribute table or does not include variable length columns (that is, tables in which the data size does not increase).

However, in the following cases, the free space reuse facility might not run, so you must output and monitor the RDAREA segment usage notification messages. Then, you must take corrective action according to the RDAREA usage status.

- The number of tables that define the free space reuse facility exceeds the reserved number specified in the `pd_assurance_table_no` operand.
- Multiple tables that use the free space reuse facility are defined for the storage RDAREA, and the number of `SEGMENT REUSE` segments specified in the table definition is not equal to or greater than the maximum data size.

14.5.6 Checking execution status

You can check whether the free space reuse facility is effective from the items in the table below. This checking can be made with the database condition analysis utility, the statistics analysis utility, or the UAP statistical report facility. If this fails, the `KFP2031-W` message is output to the message log for each table (for the partitioning RDAREA in the case of a partitioned table). The items and their explanation follow:

Item	Explanation	Measure
Number of page search mode switchovers	This is the number of times the search mode switches from the new page allocate mode to the free page reuse mode, or vice versa. Frequent switchover during allocation of reused and unused segments means that there is more added space than available space due to deletions and that the segment size (number of pages) is too small.	Consider changing the segment size or the timing of deletion execution.
Number of failed page searches by the free space reuse facility	The number of used segments reaches the number of specified segments, but even upon switchover to the free page reuse mode, there are no free pages available. In such a case, the number of search failures increases. Because a search is performed even though there is no free space, pointless search processing is performed. If the number of failed page searches by the free space reuse facility and the number of page search mode switchovers both increase, then free page reuse is being executed when there is absolutely no free space.	Re-evaluate the specified number of segments and the segment size, or consider stopping the free space reuse facility.
Number of used segments	If there is no free space in the segments used by a table, unused segments will be allocated, causing the number of the table's used segments to increase. If the increase exactly matches the number of failed free space reuse page searches, search processing to find free space will continue even though there is none.	--

Legend:

--: Not applicable.

14.5.7 Notes

- In the following cases, the free space reuse facility will not function:
 - When data is stored using the hash facility for hash row partitioning
 - When data is stored in tables by means of data loading or by the database reorganization facility (`pdorg`)
 - When there is a user LOB RDAREA
- When the free space reuse facility is used, page search processing is slower when free space due to deletions is not contiguous than when free space due to deletions is contiguous. In such a case, consider stopping the free space reuse facility or consider reorganizing the data with the database reorganization utility (`pdorg`).

- In the case of non-FIX tables, even if the free space reuse facility is used, the number of segments used might increase. If you perform an update such as the following, the number of segments used might increase even if the volume of additions and deletions of data remains the same:

If you perform repeated deletions after inserting $\downarrow (page\ length - 48) \div (row\ length + 2) \downarrow$ data items and then updating data that would become longer than the row length (when NULL values are included) or data that would branch to another page

This case generates a full page, which you can release with the free page release utility (pdreclaim), in which case that page will no longer be able to store data. You can check whether such a page has been generated by using the database condition analysis utility (pddbst), in the value for Collect Prearranged Full Page.

Release the full page by executing one of the following:

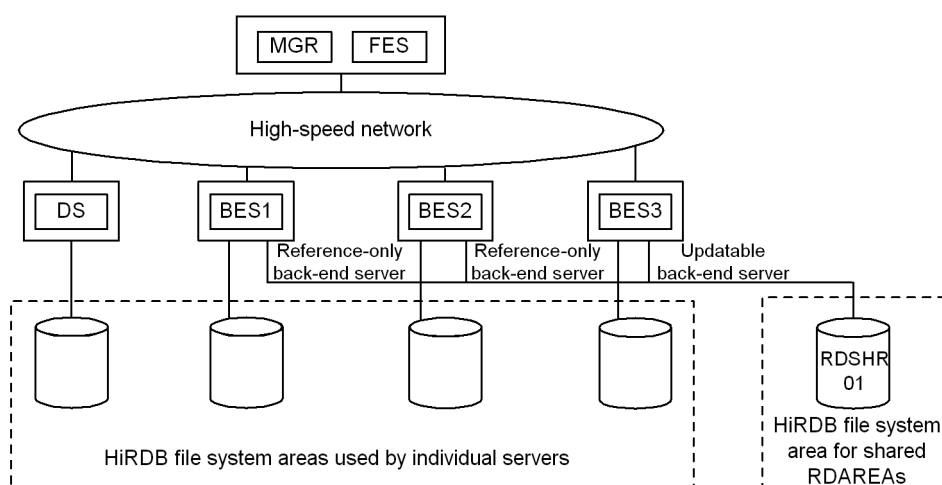
- The free page release utility (pdreclaim)
- The database reorganization utility (pdorg)
- Even when searching is executed in the free page reuse mode, deleted space cannot be reused within the same transaction.
- Even when the free space reuse facility is used, if you repeatedly perform UPDATE operations that change a non-NULL value to a NULL value, the number of segments allocated might exceed the actual data size. To prevent this, perform operations that use DELETE instead of performing an UPDATE operation on a NULL value.

14.6 Shared RDAREAs (HiRDB parallel server configuration only)

Normally, a back-end server can access only those RDAREAs that are located under that back-end server. By partitioning a table, parallel processing can be applied to table search or update operations, thereby improving the processing efficiency. In the case of a table that is heavily accessed by multiple transactions and that is difficult to partition, you can improve the efficiency of parallel processing by storing the table in a shared RDAREA. A *shared RDAREA* is a user RDAREA that can be accessed by all back-end servers. A table stored in a shared RDAREA is called a *shared table* and its index is called a *shared index*. Shared tables and indexes can be referenced by all back-end servers. Only shared tables and indexes can be stored in a shared RDAREA. The following figure provides an overview of a shared RDAREA.

Only a HiRDB parallel server configuration can define shared RDAREAs.

Figure 14–5: Overview of a shared RDAREA



Explanation:

The shared RDAREA RDSHR01 can be referenced by all back-end servers, BES1 to BES3. Note that only the updatable back-end server (BES3) can update the shared table; BES1 and BES2 are reference-only back-end servers.

(1) Effects

The efficiency of parallel processing improves because all back-end servers can access the shared RDAREA.

(2) Criteria

We recommend that you use shared RDAREAs in the following cases:

- A table is heavily accessed by multiple transactions, but it is difficult to partition the table.
- Complex search processing, such as join processing, is executed.

(3) Definition method

Specify a shared RDAREA as follows:

- Specify *Y* in the `pd_shared_rdarea_use` operand.
- Specify *SDB* in the `-k` option (purpose) of the `pdfmkfs` command and specify the name of a character special file. Also set the access path so that all back-end servers will use the same path name to access the shared RDAREA.

- Specify `shared` in the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`) to define a user RDAREA. Also specify an updatable back-end server in the `server name` operand. Any back-end server that is not specified in the `server name` operand becomes a reference-only back-end server.

Notes about definition

- You can define as many shared RDAREAs as the value specified in the `pd_max_rdarea_no` operand, which is the maximum number of RDAREAs. Note that the number of shared RDAREAs is added to the number of RDAREAs for each back-end server.
- A shared RDAREA cannot be defined in the HiRDB file system area for a back-end server that is not an updatable back-end server for the shared RDAREA.
- A shared RDAREA is defined in the HiRDB file system area for shared RDAREAs. To define a shared RDAREA, specify `SDB` in the `pdfmkfs -k` command. Only shared RDAREAs can be defined in a HiRDB file system area for shared RDAREAs.

The following shows an example of a control statement of the database structure modification utility (`pdmod`) for the shared RDAREA shown in *Figure 14-14 Overview of a shared RDAREA*:

```
create shared rdarea RDSHR01 globalbuffer buf01 for user used by PUBLIC
server name BES3      ...Specification of updatable back-end server
open attribute INITIAL
page 4096 characters
storage control segment 20 pages
file name "/HiRDB/DATABASE/SHR1/rdshr01_f01"    ...file name
initial 10000 segments;
```

(4) Updating a shared RDAREA

To update a shared RDAREA, you must specify `IN EXCLUSIVE MODE` in the `LOCK` statement to lock the shared RDAREA for all back-end servers. In the case of an `UPDATE` statement that does not change index key values, there is no need to issue the `LOCK` statement. For details about updating shared tables, see *12.18.3 Manipulating shared tables*. Updates to a shared table or shared index are written to the disk when the `COMMIT` statement is issued.

(5) Managing the shutdown status of a shared RDAREA

Accesses to a shared RDAREA are managed independently by each back-end server. Therefore, in the event of an error, the shutdown status may vary from one back-end server to another. When you execute the database structure modification utility (`pdmod`) or database recovery utility (`pdrstr`), you must use the `pdhold` command to match the shutdown status of the shared RDAREA among all back-end servers. You can use the `pddb1s -m` command to display the status of the shared RDAREA for all the back-end servers.

(6) Setting system switchover

To use a shared RDAREA, you must activate the shared disks that store the shared RDAREAs for all units in which a back-end server is installed. Therefore, in cases in which both an updatable back-end server and reference-only back-end server are installed on the same host, if the updatable back-end server performs system switchover to switch the shared disks, the reference-only back-end server is no longer able to reference the shared RDAREAs. For that reason, to perform system switchover for units on which a back-end server is installed, the method used to set system switchover differs depending on the system switchover system configuration. For details about how to set up system switchover, see the *HiRDB Version 9 System Operation Guide*.

(a) When the standby system switchover configuration is used

- For a 1-to-1 system switchover configuration:
Since the updatable back-end server and reference-only back-end server do not exist on the same host, no special settings are required for a 1-to-1 system switchover configuration. For details about how to set up system switchover, see *Shared disk access control by the cluster software* in the *HiRDB Version 9 System Operation Guide*.
- For a mutual system switchover or 2-to-1 system switchover configuration:

For a mutual system switchover configuration, no special settings are required unless system switchover is performed for systems other than the updatable back-end server and reference-only back-end server (for example, the system manager, front-end server, or dictionary server). For details about how to set up system switchover, see *Shared disk access control by the cluster software* in the *HiRDB Version 9 System Operation Guide*.

For mutual system switchover between an updatable back-end server and reference-only back-end server, switching of shared disks cannot be performed. For details about how to set up system switchover, see *Shared disk access control by HiRDB* in the *HiRDB Version 9 System Operation Guide*.

(b) When a 1-to-1 standby-less system switchover configuration is used

When a 1-to-1 standby-less system switchover configuration is used with an updatable back-end server, the system is switched to the reference-only back-end server, but the shared disks cannot be switched. For details about how to set the system switchover, see *Shared disk access control by HiRDB* in the *HiRDB Version 9 System Operation Guide*.

(c) When the standby-less system switchover configuration (effects distributed) is used

When a 1-to-1 standby-less system switchover configuration is used with an updatable back-end server, the system is switched to the reference-only back-end server, but the shared disks cannot be switched. For details about how to set system switchover, see *Shared disk access control by HiRDB* in the *HiRDB Version 9 System Operation Guide*.

Note:

Make sure that, for the shared RDAREAs, no reference-only back-end server uses shared disks as a cluster software management resource.

(7) Executing utilities and operation commands on a shared RDAREA

When a utility or operation command is used to process a shared RDAREA, shared table, or shared index, HiRDB may internally issue `LOCK TABLE` to lock the shared RDAREA for all the back-end servers. If an application is accessing a table or index in the shared RDAREA, deadlock or server-to-server global deadlock may occur. When you execute a utility or operation command, make sure that you place the target shared RDAREA in command shutdown status.

(8) Notes about using shared RDAREAs

1. If you use the system switchover facility, place the unit containing the updatable back-end server as follows:

- Place it on a separate host from a reference-only back-end server.
- Place the target system in such a manner that the updatable back-end server does not coexist with a reference-only back-end server on the same host when system switchover occurs.

Make sure that no reference-only back-end server uses the disk volume for the shared RDAREA as the cluster software's management resources.

2. No floating servers can be installed because a shared RDAREA is placed in all back-end servers.

3. A shared table cannot be the replication target.

4. Updatable online reorganization of shared RDAREAs cannot be executed for each server (`-s` option). However, if some of the back-end servers are inactive and cannot be started immediately, the following actions can be taken for each server:

- Forcibly cancel online reorganization hold (committing the database for online reorganization) only by the active back-end servers (`pdorbegin -u`).
- Forcibly cancel reflection processing for updatable online reorganization only by the active back-end servers (`pdorend -u`).

After the inactive back-end servers have started, place them in the same status as the other back-end servers.

5. When you use local buffers to update a table and index in a shared RDAREA, first issue the `LOCK TABLE` statement. If you update without issuing a `LOCK TABLE` statement, updated pages that are recovered using the global buffer might not always be retained when the server process terminates abnormally and the transaction recovery process runs. If the updated pages cannot be retained, they cannot be restored; therefore, the unit terminates abnormally with abort code `Phb3008`. If this happens, restart HiRDB.

14.7 Temporary table RDAREAs

A temporary table RDAREA is a user RDAREA for storing temporary tables and temporary table indexes.

(1) Criteria

Temporary table RDAREAs are required in order to use temporary tables.

(2) Attributes of temporary table RDAREAs

There are two types of temporary table RDAREAs:

- Temporary table RDAREAs with the SQL session shared attribute
This type of temporary table RDAREA can be used in any SQL session. Use this type to reduce the number of temporary table RDAREAs because such RDAREAs can be shared among SQL sessions.
- Temporary table RDAREAs with the SQL session-specific attribute
This type of temporary table RDAREA can be used only in specific SQL sessions (for which the temporary table RDAREA that is to be used is specified in `PDTMPTBLRDAREA` in the client environment definition). When a particular user handles a large amount of data, this type is used to prevent a shortage of space from occurring for the temporary table RDAREAs that are used by other users.

(3) How to create temporary table RDAREAs

The following describes how to create temporary table RDAREAs:

1. In the `pd_max_temporary_object_no` operand, specify the maximum number of temporary tables and temporary table indexes that can be used at the same time. If you are applying 0904 compatibility mode, in addition to specifying the `pd_max_temporary_object_no` operand, specify the maximum number of temporary table RDAREAs in the `pd_max_tmp_table_rdarea_no` operand.
2. Specify DB in the `-k` option of the `pdfmkfs` command to create a HiRDB file system area.
3. Create a user RDAREA by specifying the following two operands in the `create rdarea` statement in the database initialization utility (`pdinit`) or the database structure modification utility (`pdmod`):
 - Specify `for user used by PUBLIC`.
To use an RDAREA as a temporary table RDAREA, it must be a public user RDAREA.
 - Specify `temporary table use`.
If the temporary table RDAREA is to have the SQL session shared attribute, specify `temporary table use shared`. If it is to have the SQL session-specific attribute, specify `temporary table use occupied`.

Notes on creation

- If you are applying 0904 compatibility mode, make sure that the value of the `pd_max_tmp_table_rdarea_no` operand is smaller than the value of the `pd_max_rdarea_no` operand. When you add temporary table RDAREAs, make sure that the total number of RDAREAs, including the temporary table RDAREAs being added, does not exceed the maximum number of RDAREAs specified in the `pd_max_rdarea_no` operand. If the total number will exceed the set maximum, change the `pd_max_rdarea_no` operand's value.
- If there are multiple available temporary table RDAREAs, HiRDB determines automatically the temporary table RDAREA to be used. Therefore, we recommend that you set all temporary table RDAREAs that can be used by the same UAP to have the same RDAREA size, segment size, and page size.
If the temporary table RDAREAs have different RDAREA sizes, segment sizes, or page sizes, the following problems might arise:
 - Each time a temporary table is instantiated by the first `INSERT` statement executed, an RDAREA with a different page size might be chosen for the storage. In such a case, whether the first `INSERT` statement will execute will depend on the relationship between the row length of the data to be inserted and the page size of the storage RDAREA.

- HiRDB chooses an RDAREA with the largest number of segments as the storage RDAREA. Therefore, if the storage candidate RDAREAs have different page or segment sizes, HiRDB might choose an RDAREA with a smaller amount of free space as the storage RDAREA.
- A temporary table RDAREA is initialized when HiRDB starts or when the first INSERT statement is executed on its temporary table. Therefore, if its size is large, the amount of overhead increases. For details about initialization of temporary table RDAREAs, see (4) *Initializing the temporary table RDAREAs*.

Example

This example creates a temporary table RDAREA (RDTMP01) with the SQL session shared attribute on BES1 of a HiRDB parallel server configuration. The following shows the control statement in the database structure modification utility (pdmod).

```
create rdarea RDTMP01 Specifies the name of the temporary table RDAREA
globalbuffer tmpbuf01
for user used by PUBLIC Specifies a public user RDAREA
server name BES1
open attribute INITIAL
page 4096 characters
storage control segment 100 pages
temporary table use shared Specifies the SQL-session shared attribute
file name "/hirdb/db/rdtmp01_f01"
initial 500 segments ;
```

(4) Initializing the temporary table RDAREAs

When HiRDB starts, it initializes the temporary table RDAREAs that were used during the previous session, regardless of the start mode. Therefore, if a temporary table RDAREA to be initialized is large in size, it will take time for HiRDB to start. If you use the rapid system switchover facility or the standby-less system switchover facility and need to reduce the time required for system switchover, you can skip initialization of temporary table RDAREAs during HiRDB startup by specifying ACCESS in the `pd_tmp_table_initialize_timing` operand. Note that when ACCESS is specified, HiRDB initializes a temporary table RDAREA when the first INSERT statement is executed on a temporary table in the RDAREA. Therefore, if the size to be initialized is large, the overhead for executing the INSERT statement becomes large. To reduce the overhead, reduce the size of the temporary table RDAREA.

For details about the `pd_tmp_table_initialize_timing` operand and how to estimate the overhead for initializing temporary table RDAREAs, see the manual *HiRDB Version 9 System Definition*.

(5) Backing up temporary table RDAREAs

There is no need to back up a temporary table RDAREA because the data in temporary tables is retained only during the transaction or SQL session. If an error occurs in a temporary table RDAREA, re-initialize the RDAREA with the `initialize rdarea` statement of the database structure modification utility (pdmod).

(6) Notes about using temporary table RDAREAs

(a) Limitations

Some limitations apply when the operation commands and utilities listed below are executed on a temporary table RDAREA. For details, see the manual *HiRDB Version 9 Command Reference*.

- `pdhold` command
- `pdorcheck` command
- `pdorcreate` command
- `pdrdrefls` command
- Database copy utility (`pdcopy`)
- Database condition analysis utility (`pddbstd`)
- Database structure modification utility (`pdmod`)
This utility does not support moving RDAREAs or defining replicas for RDAREAs.
- Database recovery utility (`pdrstr`)

(b) Using a temporary table RDAREA that has the SQL session-specific attribute

A temporary table RDAREA that has the SQL session-specific attribute can be used only by the SQL sessions for which the RDAREA is specified in `PDTMPTBLRDAREA` in the client environment definition. To use such an RDAREA more exclusively (so that only one SQL session can use it), allocate a local buffer. The corresponding temporary table RDAREA will be locked in lock mode (EX).

(c) Creating a non-FIX temporary table

To create a non-FIX temporary table, the temporary table RDAREA must satisfy the applicable condition listed below. If the temporary table RDAREA does not satisfy the applicable condition below, the `KFPA11809-I` message might be issued when data is stored.

- When `PDTMPTBLRDAREA` is specified in the client environment definition

The same page length[#] is defined for all temporary table RDAREAs that are specified in `PDTMPTBLRDAREA`.

- When `PDTMPTBLRDAREA` is not specified in the client environment definition or an XDS client is used

The same page length[#] is defined for all temporary table RDAREAs with the SQL session shared attribute.

#

The page length is greater than the basic row length. To determine the basic row length, use the formula for determining the data length in *Predefined-type data lengths* in the manual *HiRDB Version 9 SQL Reference*. If the `KFPA11809-I` message is issued, specify a value that is greater than the *row length to be stored* shown in the `KFPA11809-I` message.

(d) Linking with HiRDB Datareplicator

An update operation on a temporary table is not subject to extraction. Therefore, there are no considerations related specifically to temporary tables when HiRDB Datareplicator is used.

15 Storage Requirements for HiRDB

This chapter explains the HiRDB storage requirements.

15.1 Estimating the memory size required for a HiRDB single server configuration

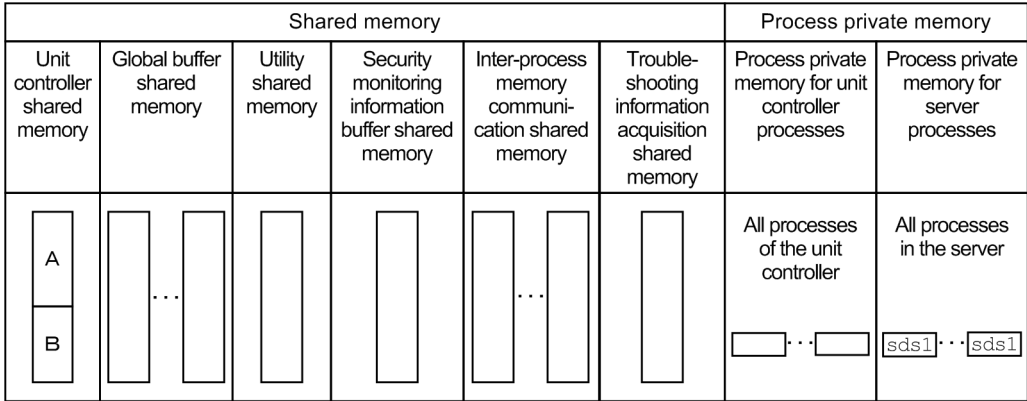
This section explains how to estimate the size of the memory required for a HiRDB single server configuration. The topics covered include:

- Memory allocation
- Calculation of required memory
- Formulas for shared memory used by a unit controller
- Formulas for shared memory used by a single server
- Formulas for size of shared memory used by global buffers
- Formulas for size of memory required during SQL execution
- Formula for size of memory required during SQL preprocessing
- Formula for size of memory required during BLOB-type data retrieval or updating
- Formula for size of memory required during block transfer or array FETCH

15.1.1 Memory allocation

The following figure shows the memory allocation for a HiRDB single server configuration.

Figure 15–1: Memory allocation for a HiRDB single server configuration



Legend:

- A: Part used by a unit controller process
- B: Part used by a Single Server process
- sds: Single Server

The following table lists details for HiRDB single server configuration shared memory.

Table 15–1: HiRDB single server configuration shared memory details

Item	Type of shared memory					
	Unit controller shared memory	Global buffer shared memory	Utility shared memory	Security monitoring information buffer shared memory	Inter-process memory communication shared memory	Troubleshooting information acquisition shared memory
Purpose	System control	Global buffers	Communication between the unit controller and utilities	Security monitoring information buffer	Client-server inter-process memory communication	Acquisition of troubleshooting information
Processes	All HiRDB processes	Single server	Utility processes	Single server	Single server, client processes	All HiRDB processes
Number of segments	1	<ul style="list-style-type: none"> If the facility for dynamic updating of global buffers is not used: 1 to 512 If the facility for dynamic updating of global buffers is used: <ul style="list-style-type: none"> 32-bit mode: 1 to 1,012 64-bit mode: 1 to 1,512 	1	1	Number of clients connected using the PDIPC=MEMORY environment variable (0-2,000) × 2	1
Maximum value per segment	See Table 15-2 <i>Size of memory required for a HiRDB single server configuration</i> . Value of the shmmax operating system parameter must be equal to or greater than the calculation value.	Divide the segment by the SHMMAX operand value. Value of the shmmax operating system parameter must be equal to or greater than the SHMMAX operand value.	See Table 15-2 <i>Size of memory required for a HiRDB single server configuration</i> . Value of the shmmax operating system parameter must be equal to or greater than the calculation value.	See Table 15-2 <i>Size of memory required for a HiRDB single server configuration</i> . Value of the shmmax operating system parameter must be equal to or greater than the calculation value.	See Table 15-2 <i>Size of memory required for a HiRDB single server configuration</i> . Value of the shmmax operating system parameter must be equal to or greater than the calculation value.	10 MB
Allocation conditions	None	There must be a global buffer definition	Specify pd_utl_exec_mode=1	Specify the pd_aud_file_name operand as the HiRDB file system area name for the audit trail file.	There are clients connected using the PDIPC=MEMORY environment variable	None
Creation timing	At unit activation (including standby unit activation when	<ul style="list-style-type: none"> At server activation (including standby unit activation 	When utilities are executed	When a HiRDB single server configuration starts up	When client and server are connected	At unit activation (including standby unit activation

Item	Type of shared memory					
	Unit controller shared memory	Global buffer shared memory	Utility shared memory	Security monitoring information buffer shared memory	Inter-process memory communication shared memory	Troubleshooting information acquisition shared memory
	user server hot standby or the rapid system switchover facility is used)	when the rapid system switchover facility is used) <ul style="list-style-type: none"> When <code>pdbufmod - k {add upd}</code> is executed 				when user server hot standby or the rapid system switchover facility is used)
Deletion timing	At next unit activation (including standby unit activation when user server hot standby or the rapid system switchover facility is used)	<ul style="list-style-type: none"> When <code>pdbufmod - k del</code> is executed For normal termination or planned termination: When the server is terminated For forced termination, abnormal termination, or termination of standby unit when the rapid system switchover facility is used: When the unit is next activated 	10 minutes after the utility terminates	When a HiRDB Single server quits	When client and server are disconnected	At unit termination
Indication by <code>pdls -d mem</code>	Indicated	Indicated	Indicated	Indicated	Not indicated	Not indicated
SHM-OWNER of <code>pdls -d mem</code>	MANAGER	Server name	UTILITY	AUDEF	Not indicated	Not indicated
Related operands	<ul style="list-style-type: none"> <code>pd_shmpool_attribute</code> <code>pd_sds_shmpool_size</code> 	<ul style="list-style-type: none"> <code>pd_dbbuff_attribute</code> <code>pd_dbbuff_modify</code> <code>pdbuffer</code> SHMMAX 	<ul style="list-style-type: none"> <code>pd_utl_exec_mode</code> 	<ul style="list-style-type: none"> Operands related to the security audit facility[#] 	<ul style="list-style-type: none"> PDIPC PDSENDME SIZE PDRECVM SIZE 	None
Remarks	--	--	Can be created only when <code>pd_utl_exec_mode=1</code> (when <code>pd_utl_exec_</code>	--	--	--

Item	Type of shared memory					
	Unit controller shared memory	Global buffer shared memory	Utility shared memory	Security monitoring information buffer shared memory	Inter-process memory communication shared memory	Troubleshooting information acquisition shared memory
			mode=0, the relevant space is allocated in the unit controller shared memory).			

Legend:

--: Not applicable.

#

For details, see the manual *HiRDB Version 9 System Definition*.

15.1.2 Calculation of required memory

The required memory used by a HiRDB single server configuration is the sum of all the terms shown in the following table.

For details about the value specified in the `shmmax` operating system parameter (for Solaris, `shmsys:shminfo_shmmax`; for Linux, `SHMMAX`), see *Chapter 20. Specifying OS Parameters*.

If you increase the size of shared memory, this might affect transaction performance by increasing the number of page faults. For each operand, specify the most appropriate value based on the guidelines for your particular system.

Table 15–2: Size of memory required for a HiRDB single server configuration

Item	Required memory (KB)
Process private area	<p>Process private area used by all unit controller processes</p> <p>■ 32-bit mode</p> $E + \uparrow \{(64 + 48 \times (u + 1)) \times (\text{value of } pd_max_server_process - b - 6) + (64 + \{48 \times (y + 1)\} \times 3 + w + R) \} \div 1,024 \uparrow$ <p>■ 64-bit mode</p> $E + \uparrow \{(64 + 64 \times (u + 1)) \times (\text{value of } pd_max_server_process - b - 6) + (64 + 64 \times (y + 1)) \times 3 + w\} \div 1,024 \uparrow + R$ <p>■ If you are using plug-ins, add:</p> $+ 1,400$ <p>■ If you are using the asynchronous READ facility, add:</p> $+ r$ <p>■ If you are using Real Time SAN Replication, add:</p> $+ \uparrow 425 \times (2 \times b + 7) \div 1,024 \uparrow$ <p>■ If fixed was specified in the <code>pd_process_terminator</code> operand, add:</p> $+ F \times (\text{value of } pd_process_terminator_max - 1)$ <p>■ If you are performing in-memory data processing, add:</p> $+ \uparrow \{K \times (\text{value of } pd_max_users \times 2 + 7)\} \div 1,024 \uparrow$

Item		Required memory (KB)
		<p>■ If you are changing the maximum number of communication traces stored, add:</p> <p>+ $\uparrow M \div 1,024 \uparrow$</p>
	Process private area used by single server process ^{#1}	<p>pd_work_buff_mode=each specified</p> <p>■ 32-bit mode</p> $\{G + g + (a + 9) \times c + h + i + m + p + q + s\} \times (b + 3) + \uparrow (64 + 48 \times (v + 1)) \div 1,024 \uparrow \times (b + 3) + J$ <p>■ 64-bit mode</p> $\{G + g + (a + 9) \times c + h + i + m + p + q + s\} \times (b + 3) + \uparrow (64 + 64 \times (v + 1)) \div 1,024 \uparrow \times (b + 3) + J$ <p>■ If you are performing in-memory data processing, add:</p> <p>+ $\uparrow \{K \times (b + 3)\} \div 1,024 \uparrow$</p> <p>■ If you are changing the maximum number of communication traces stored, add:</p> <p>+ $\uparrow P \div 1,024 \uparrow$</p>
		<p>pd_work_buff_mode=pool specified or omitted</p> <p>■ 32-bit mode</p> $(G + g + a + \uparrow a \div 128 \times 0.1 \uparrow + 11 + h + i + m + n + p + q + s) \times (b + 3) + \uparrow (64 + 48 \times (v + 1)) \div 1,024 \uparrow \times (b + 3) + J$ <p>■ 64-bit mode</p> $(G + g + a + \uparrow a \div 128 \times 0.1 \uparrow + 15 + h + i + m + n + p + q + s) \times (b + 3) + \uparrow (64 + 64 \times (v + 1)) \div 1,024 \uparrow \times (b + 3) + J$ <p>■ If you are performing in-memory data processing, add:</p> <p>+ $\uparrow \{K \times (b + 3)\} \div 1,024 \uparrow$</p> <p>■ If you are changing the maximum number of communication traces stored, add:</p> <p>+ $\uparrow P \div 1,024 \uparrow$</p>
Shared memory	Space used by the unit controller in the unit controller shared memory	$\uparrow d \div 1,024 \uparrow$
	Space used by the single server in the unit controller shared memory	E
	Global buffer shared memory	F
	In-memory data processing shared memory	L
	Utility shared memory	T
	Security audit information buffer shared memory	<p>■ For automatic calculation by the system:</p> $\uparrow 0.3 + \text{MAX}\{(H + 100), (H \times 1.2)\} \times 0.25 \uparrow$ <p>■ For user -specified values (specify the pd_audit_def_buffer_size operand):</p> <p>Value specified for pd_audit_def_buffer_size</p>
	Inter-process memory communication shared memory ^{#2}	$j \times k$

#1: If you are using plug-ins, add 300 per single server process.

#2: Add this value if you have specified `PDIPC=MEMORY` in the client environment definition. For details about the inter-process memory communication facility and client environment definitions, see the *HiRDB Version 9 UAP Development Guide*. If either the HiRDB server or the HiRDB client is in 32-bit mode, the system allocates the shared memory for the inter-process memory communication facility in the 32-bit address space.

a: Value of `pd_work_buff_size` operand

b: Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand

c: Maximum number of work tables

Obtain the number of work tables for each SQL statement from *Table 15-3 Procedure for obtaining the number of work tables for each SQL statement*. Use the largest value obtained from *Table 15-3 Procedure for obtaining the number of work tables for each SQL statement* as the maximum number of work tables.

d: Value obtained from *15.1.3 Formulas for shared memory used by a unit controller*.

e: Value obtained from *15.1.4 Formulas for shared memory used by a single server*.

f: Value obtained from *15.1.5 Formula for size of shared memory used by global buffers*.

g: Size of memory required during SQL execution

For details about the formula, see *15.1.6 Formulas for size of memory required during SQL execution*.

h: Size of memory required during SQL preprocessing

For details about the formula, see *15.1.7 Formula for size of memory required during SQL preprocessing*.

i: LOB buffer batch input/output work memory

Add 62 KB if a LOB global buffer is specified in the global buffer definition (`-b` specified in the `pdbuffer` operand of the system common definition).

j: Maximum number of concurrently executable clients that use the inter-process memory communication facility.

If you are not sure about the value, specify the number of all clients that use the inter-process memory communication facility or the value of the `pd_max_users` operand.

k: Average memory size for data transfer performed by all clients that use the inter-process memory communication facility (*value of `PDSSENDMEMSIZE` + value of `PDRECVMEMSIZE` in the client environment definition*).

m: Memory requirement for Java virtual machine

If you use Java stored procedures or Java stored functions, add the size of memory used by the Java virtual machine. This value depends on the Java virtual machine's options (`-Xms`, `-Xmx`, and `-Xmn` options for Hewlett-Packard JRE 1.2.2.04) and version. For details about the memory requirement for your Java virtual machine, see the applicable manual. Following are the guidelines for the memory required for HP-UX:

- Eight MB of memory is required to start a Java virtual machine.
- Add the maximum memory size for the Java virtual machine (value of the `-Xmx` option). Note that some Java virtual machines may use more memory than the size specified in the `-Xmx` option.

n: Work table extended memory size

When the `pd_work_buff_expand_limit` operand is specified, add the work table extended memory size.

The work table extended memory size is determined from the following formula:

Work table extended memory size (kilobytes) = work table extended buffer size + \uparrow (work table extended buffer size \div 128) \times 0.1 \uparrow

- Work table extended buffer size (kilobytes) = MAX(0, work table extended buffer size based on hash join, subquery hash execution) + MAX(0, work table extended buffer size based on the increase in the number of work tables)
- Work table extended buffer size based on hash join, subquery hash execution = MIN { (work table extended buffer size based on hash join, subquery hash execution - value of the `pd_work_buff_size` operand), (value of the `pd_work_buff_expand_limit` operand - value of the `pd_work_buff_size` operand) } \times number of concurrently executing users executing hash join, subquery hash execution

For details about determining the work table extended buffer size when executing hash join, subquery hash execution, see the *HiRDB Version 9 UAP Development Guide*.

- Work table extended buffer size based on the increase in the number of work tables = MIN { (number of work tables used \times 128 - value of the `pd_work_buff_size` operand), (value of `pd_work_buff_expand_limit` operand

- value of `pd_work_buff_size operand` } \times (number of users such that the number of work tables is greater than the value of the `pd_work_buff_size operand` \div 128)

Number of work tables used = MAX(number of work table files used per SQL statement, number of work table files used by the `ASSIGN LIST` statement)

For details about determining the number of work table files used per SQL statement and the number of work table files used by the `ASSIGN LIST` statement, see *18.3 Determining the maximum number of files (pdfmkfs -l command)*.

p: Memory requirements required for BLOB data type

For details about the formula, see *15.1.8 Formula for size of memory required during BLOB data retrieval or updating (HiRDB single server configuration)*.

q: Memory requirements required for server-side block transfer or array `FETCH`

For details about the formula, see *15.1.9 Formula for size of memory required during block transfer or array FETCH*.

r: Memory size used by asynchronous `READ`

This is applicable when the asynchronous `READ` facility is used; use the following formula (in kilobytes) for the calculation:

$$\begin{aligned} & (90 + \\ & 90 \\ & \sum_{i=1} \text{Memory used by the RDAREA for management of the HiRDB file system area}) \\ & \times \text{value of } pd_max_ard_process \end{aligned}$$

For the memory used by the RDAREA for management of the HiRDB file system area, use 90 areas as the maximum in the calculation. If the number of areas used by the server is fewer than 90, assume that amount anyway.

The memory used by the RDAREA for management of the HiRDB file system area (in kilobytes) is calculated from the formula below based on the initial settings:

Note that the parameters at the time of initialization of the area can be checked by running the `pdfstatfs` command with the `-A` option specified.

$$\{(\text{Number of files}^{\#1} + \text{number of extensions}^{\#2}) \div 64\} \times 1.5^{\#3}$$

#1: Value specified by `pdfmkfs -l`, or displayed in available file count in the execution results of the `pdfstatfs` command.

#2: Value specified by `pdfmkfs -e`, or displayed in available expand count in the execution results of the `pdfstatfs` command.

#3: Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2,048.

s: HiRDB file system memory size

Determine with the following formula (in kilobytes):

$$\begin{aligned} & 347 + \text{Memory used by the work tables for management of the HiRDB file system area} + \text{Memory used by the system logs for management} \\ & \text{of the HiRDB file system area} + \\ & 90 \\ & \sum_{i=1} \text{Memory used by the RDAREA for management of the HiRDB file system area} \end{aligned}$$

The memory used by the HiRDB file system area for management of work tables and system logs uses the maximum value calculated for the memory used by the HiRDB file system area for management used by the server. For RDAREAs, use 90 areas as the maximum calculation value. If the number of areas used by the server is fewer than 90, assume that amount anyway.

The memory used by the RDAREA for management of the HiRDB file system area (in kilobytes) is calculated with the formula below based on the initial settings:

Note that the parameters at the time of initialization of the area can be checked by running the `pdfstatfs` command with the `-A` option specified.

$$\{(Number\ of\ files^{#1} + number\ of\ extensions^{#2}) \div 64\} \times 1.5^{#3}$$

#1: Value specified by `pdfmkfs -l`, or displayed in available file count in the execution results of the `pdfstatfs` command.

#2: Value specified by `pdfmkfs -e`, or displayed in available expand count in the execution results of the `pdfstatfs` command.

#3: Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2,048.

t: When value of `pd_utl_exec_mode` is 0: 0

When value of `pd_utl_exec_mode` is 1: $\uparrow \{(b \times 2,000 + 136) \div 1,024\} \uparrow \times 1,024$

u: Valid value of `pd_module_trace_max` for the unit control information definition

v: Valid value of `pd_module_trace_max` for the single server definition

w: Memory size for restarting HiRDB

If this memory size cannot be allocated, HiRDB restart fails. Use the formula below to determine the size (in bytes):

y: If the operand `pd_module_trace_max` is specified in the system common definition or the unit control information definition: the value of `pd_module_trace_max`

If not: 16,383

A + *B*

- If `commit` is specified in the `pd_dbsync_point` operand, add:

$$+ 112 \times ((value\ of\ pd_max_users + value\ of\ pd_max_reflect_process_count) \times 2 + 7)$$

- If 1 or a greater value is specified in the `pd_inner_replica_control` operand, add:

+ *C*

- If the number of HiRDB file system areas that store RDAREAs created in a character special file is 1,001 or more, add:

+ *D*

Use the following variables in the formula to calculate the size of memory used by HiRDB to restart:

Variable	Value
<i>A</i>	<ul style="list-style-type: none"> • 32-bit mode $246,762 + 4 \times value\ of\ pd_max_rdarea_no$ $+ \{48 \times (value\ of\ pd_max_rdarea_no + number\ of\ tables) + 304\} \times ((value\ of\ pd_max_users + value\ of\ pd_max_reflect_process_count) \times 2 + 7)$ <ul style="list-style-type: none"> • 64-bit mode $305,274 + 8 \times value\ of\ pd_max_rdarea_no$ $+ \{64 \times (value\ of\ pd_max_rdarea_no + number\ of\ tables) + 512\} \times ((value\ of\ pd_max_users + value\ of\ pd_max_reflect_process_count) \times 2 + 7)$ <p>Number of tables: $62 + MAX\ \{value\ of\ pd_max_access_tables, 500\}$</p>
<i>B</i>	$b1 \times X + b2 \times Y$ <p><i>b1</i>: When the record length of the server status file < 4,096</p> $MAX((\downarrow(3,400 \div ((\downarrow((record\ length - 40) - 308) \div 20) \downarrow))$ $+ (\downarrow((record\ length - 40) \div 20) \downarrow) \times (MAX(\downarrow(4,096 \div record\ length) \downarrow, 2) - 1))$ $+ 0.7) \downarrow, 1) \times MAX(\downarrow(4,096 \div record\ length) \downarrow, 2) \times (record\ length - 40)$ <p>When $4,096 \leq$ record length of server status file < 12,288</p> $MAX(\downarrow(3,400 \div (\downarrow(((record\ length - 40) - 308) \div 20) \downarrow) + 0.7) \downarrow, 1)$ $\times (record\ length - 40)$ <p>When $12,288 \leq$ record length of server status file</p>

Variable	Value
	$\text{MAX}(\downarrow(3,400 \div (\downarrow(((\text{record length} - 40) - 836) \div 20)\downarrow) + 0.7)\downarrow, 1)$ $\times (\text{record length} - 40)$ <p>X: When the number of RDAREAs $\leq 3,400$: 1 When $3,401 \leq \text{number of RDAREAs} \leq 6,800$: 2 When $6,801 \leq \text{number of RDAREAs}$: 3</p> <p>b2: When the record length of the server status file $< 4,096$ $(\downarrow(5,662,310 \div ((\downarrow((\text{record length} - 40) - 308) \div 20)\downarrow) + (\downarrow(\text{record length} - 40) \div 20)\downarrow) \times (\text{MAX}(\downarrow(4,096 \div \text{record length}\downarrow, 2) - 1))$ $+ 0.7)\downarrow) \times \text{MAX}(\downarrow(4,096 \div \text{record length}\downarrow, 2) \times (\text{record length} - 40)$ <p>When $4,096 \leq \text{record length of server status file} < 12,288$ $\downarrow(5,662,310 \div (\downarrow(((\text{record length} - 40) - 308) \div 20)\downarrow) + 0.7)\downarrow$ $\times (\text{record length} - 40)$ <p>When $12,288 \leq \text{record length of server status file}$ $\downarrow(5,662,310 \div (\downarrow(((\text{record length} - 40) - 836) \div 20)\downarrow) + 0.7)\downarrow$ $\times (\text{record length} - 40)$ <p>Y: When the number of RDAREAs $\leq 10,200$: 0 When $10,201 \leq \text{number of RDAREAs} \leq 5,672,510$: 1 When $5,672,511 \leq \text{number of RDAREAs} \leq 11,334,820$: 2 When $11,334,821 \leq \text{number of RDAREAs}$: 3</p> </p></p></p>
C	<ul style="list-style-type: none"> 32-bit mode $(48 \times \text{value of } pd_max_rdarea_no + 80) \times ((\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$ <ul style="list-style-type: none"> 64-bit mode $(64 \times \text{value of } pd_max_rdarea_no + 160) \times ((\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$
D	<ul style="list-style-type: none"> 32-bit mode $12,012 \times (\uparrow(\text{number of HiRDB file system areas that store RDAREAs created in a character special file} - 1,000) \div 1,000 \uparrow)$ <ul style="list-style-type: none"> 64-bit mode $16,016 \times (\uparrow(\text{number of HiRDB file system areas that store RDAREAs created in a character special file} - 1,000) \div 1,000 \uparrow)$

E, F, G: Fixed value

These values depend on the OS being used. The following table presents the values for each OS (in kilobytes):

OS	Value of E	Value of F	Value of G
HP-UX (32-bit mode)	90,500	2,800	5,600
HP-UX (32-bit-mode POSIX library version)	130,000	2,800	4,900
HP-UX (64-bit mode)	92,100	2,900	6,600
HP-UX (IPF)	211,000	2,600	7,500
AIX (32-bit mode)	75,400	2,100	5,800
AIX (32-bit-mode POSIX library version)	138,200	4,900	8,500
AIX (64-bit mode)	151,900	6,000	12,200
Solaris (32-bit mode)	79,500	1,400	3,600

OS	Value of E	Value of F	Value of G
Solaris (32-bit-mode POSIX library version)	125,100	1,700	3,300
Solaris (64-bit mode)	114,800	2,200	4,500
Linux (32-bit mode)	101,400	3,300	5,200
Linux (EM64T)	154,300	6,700	14,500
Linux 6 (64-bit x86_64)	117,200	1,900	5,800

H: If you are making a rough estimate, this is the number of audit events (the number of executions of CREATE AUDIT). If you are making a detailed estimate, it is the number of entries in the security audit information buffer

J: Memory required when using the facility for acquiring syncpoint output synchronization control information (bytes)

If 1 is specified in the `pd_dbbuff_trace_level` operand and the `pd_dfw_awt_process` operand is omitted, add:

32-bit mode

$$320 \times \text{number of global buffers defined in a single server}$$

64-bit mode

$$640 \times \text{number of global buffers defined in a single server}$$

K: If 1 or a greater value is specified in the `pd_max_resident_rdarea_no` operand, add:

$$1,648 + 16 \times \text{value of } pd_max_resident_rdarea_no + 16 \times \text{value of } pd_max_resident_rdarea_shm_no$$

L: Memory required by in-memory data processing

For the applicable formulas, see 15.1.10 Memory required by in-memory data processing.

M: Memory required by communication trace processing

32-bit mode

$$(16 \times (N - 1,024) \times 2) \times (\text{value of } pd_max_server_process - \text{value of } pd_max_users - 3)$$

64-bit mode

$$(32 \times (N - 1,024) \times 2) \times (\text{value of } pd_max_server_process - \text{value of } pd_max_users - 3)$$

N: Value of `pd_ptth_trace_max` enabled as unit control information definition

The value specified for the operand rounded up to a power of two.

P: Memory required by communication trace processing

32-bit mode

$$(16 \times (Q - 1,024) \times 2) \times (\text{value of } pd_max_users + 3)$$

64-bit mode

$$(32 \times (Q - 1,024) \times 2) \times (\text{value of } pd_max_users + 3)$$

Q: Value of `pd_ptth_trace_max` enabled as a single server definition

The value specified for the operand rounded up to a power of two.

R: Size of memory for signal handler

HP-UX (IPF) version: 512

Otherwise: 0

Table 15–3: Procedure for obtaining the number of work tables for each SQL statement

SQL statement	Procedure for obtaining the number of work tables
SELECT statement	When none of 1-8 as follows are applicable: 0
INSERT (–SELECT) statement	When any of 1-8 as follows are applicable: Sum of the applicable values from 1-8
	1. When multiple tables are joined for retrieval Number of additional work tables = (Number of joined tables - 1) \times 2 + 1
	2. When specifying the ORDER BY clause Number of additional work tables = 2

SQL statement	Procedure for obtaining the number of work tables
	3. When specifying the GROUP BY clause Number of additional work tables = Number of GROUP BY clauses specified 4. When specifying the DISTINCT clause Number of additional work tables = Number of DISTINCT clauses specified 5. When specifying the UNION, UNION ALL, or EXCEPT[ALL] clause Number of additional work tables = (Number of UNION or UNION ALL clauses specified) $\times 2 + 1$ 6. When search condition contains columns with index defined Number of additional work tables = Number of columns with index defined in the search condition 7. When specifying the FOR UPDATE or FOR READ ONLY clause Number of additional work tables = 1 8. When specifying a subquery (quantified predicate) Number of additional work tables = Number of subqueries specified
UPDATE statement DELETE statement	Number of columns with index defined in the search condition + 1
DROP SCHEMA statement DROP TABLE statement DROP INDEX statement CREATE INDEX statement REVOKE statement to revoke access privilege	2

15.1.3 Formulas for shared memory used by a unit controller

(1) 32-bit mode HiRDB

The size of the memory required for the unit controller from startup to termination of a HiRDB single server configuration is the sum of the process items listed as follows:

Ensure that the size of the shared memory for the entire unit controller does not exceed 2 gigabytes.

Process item	Shared memory calculation formula (bytes)
Scheduler	Value of <code>pd_utl_exec_mode</code> set to 0: $\{ \uparrow (432 + 304 \times n) \div 1,024 \uparrow + 500 + x + \uparrow (134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \uparrow \} \times 1,024$ Value of <code>pd_utl_exec_mode</code> set to 1: $\{ \uparrow (432 + 304 \times n) \div 1,024 \uparrow + \uparrow (m \times 2,000 + 136) \div 1,024 \uparrow + y + \uparrow (134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \uparrow \} \times 1,024$ x : Single server: $116 + 5 \times (m + 3) + 14$ Utility special unit: 0 y : Single server: $5 \times (m + 3) + 14$ Utility special unit: 0 m : Value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code> n : Number of servers in unit + number of utility servers in unit + 1 Number of utility servers in the unit: $27 + \alpha$ α : If there is a single server in the unit, 12; otherwise, 0
Lock server	<ul style="list-style-type: none"> For a unit other than a utility special unit $(320 + 48 + c + d + 48 + 4,096 + g + 48 + i + 48 + 4,096 + 48 + n + t + u + 16) \times \text{value of } pd_lck_pool_partition$

Process item	Shared memory calculation formula (bytes)
	<p>c: When <code>pd_lck_hash_entry</code> is omitted or 0 is specified:</p> $(\downarrow(8 + 4 \times \text{MAX}(\text{largest prime number that is less than } \uparrow((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2))$ $+ \downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 6) \div 10 \uparrow,$ $11,261)) \div 16 \downarrow + 1) \times 16$ <p>When 2 or a greater non-prime number is specified for <code>pd_lck_hash_entry</code>:</p> $(\downarrow(8 + 4 \times \text{largest prime number that is less than the value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$ <p>When 1 or a prime number is specified for <code>pd_lck_hash_entry</code>:</p> $(\downarrow(8 + 4 \times \text{value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$ <p>d: $((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2) +$ $\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 6) \times 96$</p> <p>g: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $((p + 3) \times 3 + p) \times 256$ <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $((p + 3) \times 3 + 32) \times 256$ <p>i: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $((\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 8 + ((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4)) \times 2)$ $+ p \times (\text{value of } \text{pd_max_rdarea_no} + 1) + (p + 3) \times 2 \times 2 \times 5)$ <p>rounded up to the next even value $\times 64$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $((\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 8 + ((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4)) \times 2)$ $+ 32 \times (\text{value of } \text{pd_max_rdarea_no} + 1) + (p + 3) \times 2 \times 2 \times 5)$ <p>rounded up to the next even value $\times 64$</p> <p>n: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $((p + 3) \times 3 + p) \times 48$ <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $((p + 3) \times 3 + 32) \times 48$ <p>p: value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code></p> <p>r: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $48 + ((p + 3) \times 3 + p) \times \uparrow \text{value of } \text{pd_max_open_holdable_cursors} \div 16 \uparrow \times 4$ <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $48 + ((p + 3) \times 3 + 32) \times \uparrow \text{value of } \text{pd_max_open_holdable_cursors} \div 16 \uparrow \times 4$ <p>u: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $48 + ((\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 8 + ((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4)) \times 2)$ $+ p \times (\text{value of } \text{pd_max_rdarea_no} + 1) + (p + 3) \times 2 \times 2 \times 5)$ <p>rounded up to the next even value $\times \uparrow \text{value of } \text{pd_max_open_holdable_cursors} \div 16 \uparrow \times 4$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $48 + ((\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 8 + ((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4)) \times 2)$ $+ 32 \times (\text{value of } \text{pd_max_rdarea_no} + 1) + (p + 3) \times 2 \times 2 \times 5)$ <p>rounded up to the next even value $\times \uparrow \text{value of } \text{pd_max_open_holdable_cursors} \div 16 \uparrow \times 4$</p> <ul style="list-style-type: none"> For a utility special unit

Process item	Shared memory calculation formula (bytes)
	8,416
Transaction server	$288 + 32 + 192 \times m \times 2 + 1,028$ $+ (420 + 564 + 256 + 384 \times 2 + 128) \times m \times 2 + 7 + 256 \times 2$ <i>m</i> : Value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>
Timer server	$32 \times (\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count + 3) \times (1 + \text{number of utility servers in unit} + 1) + 1,440$ Number of utility servers in unit is $26 + \alpha$ α : For HP-UX, Solaris, and AIX, with a single server in the unit: 12. For Linux, with a single server in the unit: 3.
Statistics log server	$384 + 128 \times 16 + 32 + 288 \times 2 + 1,024 + 128 \times 3$ $+ \text{value of } pd_stj_buff_size \times 1,024 \times 3 + 64 + 4,096 + 8,192$
Process server	$192 + 512 \times a + 80 + 256 + (\text{value of } pd_max_server_process + 50) \times (256 + 144) + 16 + 8 \times 16 + 16$ $+ 16 + 48 + 48 \times (b + 1)$ <i>a</i> : For a single server: 131 For a utility special unit: 106 <i>b</i> : If the operand <code>pd_module_trace_max</code> is specified in the system common definition or the unit control information definition: value of <code>pd_module_trace_max</code> If not: 16,383
Single server	$640 + (44 + 4) \times a \times 2 + (100 + 4) \times (b + 30 + 2) + (100 + 4) \times (c + 1) + 40 \times b \times 14 + 256 + 256$ $+ 36 \times d + 12 \times e + 8 + 5,844 + 224 + f + 16 + 1,024 + 272 \times h$ <i>a</i> : Number of single server definitions <i>b</i> : Number of single servers in unit <i>c</i> : Number of units <i>d</i> : Number of <code>-c</code> options specified in <code>pdunit</code> operand <i>e</i> : Number of <code>pdcltgrp</code> operands specified <i>f</i> : $2,052 + 128 \times (g + 3)$ <i>g</i> : For a single server: 92. For a utility special unit: 74. <i>h</i> : Number of IP addresses for hosts specified in the <code>pd_security_host_group</code> operand If the <code>pd_security_host_group</code> operand is not specified, 0
Name server	169,984
Node manager	$\uparrow (1,152 + 432 \times \text{total number of units} + 80 \times \text{total number of servers} + 7,680 + 1,008 + 56 \times \text{number of servers in unit} + 240 \times A + 44 \times A + 28 \times A + 16 \times B + 32)$ $\div 1,024 \uparrow \times 1,024$ <i>A</i> : <code>pd_utl_exec_mode</code> = 0: 1,024 <code>pd_utl_exec_mode</code> = 1 and the unit contains a single server: Value of <code>pd_max_users</code> $\times 10 + 400$ <code>pd_utl_exec_mode</code> = 1 and the unit contains no single server: Value of <code>pd_max_users</code> $\times 7$ If the value of <i>A</i> does not exceed 1,024, use 1,024 as the value of <i>A</i> . <i>B</i> : <code>pdcltgrp</code> operand not specified: 0 <code>pdcltgrp</code> operand specified: Number of <code>pdcltgrp</code> operands specified + 1
I/O server	$\uparrow (28 + (\uparrow (32 + A) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128$ <i>A</i> : $3,248 + (14 + 16) \times 972 + 1 \times 276 + 534 \times 276 + 16 \times 276$ $+ \text{value of } pd_max_file_no \times 972$ For utility special unit, the size is $(3,248 + (14 + 16) \times 972 + 1 \times 276)$ bytes.
Log server	$32 + 48 + 128 \times 19 + 384 + 128 \times 7 + 1,024 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of } pd_log_rec_leng \uparrow$ $\times \text{value of } pd_log_rec_leng + 64 + 4,096 \times 2 + (736 + 512) \times B + 128$

Process item	Shared memory calculation formula (bytes)
	$\times \text{value of } pd_log_write_buff_count$ $+ (\text{value of } pd_log_write_buff_count + A)$ $\times \uparrow \{ \text{value of } pd_log_max_data_size + (68 + 44 + 96 + 160) \} \div 4,096 \uparrow \times 4,096$ $+ C + \uparrow \{ (B + 1) \div 12 \} \uparrow \times 8,320$ Add this when the <code>pd_max_reflect_process_count</code> operand is specified. $128 + 704$ $A: 16$ $B: \text{Number of } pdlogadfg -d \text{ sys operands specified}$ $C: 0$
Synchronization point dump server	$\uparrow (368 + 1,456 \times 2) \div 1,024 \uparrow \times 1,024$ $+ \uparrow \{ (96 + 80 + 208 + 208) + 192 \times (\text{number of } pdlogadfg -d \text{ spd operands specified})$ $+ 416 \times (\text{number of } pdlogadpf -d \text{ spd operands specified}) + 1,023 \}$ $\div 1,024 \uparrow \times 1,024$
Common to all units	$a + b + 64 + (m + 3) \times c + 64 + 48 + d + e$ $+ (\text{value of } pd_max_server_process \times 2 + 100) \times (48 + 16) + 32$ $+ (\text{value of } pd_max_server_process \times 2 + 100 + 384) \times 32 + 32 + f + g + h + i + q$ $+ (\text{value of } pd_max_server_process + 127 + r) \times 32 + 32$ $a: 25,248 + p \times 4$ $b: 2,988$ $c: 1,956$ $d: 32 \times 32$ $e: 64 + 64 \times \{ (m + 3) \times 2$ $\quad + \text{MAX} (5, (\downarrow m + 3) \div 10 \downarrow) + 7 \}$ $f: 512 \times (13 + 3) \times 2$ $g: \{ (96 + \text{value of } pd_lck_until_disconnect_cnt \times 112 + 4,095) \div 4,096 \} \times 4,096 \times 2$ $h: \uparrow (\text{number of port numbers specified with } pd_registered_port \times 16 + 32 + 1,023) \div 1,024 \uparrow \times 1,024$ $\quad \text{If } pd_registered_port \text{ is omitted: } 0$ $i: \text{If } k \text{ is 2 or more, } 32 + (8 + 8 \times k) \times n$ $\quad \text{Otherwise, } 0$ $k: \text{Value of } pd_lck_pool_partition$ $m: \text{Value of } pd_max_users + \text{value of } pd_max_reflect_process_count$ $n: (m + 3) \times 2 + \text{MAX} \{ 5, \downarrow (m + 3) \div 10 \downarrow \} + 7$ $p: \text{If } Y \text{ is specified for } pd_dbbuff_modify: 512 + \text{value of } pd_max_add_dbbuff_shm_no$ $\quad \text{Otherwise: } 512$ $q: 144$ $r: \text{When the value of } pd_utl_exec_mode \text{ is set to 1 and } m > 32: (m + 3) \times 3 + m$ $\quad \text{When the value of } pd_utl_exec_mode \text{ is set to 0 or } m \leq 32: (m + 3) \times 3 + 32$
Transaction log server	$1,024 + 512 \times A$ $+ \{$ $\quad 128 \times B + 128$ $\quad + [F + \uparrow (128 + 256 + 8 + 224) \div \text{value of } pd_log_rec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $\quad + \uparrow (\text{value of } pd_log_max_data_size + 68 + 44 + 96 + 160) \div \text{value of } pd_log_rec_leng \uparrow \times \text{value of}$ $\quad pd_log_rec_leng] \times D$ $\quad + E + (48 + 8) \times B \times 2$

Process item	Shared memory calculation formula (bytes)
	$\} \times \text{number of servers in unit}$ $+ 584 \times B + 128 \times B + 64 \times B \times C + 128$ $+ \{$ $F + \lceil (128 + 256 + 8 + 224) \div \text{value of } pd_log_rec_leng \rceil \times \text{value of } pd_log_rec_leng$ $+ \lceil (\text{value of } pd_log_max_data_size + 68 + 44 + 96 + 160) \div \text{value of } pd_log_lec_leng \rceil \times \text{value of } pd_log_rec_leng$ $\}$ $+ E + (48 + 8) \times (B \times 2 + 2)$ <p>A: 2</p> <p>B: $7 + (\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count) \times 2$</p> <p>C: 1</p> <p>D: For a single server, if the value of <code>pd_log_rollback_buff_count</code> is 0, D is 8; otherwise, it is the value of <code>pd_log_rollback_buff_count</code></p> <p>For a utility special unit, 0</p> <p>E: 0</p> <p>F: 60</p>
Status server	$\lceil 64 \div 32 \rceil \times 32$
Audit trail management server	$\lceil A \div 1,024 \rceil \times 1,024$ <p>A: 640 if the <code>pd_aud_file_name</code> operand is omitted</p> <p>640 + (304 × 200) + B + C if the <code>pd_aud_file_name</code> operand is specified</p> <p>B: 0 if the <code>pd_aud_async_buff_size</code> operand value is 0</p> <p>The following value if the <code>pd_aud_async_buff_size</code> operand value is 4,096 or greater:</p> <p>For Linux:</p> $(160 \times \text{value of } pd_aud_async_buff_count \text{ operand})$ $+ \{(\lceil \text{value of } pd_aud_async_buff_size \text{ operand} \div 4,096 \rceil \times 4,096)$ $\times \text{value of } pd_aud_async_buff_count \text{ operand}\} + 4,096$ <p>For a system other than Linux:</p> $(160 \times \text{value of } pd_aud_async_buff_count \text{ operand})$ $+ \{(\lceil \text{value of } pd_aud_async_buff_size \text{ operand} \div 4,096 \rceil \times 4,096)$ $\times \text{value of } pd_aud_async_buff_count \text{ operand}\}$ <p>C: If N is specified for the <code>pd_aud_auto_loading</code> operand, 0</p> <p>If Y is specified for the <code>pd_aud_auto_loading</code> operand, $256 \times 2 + 240$</p>

(2) 64-bit mode HiRDB

The size of the memory required for the unit controller from startup to termination of a HiRDB single server configuration is the sum of the process items listed as follows:

Process item	Shared memory calculation formula (bytes)
Scheduler	<p>Value of <code>pd_utl_exec_mode</code> set to 0:</p> $\{ \lceil (432 + 304 \times n) \div 1,024 \rceil + 500 + x + \lceil (134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \rceil \} \times 1,024$ <p>Value of <code>pd_utl_exec_mode</code> set to 1:</p> $\{ \lceil (432 + 304 \times n) \div 1,024 \rceil + \lceil (m \times 2,000 + 136) \div 1,024 \rceil + y + \lceil (134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \rceil \} \times 1,024$

Process item	Shared memory calculation formula (bytes)
	<p>x: Single server: $116 + 5 \times (m + 3) + 14$ Utility special unit: 0</p> <p>y: Single server: $5 \times (m + 3) + 14$ Utility special unit: 0</p> <p>m: Value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code></p> <p>n: Number of servers in unit + number of utility servers in unit + 1 Number of utility servers in the unit: $27 + \alpha$ α: If there is a single server in the unit, 12; otherwise, 0</p>
Lock server	<ul style="list-style-type: none"> For a unit other than a utility special unit <p>$(496 + 80 + c + d + 64 + 8,192 + g + 80 + i + 64 + 8,192 + 64 + n + t + u + 16)$ \times value of <code>pd_lck_pool_partition</code></p> <p>c: When <code>pd_lck_hash_entry</code> is omitted or 0 is specified: $(\downarrow(8 + 8 \times \text{MAX}(\text{largest prime number} < \uparrow((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2))$ $+ \downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 4) \div 10 \uparrow \uparrow, 11,261)) \div 16 \downarrow$ $+ 1) \times 16$</p> <p>When 2 or a greater non-prime number is specified for <code>pd_lck_hash_entry</code>: $(\downarrow(8 + 8 \times \text{largest prime number} < \text{value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1)$ $\times 16$</p> <p>When 1 or a prime number is specified for <code>pd_lck_hash_entry</code>: $(\downarrow(8 + 8 \times \text{value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$</p> <p>$d$: $((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2) +$ $\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 4) \times 128$</p> <p>$g$: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $((p + 3) \times 3 + p) \times 320$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $((p + 3) \times 3 + 32) \times 320$</p> <p>$i$: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $((\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 5 + ((p + 3)$ $\times (\text{value of } \text{pd_max_access_tables} + 4)) \times 2)$ $+ \downarrow(\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow) \div 3 \downarrow + p \times (\text{value of } \text{pd_max_rdarea_no} + 1)$ $+ (p + 3) \times 2 \times 2 \times 5)$ rounded up to the next even value $\times 112$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $((\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow \times 5 + ((p + 3)$ $\times (\text{value of } \text{pd_max_access_tables} + 4)) \times 2)$ $+ \downarrow(\downarrow \text{value of } \text{pd_lck_pool_size} \div \text{value of } \text{pd_lck_pool_partition} \downarrow) \div 3 \downarrow + 32 \times (\text{value of } \text{pd_max_rdarea_no} + 1)$ $+ (p + 3) \times 2 \times 2 \times 5)$ rounded up to the next even value $\times 112$</p> <p>n: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $((p + 3) \times 3 + p) \times 80$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $((p + 3) \times 3 + 32) \times 80$</p> <p>$p$: value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code></p> <p>t: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p>

Process item	Shared memory calculation formula (bytes)
	$48 + ((p + 3) \times 3 + p) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $48 + ((p + 3) \times 3 + 32) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <i>u</i> : When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $48 + ((\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition} \downarrow \times 5 + ((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 2)$ $+ \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition} \downarrow \div 3 \downarrow + p \times (\text{value of } pd_max_rdarea_no + 1)$ $+ (p + 3) \times 2 \times 2 \times 5)$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $48 + ((\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition} \downarrow \times 5 + ((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 2)$ $+ \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition} \downarrow \div 3 \downarrow + 32 \times (\text{value of } pd_max_rdarea_no + 1)$ $+ (p + 3) \times 2 \times 2 \times 5)$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <ul style="list-style-type: none"> For a utility special unit 16,704
Transaction server	$304 + 32 + 192 \times m \times 2 + 1,048$ $+ (416 + 720 + 256 + 392 \times 2 + 128) \times (m \times 2 + 7) + 256 \times 2$ <i>m</i> : Value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>
Timer server	$32 \times (\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count + 3) \times (1 + \text{number of utility servers in unit} + 1) + 1,440 + (48 - 32) \times 2$ Number of utility servers in unit is $26 + \alpha$ α : For HP-UX, Solaris, and AIX, with a single server in the unit: 12. For Linux, with a single server in the unit: 3.
Statistics log server	$424 + 128 \times 16 + 32 + 288 \times 2 + 1,168 + 144 \times 3$ $+ \text{value of } pd_stj_buff_size \times 1,024 \times 3 + 64 + 4,096 + 8,192$
Process server	$208 + 528 \times a + 80 + 256 + (\text{value of } pd_max_server_process + 50) \times (256 + 160) + 16 + 8 \times 16 + 16 + 16 + 64 + 64 \times (b + 1)$ <i>a</i> : For a single server: 131 For a utility special unit: 106 <i>b</i> : If the operand <code>pd_module_trace_max</code> is specified in the system common definition or the unit control information definition: value of <code>pd_module_trace_max</code> If not: 16,383
Single server	$736 + (48 + 8) \times a \times 2 + (112 + 8) \times (b + 30 + 2) + (104 + 8) \times (c + 1) + 40 \times b \times 14 + 256 + 256 + 40 \times d + 16 \times e + 8 + 5,864 + 256 + f + 16 + 1,024 + 272 \times h$ <i>a</i> : Number of single server definitions <i>b</i> : Number of single servers in unit <i>c</i> : Number of units <i>d</i> : Number of <code>-c</code> options specified in <code>pdunit</code> operand <i>e</i> : Number of <code>pdcltgrp</code> operands specified <i>f</i> : $2,056 + 128 \times (g + 3)$ <i>g</i> : For a single server: 92. For a utility special unit: 74. <i>h</i> : Number of IP addresses for the host specified in the <code>pd_security_host_group</code> operand

Process item	Shared memory calculation formula (bytes)
	If the <code>pd_security_host_group</code> operand is not specified, 0
Name server	169,984
Node manager	$\uparrow (1,312 + 464 \times \text{total number of units in HiRDB system}$ $+ 96 \times \text{total number of servers in HiRDB system} + 10,240 + 1,200$ $+ 72 \times \text{number of HiRDB servers in the unit} + 240 \times A + 44 \times A + 28 \times A + 16 \times B + 48)$ $\div 1,024 \uparrow \times 1,024$ <i>A:</i> <code>pd_utl_exec_mode</code> = 0: 1,024 <code>pd_utl_exec_mode</code> = 1 and the unit contains a single server: (value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>) $\times 10 + 400$ <code>pd_utl_exec_mode</code> = 1 and the unit contains no single server: (value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>) $\times 7$ If the value of <i>A</i> does not exceed 1,024, use 1,024 as the value of <i>A</i> . <i>B:</i> <code>pdcltgrp</code> operand not specified: 0 <code>pdcltgrp</code> operand specified: Number of <code>pdcltgrp</code> operands specified + 1
I/O server	$\uparrow (56 + (\uparrow (56 + A) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128$ <i>A:</i> $3,248 + (14 + 16) \times 972 + 1 \times 276 + 534 \times 276 + 16 \times 276$ $+ \text{value of } pd_max_file_no \times 972 + (48 - 32) \times 3$ For a utility special unit: $(3,248 + (14 + 16) \times 972 + 1 \times 276) + (48 - 32) \times 3$ bytes
Log server	$32 + 48 + 128 \times 19 + 432 + 128 \times 7 + 1,168 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of } pd_log_rec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $+ 64 + 4,096 \times 2 + (768 + 512) \times B$ $+ 144 \times \text{value of } pd_log_write_buff_count$ $+ (\text{value of } pd_log_write_buff_count + A)$ $\times \uparrow \{ \text{value of } pd_log_max_data_size + (68 + 44 + 96 + 160) \} \div 4,096 \uparrow \times 4,096$ $+ C + \uparrow \{ (B + 1) \div 12 \} \uparrow \times 8,320$ Add this when the <code>pd_max_reflect_process_count</code> operand is specified. 128 + 704 <i>A:</i> 16 <i>B:</i> Number of <code>pdlogadfg -d sys</code> operands specified <i>C:</i> 0
Synchronization point dump server	$\uparrow (384 + 1,536 \times 2) \div 1,024 \uparrow \times 1,024$ $+ \uparrow \{ (128 + 80 + 240 + 240) + 192 \times (\text{number of } pdlogadfg -d spd \text{ operands specified})$ $+ 416 \times (\text{number of } pdlogadpf -d spd \text{ operands specified}) + 1,023 \}$ $\div 1,024 \uparrow \times 1,024$
Common to all units	$a + b + 80 + (m + 3) \times c + 64 + 48 + d + e$ $+ (\text{value of } pd_max_server_process \times 2 + 100) \times (64 + 16) + 32$ $+ (\text{value of } pd_max_server_process \times 2 + 100 + 384) \times 32 + 32 + f + g + h + i + q$ $+ (\text{value of } pd_max_server_process + 127 + r) \times 48 + 32$ <i>a:</i> 34,656 + <i>p</i> $\times 4$ <i>b:</i> 3,480 <i>c:</i> 2,760 <i>d:</i> 48×32 <i>e:</i> $80 + 96 \times \{ (m + 3) \times 2$ $+ \text{MAX}(5, (\downarrow m + 3] \div 10 \downarrow) + 7 \}$ <i>f:</i> $512 \times (13 + 3) \times 2$

Process item	Shared memory calculation formula (bytes)
	<p> $g: \{(128 + \text{value of } pd_lck_until_disconnect_cnt \times 112 + 4,095) \div 4,096\} \times 4,096 \times 2$ $h: \uparrow (\text{number of port numbers specified with } pd_registered_port \times 16 + 32 + 1,023) \div 1,024 \uparrow \times 1,024$ If $pd_registered_port$ is omitted: 0 $i: \text{If } k \text{ is 2 or more, } 32 + (8 + 8 \times k) \times n$ Otherwise, 0 $k: \text{Value of } pd_lck_pool_partition$ $m: \text{Value of } pd_max_users + \text{value of } pd_max_reflect_process_count$ $n: (m + 3) \times 2 + \text{MAX}\{5, \downarrow (m + 3) \div 10 \downarrow\} + 7$ $p: \text{If } Y \text{ is specified for } pd_dbbuff_modify: 512 + \text{value of } pd_max_add_dbbuff_shm_no + \text{value of } pd_max_resident_rdarea_shm_no$ Otherwise: $512 + \text{value of } pd_max_resident_rdarea_shm_no$ $q: \text{HP-UX (IPF) 64-bit mode: 256}$ Otherwise: 144 $r: \text{When the value of } pd_utl_exec_mode \text{ is set to 1 and } m > 32: (m + 3) \times 3 + m$ When the value of $pd_utl_exec_mode$ set to 0 or $m \leq 32: (m + 3) \times 3 + 32$ </p>
Transaction log server	<p> $1,168 + 688 \times A$ $+ \{$ $128 \times B + 144$ $+ [G + \uparrow (128 + 256 + 8 + 224) \div \text{value of } pd_log_lec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $+ \uparrow (\text{value of } pd_log_max_data_size + 68 + 44 + 96 + 160) \div \text{value of } pd_log_lec_leng \uparrow$ $\times \text{value of } pd_log_rec_leng] \times D$ $+ E + (48 + 8) \times B \times 2$ $\} \times \text{number of servers in unit}$ $+ 600 \times B + 128 \times B + 64 \times B \times C + 144$ $+ \{$ $G + \uparrow (128 + 256 + 8 + 224) \div \text{value of } pd_log_lec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $+ \uparrow \text{value of } pd_log_max_data_size + 68 + 44 + 96 + 160$ $\div \text{value of } pd_log_lec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $\}$ $+ E + (48 + 8) \times (B \times 2 + 2)$ $A: 2$ $B: 7 + (\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count) \times 2$ $C: 1$ $D: \text{For a single server, if the value of } pd_log_rollback_buff_count \text{ is 0, } D \text{ is 8;}$ otherwise, it is the value of $pd_log_rollback_buff_count$ For a utility special unit, 0 $E: 0$ $G: 64$ </p>
Status server	$\uparrow 64 \div 32 \uparrow \times 32$
Audit trail management server	<p> $\uparrow A \div 1,024 \uparrow \times 1,024$ $A: 704 \text{ if the } pd_aud_file_name \text{ operand is omitted}$ $704 + (320 \times 200) + B + C \text{ if the } pd_aud_file_name \text{ operand is specified}$ $B: 0 \text{ if the } pd_aud_async_buff_size \text{ operand value is 0}$ The following value if the $pd_aud_async_buff_size$ operand value is 4,096 or greater: For Linux version: $(176 \times \text{value of } pd_aud_async_buff_count \text{ operand})$ </p>

Process item	Shared memory calculation formula (bytes)
	$+ \{(\uparrow \text{value of } \text{pd_aud_async_buff_size} \text{ operand} \div 4,096 \uparrow \times 4,096) \\ \times \text{value of } \text{pd_aud_async_buff_count} \text{ operand}\} + 4,096$ <p>For non-Linux versions:</p> $(176 \times \text{value of } \text{pd_aud_async_buff_count} \text{ operand}) \\ + \{(\uparrow \text{value of } \text{pd_aud_async_buff_size} \text{ operand} \div 4,096 \uparrow \times 4,096) \\ \times \text{value of } \text{pd_aud_async_buff_count} \text{ operand}\}$ <p>C: If N is specified for the <code>pd_aud_auto_loading</code> operand, 0 If Y is specified for the <code>pd_aud_auto_loading</code> operand, $256 \times 2 + 256$</p>

15.1.4 Formulas for shared memory used by a single server

This subsection lists and describes the formulas used for calculating the shared memory used by a HiRDB single server configuration.

For 32-bit mode (KB):

Formula 1 + $\uparrow \{(\uparrow (40 + (\text{value obtained by adding Formulas 2 through 7 and Formula 9}) + (6.5 \times 1,024 \times 1,024)) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$

For 64-bit mode (KB):

Formula 1 + $\uparrow \{(\uparrow (72 + (\text{value obtained by adding Formulas 2 through 9}) + (6.5 \times 1,024 \times 1,024)) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$

Notes on Formulas 1 to 10:

- If Y is specified in the `pd_rdarea_open_attribute_use` or `pd_lv_mirror_use` operand, add Formula 3.
- If `commit` is specified in either the `pd_dbsync_point` operand or the `pd_system_dbsync_point` operand, add Formula 4. The default for the `pd_system_dbsync_point` operand is `commit`. Otherwise, add Formula 7.
- If the `pd_inner_replica_control` operand is specified, add Formula 5.
- If the `pd_dfw_awt_process` operand is specified, add Formula 6.
- If you omit the `pd_sds_shmpool_size` operand, the following value is set:
For 32-bit mode:
 $\uparrow \{(\uparrow (40 + (\text{value obtained by adding Formulas 2 through 7 9, 10})) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$
For 64-bit mode:
 $\uparrow \{(\uparrow (72 + (\text{value obtained by adding Formulas 2 through 9, 10})) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$
- If you specify the `pd_max_resident_rdarea_no` operand, add Formula 8.
- If the value of `pd_max_temporary_object_no` is 1 or greater, add Formula 9.
- If the value of `pd_max_tmp_table_rdarea_no` is 1 or greater, add Formula 10.

The following table shows Formulas 1 through 10.

Formula	Shared memory calculation formula
Formula 1 (KB)	<p>■ 32-bit mode</p> $b \times 1.3 + c + d + f + 1.6 + q + r + 4$ $+ \{[(a + 12) \div 13] \times 1.1 + [(a + 62) \div 63] + 3.7\} \times (e + 3) + 3.5$ $+ \{$ $\uparrow (\uparrow (b \div 64) \uparrow) \times (8 \div 16) \uparrow \times 4 \times 4$ $+ 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\}$ $+ 8 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} + 32 + 20,000$

Formula	Shared memory calculation formula
	$ \begin{aligned} & + \uparrow \{(c \div 8) + 7\} \div 64 \uparrow \times 8 + \uparrow \{(f \div 8) + 7\} \div 64 \uparrow \times 8 \\ & + \text{MAX} \{a \times (e + 3), c \div 8\} \times 104 + \text{MAX} \{a \times (e + 3), f \div 8\} \times 24 \\ & + \uparrow \{(q \div 4) + 7\} \div 64 \uparrow \times 8 \\ & + \uparrow \{[(r - (s \times 592 + t \times 916 + u \times 172)) \div 2] + 7\} \div 64 \uparrow \times 8 \\ & + \text{MAX} \{13 \times (e + 3), q \div 4\} \times 88 \\ & + \text{MAX} \{21 \times (e + 3), [r - (s \times 592 + t \times 916 + u \times 172)] \div 2\} \times 60 \\ & + 44 + 256 + 1,024 + 512^{\#1} \\ & \} \div 1,024 + y + 7.5 \\ & + \uparrow \{248 \times v \times w + 47 \times v + 72\} \div 1,024 \uparrow \\ & + \uparrow \{ \uparrow (28 + (\uparrow (32 + ((\uparrow g \div 127 \uparrow + 1) \times 2,048 + 128)) \div 32 \uparrow \times 32)) \\ & \div 128 \uparrow \times 128 \\ & \} \div 1,024 \uparrow \\ & D \\ & + \sum_{i=1} (E_i) \\ & \text{Add this if the pd_def_buf_control_area_assign operand is specified as INITIAL or if the operand is omitted.} \\ & + \{[(a + 12) \div 13] \times 1.1 + [(a + 62) \div 63] + 3.7\} \times (e + 7) \\ & \blacksquare \text{ 64-bit mode} \\ & b \times 1.3 + c + d + f + 1.6 + g + r + 5 \\ & + \{ [(a + 12) \div 13] \times 1.2 + [(a + 62) \div 63] \times 1.5 + 4.1\} \times (e + 3) + 3.5 \\ & + \{ \\ & \quad \uparrow (\uparrow (b \div 64) \uparrow) \times (8 \div 16) \uparrow \times 4 \times 4 + 12 \\ & \quad \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\} \\ & \quad + 8 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} + 48 + 20,000 \\ & \quad + \uparrow \{(c \div 8) + 7\} \div 64 \uparrow \times 8 + \uparrow \{(f \div 8) + 7\} \div 64 \uparrow \times 8 \\ & \quad + \text{MAX} \{a \times (e + 3), c \div 8\} \times 104 + \text{MAX} \{a \times (e + 3), f \div 8\} \times 40 \\ & \quad + \uparrow \{(q \div 4) + 7\} \div 64 \uparrow \times 8 \\ & \quad + \uparrow \{[(r - (s \times 600 + t \times 936 + u \times 182)) \div 2] + 7\} \div 64 \uparrow \times 8 \\ & \quad + \text{MAX} \{13 \times (e + 3), q \div 4\} \times 104 \\ & \quad + \text{MAX} \{21 \times (e + 3), [r - (s \times 600 + t \times 936 + u \times 184)] \div 2\} \times 72 \\ & \quad + 72 + 256 + 1,536 + 512^{\#1} \\ & \} \div 1,024 + y + 7.5 \\ & + \uparrow \{248 \times v \times w + 64 \times v + 72\} \div 1,024 \uparrow \\ & + \uparrow \{ \uparrow (56 + (\uparrow (56 + ((\uparrow g \div 127 \uparrow + 1) \times 2,048 + 128)) \div 32 \uparrow \times 32)) \\ & \div 128 \uparrow \times 128 \\ & \} \div 1,024 \\ & D \\ & + \sum_{i=1} (E_i) \\ & \text{Add this if the pd_def_buf_control_area_assign operand is specified as INITIAL or if the operand is omitted.} \\ & + \{[(a + 12) \div 13] \times 1.2 + [(a + 62) \div 63] \times 1.5 + 4.1\} \times (e + 7) \end{aligned} $
Formula 2 (bytes)	$ \begin{aligned} & \blacksquare \text{ 32-bit mode} \\ & 500 \times 1,024 \\ & + 5,072 \times (e + 15) + (\uparrow 372 \times g \div 16 \uparrow \times 16) + 48^{\#1} \times g + 432 \times h \\ & + 112 \times (p + 240) \\ & + 96 \times x + 32 \times j + 132 \times \{19 + (e + 3) \times 3\} \end{aligned} $

Formula	Shared memory calculation formula
	$+ 48 \times n + 48 \times \{(e+3) \times 2 + 1 + \text{MAX}(5, (e+3) \div 10)\}$ $+ 68 \times B + 152 \times (A + 120) + 80 + 32 \times g + 64^{#2} + 96^{#3} + 368^{#4}$ $+ ((\downarrow(\uparrow(g \div 8) \uparrow + 3) \div 4 \downarrow) \times 4) \times j$ ■ 64-bit mode $500 \times 1,024$ $+ 9,416 \times (e + 15) + (\uparrow 472 \times g \div 16 \uparrow \times 16)$ $+ (\uparrow 56^{#1} \times g \div 16 \uparrow \times 16) + 448 \times h$ $+ (\uparrow 136 \times (p + 240) \div 16 \uparrow \times 16)$ $+ 144 \times x + 48 \times j + 240 \times \{19 + (e+3) \times 3\}$ $+ 64 \times n + 96 \times \{(e+3) \times 2 + 1 + \text{MAX}(5, (e+3) \div 10)\}$ $+ 68 \times B + 184 \times (A + 120) + 96 + 48 \times g + 64^{#2} + 128^{#3} + 448^{#4}$ $+ ((\downarrow(\uparrow(g \div 8) \uparrow + 7) \div 8 \downarrow) \times 8) \times j$
Formula 3 (bytes)	■ 32-bit mode $\{[(\uparrow \uparrow g \div 8 \uparrow \div 4 \uparrow) \times 4] + 8\} \times \{(e+3) \times 2 + 12\}$ ■ 64-bit mode $\{[(\uparrow \uparrow g \div 8 \uparrow \div 8 \uparrow) \times 8] + 8\} \times \{(e+3) \times 2 + 12\}$
Formula 4 (bytes)	■ 32-bit mode $(32 + 16 \times x) \times (e \times 2 + 7 + 1) + 16$ ■ 64-bit mode $(48 + 32 \times x) \times (e \times 2 + 7 + 1) + 16$
Formula 5 (bytes)	$56 \times z + 16$
Formula 6 (bytes)	■ 32-bit mode $72 + 52 \times C + 68 \times x$ If you specify 1 in the pd_dbbuff_trace_level operand, add: $+ 320 \times x$ ■ 64-bit mode $96 + 56 \times C + 72 \times x$ If you specify 1 in the pd_dbbuff_trace_level operand, add: $+ 640 \times x$
Formula 7 (bytes)	■ 32-bit mode $(32 + 16 \times x) \times 10 + 16$ ■ 64-bit mode $(48 + 32 \times x) \times 10 + 16$
Formula 8 (bytes)	$16 + 112 + (48 + 48 \times G) + (48 + 32 \times H)$
Formula 9 (bytes)	$16 + 80 \times I$
Formula 10 (bytes)	$\uparrow(112 + (28 + J \times 52)) \div 8 \uparrow \times 8$

a: Value of pd_max_access_tables operand

b: Value of pd_sql_object_cache_size operand

c: Value of pd_table_def_cache_size operand

d: Value of pd_auth_cache_size operand

- e*: Value of `pd_max_users` operand
- f*: Value of `pd_view_def_cache_size` operand
- g*: Value of `pd_max_rdarea_no` operand
- h*: Value of `pd_max_file_no` operand
- j*: Number of global buffers for indexes
If *Y* is specified in the `pd_dbbuff_modify` operand, add the `pd_max_add_dbbuff_no` operand value to the number of `pdbuffer` commands that have been specified.
- n*: Value of `pd_lck_until_disconnect_cnt` operand
- p*: Value of `pd_assurance_index_no` operand
- q*: Value of `pd_type_def_cache_size` operand
- r*: Value of `pd_routine_def_cache_size` operand
- s*: Number of plug-ins installed
- t*: Total number of plug-in functions used with DML^{#5}
- u*: Total number of parameters for the plug-in functions used with DML^{#5}
- v*: Value of `pd_max_list_users` operand
- w*: Value of `pd_max_list_count` operand
- x*: Total number of global buffers (number specified in the `pdbuffer` operand)
If *Y* is specified in the `pd_dbbuff_modify` operand, add the `pd_max_add_dbbuff_no` operand value to the number of `pdbuffer` commands that have been specified.
- y*: Value of `pd_registry_cache_size` operand
- z*: Value of the `pd_inner_replica_control` operand
- A*: Value of the `pd_assurance_table_no` operand
- B*: Maximum number of transactions in the server ($2 \times e + 7$)
- C*: Value of the `pd_dfw_awt_process` operand
- D*: Total number of specified `pdplgprm` operands
- E_i*: Size of the shared memory specified by the *i*th `pdplgprm` operand.
- G*: Value of the `pd_max_resident_rdarea_no` operand
- H*: Value of the `pd_max_resident_rdarea_shm_no` operand
- I*: Value of the `pd_max_temporary_object_no` operand
- J*: Value of the `pd_max_tmp_table_rdarea_no` operand
- T*: Number of tables that will use the free space reuse facility
- #1: Add this value if neither the `pd_max_list_users` nor `pd_max_list_count` operand is 0.
- #2: Add this value if at least 1 is specified in the `pd_max_ard_process` operand.
- #3: Add this value if at least 1 is specified in the `pd_max_reflect_process_count` operand.
- #4: Add this value if the facility for predicting reorganization time is used.
- #5: You can use the following SQL statement to obtain the total number of plug-in functions and the total number of parameters for the plug-in functions used with DML:

```
SELECT COUNT(*), SUM(N_PARAM) FROM MASTER.SQL_PLUGIN_ROUTINES
WHERE PLUGIN_NAME = 'plug-in-name'
AND (TIMING_DESCRIPTOR = 'ADT_FUNCTION'
     OR TIMING_DESCRIPTOR IS NULL
     OR TIMING_DESCRIPTOR = 'BEFORE_INSERT'
     OR TIMING_DESCRIPTOR = 'AFTER_INSERT'
     OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE'
     OR TIMING_DESCRIPTOR = 'AFTER_UPDATE')
```

```

OR TIMING_DESCRIPTOR = 'BEFORE_DELETE'
OR TIMING_DESCRIPTOR = 'AFTER_DELETE'
OR TIMING_DESCRIPTOR = 'BEFORE_PURGE_TABLE'
OR TIMING_DESCRIPTOR = 'AFTER_PURGE_TABLE'
OR TIMING_DESCRIPTOR = 'INDEX_SEARCH'
OR TIMING_DESCRIPTOR = 'INDEX_COUNT'
OR TIMING_DESCRIPTOR = 'INDEX_INSERT'
OR TIMING_DESCRIPTOR = 'INDEX_BEFORE_UPDATE'
OR TIMING_DESCRIPTOR = 'INDEX_AFTER_UPDATE'
OR TIMING_DESCRIPTOR = 'INDEX_DELETE'
OR TIMING_DESCRIPTOR = 'PURGE_INDEX'
OR TIMING_DESCRIPTOR = 'INDEX_MAINTENANCE_DEFERRED'
OR TIMING_DESCRIPTOR = 'BEFORE_INSERT_DC'
OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE_DC'
OR TIMING_DESCRIPTOR = 'BEFORE_DATA_CHECK'
OR TIMING_DESCRIPTOR = 'AFTER_DATA_CHECK')

```

15.1.5 Formula for size of shared memory used by global buffers

The size of the shared memory used by the global buffers is calculated for each `pdbuffer` statement using Formula 1. If `Y` is specified in the `pd_dbbuff_modify` operand, add Formula 2. The total value obtained from Formulas 1 and 2 is the memory size required for use by the global buffers.

If `fixed` is specified for the `pd_dbbuff_attribute` operand, pages are locked in real memory, which reduces the real memory portion of the virtual memory by the same size. In addition, the same size is allocated from virtual memory, which consists of the remaining real memory and swap area.

If the `pdbuffer` operand is omitted, HiRDB will automatically calculate the shared memory size, so it need not be estimated.

Formula	Shared memory calculation formula (KB)
Formula 1	<p>32-bit mode</p> $n \sum_{i=1} \left\{ \begin{aligned} &\uparrow \{752 + 64 + (296 + 64^{\#1}) \times (Pi + 4) \\ &+ (124 + 80^{\#2} + 96 \times A \times Mi) \times Ui\} \div 4,096 \uparrow \times 4,096 \\ &+ Si \times \{Pi + 4 + (Ui \times Mi \times A)\} \end{aligned} \right\} \div 1,024$ <p>64-bit mode</p> $n \sum_{i=1} \{management\ region\ part + data\ storage\ part\} \div 1,024$ <p><i>management region part:</i></p> $\uparrow \{944 + 64 + (480 + 112^{\#1}) \times (Pi + 4) + (176 + 96^{\#2} + 136 \times A \times Mi) \times Ui\} \div 4,096 \uparrow \times 4,096$ <p><i>data storage part:</i></p> $Si \times \{Pi + 4 + (Ui \times Mi \times A)\}$
Formula 2	<p>32-bit mode</p> $\{ \{ \uparrow (\uparrow (s \times 1,024 \div 2) \div 8 \uparrow + 112) \div 2,048 \uparrow \times 2,048 \times \uparrow a \div (s \times 1,024) \uparrow \} \div 1,024$ <p>64-bit mode</p> $\{ \{ \uparrow (\uparrow (s \times 1,024 \div 2) \div 8 \uparrow + 144) \div 2,048 \uparrow \times 2,048 \times \uparrow a \div (s \times 1,024) \uparrow \} \div 1,024$

n : Number of global buffer pools

i : Global buffer pool definitions to be calculated

P : Number of global buffer sectors

A : If the asynchronous READ facility is used, 2; if it is not used, 1.

M : Maximum number of batch input pages

If at least 1 is specified in the `pd_max_ard_process` operand, this is twice the specified value.

U : Maximum number of concurrently executable prefetch operations

S : Maximum page length of the RDAREAs allocated to global buffer

s : SHMMAX specified value

a : Total from Formula 1

#1: Add this value for the LOB global buffer.

#2: Add this value if at least 1 is specified in the `pd_max_ard_process` operand.

15.1.6 Formulas for size of memory required during SQL execution

(1) Procedure for obtaining the size of the memory required during execution of rapid grouping facility

If `PDSQLOPTLVL` is specified in the client environment definition, `pd_optimize_level` is specified in the system common definition, or this operand is omitted, executing an SQL statement that satisfies the applicable conditions will activate the rapid grouping facility. In such a case, HiRDB allocates process private area on the basis of the value of the `PDAGGR` operand in the client environment definition. The size of the memory can be obtained from the following formula (in bytes).

Formula:

$$e + \lceil d \div 4 \rceil \times 4 + \lceil (17 + 4 \times a + 4 \times b + c + d) \div 4 \rceil \times 4 \times (N + 1) \text{ (bytes)}$$

a : Number of columns subject to grouping

b : Number of operations by set functions

Each of COUNT, SUM, MAX, and MIN is counted as 1.

Each of AVG (COUNT) and AVG (SUM) is counted as 2.

c : Length of rows subject to grouping (see Table 15-4 Length of column subject to grouping and length of operation area for set functions)

d : Length of operation area for set functions (see Table 15-4 Length of column subject to grouping and length of operation area for set functions)

e : 32-bit mode: MAX ($4 \times N \times 24$, 16,408)

64-bit mode: MAX ($8 \times N \times 40$, 32,808)

N : Value of `PDAGGR` operand in the client environment definition

Table 15-4: Length of column subject to grouping and length of operation area for set functions

Column's data type	Column length	Length of operation area for set function ^{#1}
INTEGER	4	6
SMALLINT	2	4 ^{#2}
DECIMAL (p, s)	$\lceil (p + 1) \div 2 \rceil$	$\lceil (p + 7) \div 2 \rceil$ ^{#3}
FLOAT	8	10
SMALLFLT	4	6

Column's data type	Column length	Length of operation area for set function ^{#1}
INTERVAL YEAR TO DAY	5	8
INTERVAL HOUR TO SECOND	4	6
CHAR (n)	N	n + 3
VARCHAR (n)	n + 2	n + 5
NCHAR (n)	2 × n	2 × n + 2
NVARCHAR (n)	2 × n + 2	2 × n + 4
MCHAR (n)	N	n + 3
MVARCHAR (n)	n + 2	n + 5
DATE	4	6
TIME	3	6
BLOB (n)	--	
BINARY (n)	n + 2	n + 5

Legend:

--: Not applicable

#1

If the set function is COUNT, the length of the operation area is always 6, regardless of the data type.

#2

If the set function is AVG or SUM, the length of the operation area is 6.

#3

If the set function is AVG or SUM, the length of the set function operation region is the following value:

If the set function value type is DECIMAL and precision is 29 digits: 18

If the set function value type is DECIMAL and precision is 38 digits: 23

For details about the data type rules of set functions, see *Set functions* in the manual *HiRDB Version 9 SQL Reference*.

(2) Procedure for obtaining the size of the memory required when data suppression by column is specified

The following formula can be used to obtain the size of the memory (in bytes) required to access a table for which data suppression by column is specified (table for which SUPPRESS is specified in the column definition of CREATE TABLE):

Formula:

$$a + 128 \text{ (bytes)}$$

a: Sum of the lengths of columns in the table for which data suppression by column is specified

(3) Procedure for obtaining the size of the memory required during hash join and subquery hash execution

If you specify the PDADDITIONALOPTLVL operand in the client environment definition or the pd_additional_optimize_level operand in the HiRDB system definition, the SQL extension optimizing option becomes available. If you specify an application of hash join, subquery hash execution (APPLY_HASH_JOIN) with this SQL extension optimizing option, the system allocates the following size of process private area when a table join or subquery SQL statement is executed:

Formula**32-bit mode** a

$$\sum_{i=1}^a (13 \times 1,024 + 6 \times 1,024 \times b + c)$$

 $i=1$ **64-bit mode** a

$$\sum_{i=1}^a (13 \times 1,024 + 7 \times 1,024 \times b + c) \text{ (bytes)}$$

 $i=1$

a : Maximum number of hash joins in the `SELECT` statement

For details about the maximum number of hash joins in the `SELECT` statement, see the *HiRDB Version 9 UAP Development Guide*.

b : Obtain the hash join processing to be applied on the basis of the number of hash table rows, and then determine this value from the following table:

Guidelines for the number of hash table rows	Hash join processing to be applied		Value of b
1,500 or less	Batch hash join		0.5
$1,500 \times (\text{packet split count} \div 3)$ or less	Packet split hash join	1-level packet split	1
$1,500 \times (\text{packet split count} \div 3)^2$ or less		2-level packet split	2
Greater than $1,500 \times (\text{packet split count} \div 3)^2$		3-level packet split	3

Number of hash table rows: For join, it is the inner table count; for subquery, it is the subquery search count excluding the predicates that contain external reference rows in the search condition.

Packet split count: $\text{MIN} \{ \downarrow (\text{size of hash table} \div 2) \div \text{page length of hash table} \downarrow, 64 \}$

Hash table size: Value of the `pd_hash_table_size` operand specified in the HiRDB system definition or the value of the `PDHASHTBLSIZE` operand specified in the client environment definition.

Page length of hash table: Select the page length of the hash table corresponding to c (maximum length of hash table row) from the following table:

Maximum length of hash table row	Page length of hash table (bytes)
0 to 1,012	4,096
1,013 to 2,036	8,192
2,037 to 4,084	16,384
4,085 to 16,360	32,768
16,361 to 32,720	$\uparrow (\text{maximum length of hash table row} + 48) \div 2,048 \uparrow \times 2,048$

c : Maximum length of hash table row

For details about the length of a hash table row, see the *HiRDB Version 9 UAP Development Guide*.

(4) Procedure for obtaining the size of the memory required when the snapshot method is used

If the `pd_pageaccess_mode` operand is omitted, or if `SNAPSHOT` is specified, then the page access method for data retrieval uses the snapshot method when an SQL statement for which the snapshot method is applicable is executed. At this time, memory in the process private area is allocated automatically, as shown below, based on the page size of the table or index storage RDAREA.

Formula

$$a \times 2 \text{ (bytes)}$$

a : Maximum page length in the RDAREA where the relevant table or index is stored
However, LOB RDAREAs are excluded.

(5) Determining the size of the memory required to retrieve the first n records

If the function for retrieving n rows of search results from the top is used, you can retrieve n rows from the top of the search results (or from the location resulting from skipping as many rows from the top as specified by the user as an offset).

If the number of rows specified in the `LIMIT` clause is 1 or greater and the value of (number of offset rows + number of rows specified in the `LIMIT` clause) is 32,767 or less, as many rows are retained in memory as will fit in (number of offset rows + number of rows specified in the `LIMIT` clause). The size of the process private area to be allocated can be determined by the formula shown below. If the value of (number of offset rows + number of rows specified in the `LIMIT` clause) is 32,768 or greater, see *Chapter 18. Determining Work Table File Size* because a work table is created.

Formula

$$\{100 + (a + 2) \times (\text{number of offset rows} + \text{number of rows specified in the } \texttt{LIMIT} \text{ clause})\} \times b \text{ (bytes)}$$

a : Row length

The row length cannot exceed 32,720 bytes. The row length is calculated with the following formula:

m

$$\sum_{i=1} (A_i) + 2 \times m + 4 + c \text{ (bytes)}$$

$i=1$

m : Number of rows specified in the selection formula, `GROUP BY` clause, or `ORDER BY` clause

Add 1 if the `FOR UPDATE` clause is specified. However, if `ROW` is specified in the selection formula, this becomes the total number of rows in the table.

A_i : Data length of the i th column of the records stored in the first n records of the allocation area

For details about column data length, see *Table 16-1 List of data lengths*, and determine the length beginning by assigning the defined length to d .

However, for `BLOB` data, character data with a fixed length of at least 256 bytes (including National character data and mixed character string data), or `BINARY` data of columns without the following attributes, the value is 12:

- Columns specified in a selection formula with the `DISTINCT` clause specified
- A query specification selection formula using a concatenation operation based on `UNION [ALL]`
- Columns specified in the `ORDER BY` clause

Also, if the `FOR UPDATE` clause is specified and 1 is added for m , use 12 bytes for A_i .

c : 8

However, in the following cases, use 0:

- There is an exclusive lock in the `EX` mode on the retrieval table
- `WITHOUT LOCK` is specified
- The rapid grouping facility is specified
- Multiple tables are combined

b : Number of allocated areas for the first n records

The number of allocated areas for the first n records is calculated with the following formula:

$$1 + \text{number of } \texttt{UNION [ALL]} \text{ clause specifications}$$

(6) Determining the size of the memory required for executing SQL statements specifying an index-type plug-in function as search condition

To determine the size of memory that is allocated in the process private area when an SQL statement specifying an index-type plug-in function as search condition is executed, use the following formula:

Formula

$$a \times 500 + (20 + 6) \times 800 + 16 \text{ (bytes)}$$

a : Row length. To determine the row length, use the following formula:

$$\sum_{i=1}^m (Ai) + 4 \times (m + 2) + 12 + 4 + 8 \text{ (bytes)}$$

m : Number of columns specified in the selection formula, join condition, GROUP BY clause, or ORDER BY clause

If you specified the FOR UPDATE clause, add 1. If ROW is specified in the selection formula, the total number of rows in the table is assumed.

Ai : Length of column data i in the row to be retrieved

For details about column data length, see *Table 16-1 List of data lengths*, and determine the length beginning by assigning the defined length to d .

A length of 12 bytes is assumed for a column with BLOB data or character string data with a defined length of 256 bytes or greater (including national character data and mixed character string data) that is none of the following:

- Column specified in a join condition (join column)
- Column specified in a selection formula specifying the DISTINCT clause
- Column specified in a selection formula in a subquery of a quantified predicate
- Column specified in the selection formula in a subquery of IN predicate
- Selection formula in a subquery that is the target of Set Operation due to UNION [ALL] or EXCEPT [ALL]
- Column specified in an ORDER BY clause

If the FOR UPDATE clause is specified, Ai corresponding to 1 that was added to m is 12 bytes.

(7) Determining the size of the memory required to use the facility for output of extended SQL error information

When the facility for output of extended SQL error information is used, a process private area is allocated in the following cases:

(a) When the OPEN statement is executed

Formula

32-bit mode

$$(16 + 16 \times m) + a \text{ (bytes)}$$

64-bit mode

$$(24 + 24 \times m) + a \text{ (bytes)}$$

a : Total data length of ? parameters or embedded variables

m

$$a = \sum_{i=1}^m (a_i)$$

$i=1$

m : Number of ? parameters or embedded variables in the SQL statement

a_i : Data length of the i^{th} ? parameter or the embedded variable

The following table lists the data lengths of embedded variables and ? parameters.

Table 15–5: Data length of embedded variables and ? parameters

Data type	Column length (without indicator variable)	Column length (with indicator variable, embedded variable, and ? parameter)
INTEGER	4	6
SMALLINT	2	4
DECIMAL (p, s)	$\uparrow (p + 1) \div 2 \uparrow$	$\uparrow (p + 5) \div 2 \uparrow$
FLOAT	8	10
SMALLFLT	4	6
INTERVAL YEAR TO DAY	5	7
INTERVAL HOUR TO SECOND	4	6
CHAR (n)	n	$n + 2$
VARCHAR (n)	$n + 2$	$n + 4$
NCHAR (n)	$2 \times n$	$2 \times n + 2$
NVARCHAR (n)	$2 \times n + 2$	$2 \times n + 4$
MCHAR (n)	n	$n + 4$
MVARCHAR (n)	$n + 2$	$n + 4$
DATE	4	6
TIME	3	5
BLOB (n)	$n + 4$	$n + 8$
TIMESTAMP (p)	$7 + (p \div 2)$	$9 + (p \div 2)$
BINARY (n)	$n + 4$	$n + 8$

(b) When the PREPARE statement of the definition SQL is executed

Formula

$SQL\ statement\ length + 20$ (bytes)

(8) Calculating required memory for defining substructure indexes or updating tables that define substructure indexes

(a) When a substructure index is defined

Use the following formula to calculate the process private area used to define a substructure index with CREATE INDEX of the definition SQL.

Formula

$(index\ key\ length^{\#} \times 100 + 64)$ (bytes)

#

The maximum definition length of the substructure index defined in the table.

(b) When a table is updated that defines a substructure index

Use the following formula to calculate the process private area used to update a table that defines a substructure index with INSERT, UPDATE or DELETE of the data manipulation SQL.

Formula

$$(\text{index key length}^{\#1} \times 100 + 64 + 128) + \sum (\text{index key length} + 128)^{\#2} \text{ (bytes)}$$

#1

The maximum definition length of the substructure index defined in the table.

#2

The number of substructure indexes that specify USING UNIQUE TAG.

(9) Determining the size of the memory required to execute data manipulation SQL statements on compressed columns

If SQL statement execution, data storage processing, or extraction processing involves compressed columns, HiRDB allocates a process private area whose memory size is as shown below.

Formula

$$\text{MIN}(\text{split compression size, definition length of compressed column})^{\#} \times C + L \text{ (bytes)}$$

C: If any of the following conditions is true, 2; if not, 1:

- The SUBSTR function is used.
- The POSITION function is used.
- Backward deletion/updating of data is performed.

L: Page length of the RDAREA containing the compressed table to be processed by the SQL statement

If multiple RDAREAs are processed, use the maximum page length.

#

Use the maximum value for all the compressed columns subject to SQL statement processing.

15.1.7 Formula for size of memory required during SQL preprocessing

(1) Size of memory required when no stored procedure is used

If no stored procedure is used, the following formula can be used to obtain the size of the memory that is allocated during SQL preprocessing (KB).

Formula

$$\begin{aligned} & \uparrow \{ \\ & 2,586 + Si \times 60 + Pi \times 20 + Ti \times 1,424 + Ci \times Ti \times 72 + Wi \times 776 + Ti \times Wi \times 72 \\ & + Ki \times 276 + Ki \times Ti \times 72 + Li \times 3 + Li \times Ti + Di \times Ti \times 134 + Ari \times 108 \\ & + Gi \times 44 + Sli \times 40 + Upi \times 110 \\ & + Fi \times 90 + Ti \times Cwi \times 48 \\ & + \text{MAX}(Pi, Wpi) \times 60 \\ & \} \times 1.2 \div 1,024 \uparrow \text{ (KB)} \end{aligned}$$

Si: Number of items to be retrieved in SQL statements

Pi: Number of embedded variables, ? parameters, or SQL parameters in SQL statements

Ti: Number of table names in SQL statements

Ci: Number of column names in SQL statements

Wi: Number of predicates used in Boolean operators (AND and OR) in SQL statements

Ki: Number of literals in SQL statements

Li: Total length of literals in SQL statements (bytes)

Di: Total number of storage conditions defined in SQL statements

Ari: Number of arithmetic operations and concatenation operations in SQL statements

Gi: Number of columns specified in GROUP BY clause of SQL statements

Ori: Number of column specification or sort item specification numbers in ORDER BY clause of SQL statements

Fi: Total number of set functions and scalar functions in SQL statements

Sli: Number of queries specified in SQL statements

Upi: Number of columns to be updated in SQL statements

Cwi: Number of WHENs in CASE expression of SQL statements

Wpi: Number of variables corresponding to WITH clause of SQL statements

Note

If `SELECT_APSL` is applied, this value is 3; otherwise, it is 1. The access path display utility (`pdvwopt`) can be used to determine whether `SELECT_APSL` is applied. For details about the access path display utility (`pdvwopt`), see the manual *HiRDB Version 9 Command Reference*.

(2) Procedure for obtaining the size of the memory required when using stored procedures

If stored procedures are used, the size of the memory (in KB) to be allocated during SQL preprocessing is the value obtained from the formula shown in (1) above plus the length of the procedure control object for each stored procedure. For details about the formula for obtaining the length of a procedure control object, see the section on the `pd_sql_object_cache_size` operand of the system common definition. For details about the length of the procedure control object per stored procedure, see *Formula for determining the size of the routine control object of a routine* in the manual *HiRDB Version 9 System Definition*.

15.1.8 Formula for size of memory required during BLOB data retrieval or updating (HiRDB single server configuration)

Use the following formula to determine the size of the memory required during BLOB data retrieval or updating.

Formula

$$a + b + 17 \text{ (KB)}$$

a: Maximum value from the following formula for BLOB input variables or output variables specified in one SQL statement:

$$\begin{aligned} &\uparrow \{ \\ &\quad c \\ &\quad \sum_{i=1}^c (\text{actual length of BLOB input variable } i^{\#1} + 118) + \\ &\quad d \\ &\quad \sum_{j=1}^d (\text{specified length of BLOB output variable } j^{\#2} + 86) \\ &\quad \} \div 1,024 \uparrow \end{aligned}$$

#1: This is the actual length of BLOB data passed as embedded variables from the UAP to the HiRDB server.

#2: This is the declared length of the UAP embedded BLOB data type variables received from the UAP and returned from HiRDB to the UAP. If it is an INSERT or SELECT statement, the BLOB type reflected from the SELECT side is an output variable.

b : Maximum value from the following formula for a combination of SQL statements performing join retrieval with simultaneously open cursors:

$$\uparrow \left\{ \sum_{i=1}^e \sum_{j=1}^d (\text{defined length of BLOB output variable } j + 18) \right\} \div 1,024 \uparrow$$

c : Number of input variables

d : Number of output variables

e : Number of simultaneously open cursors

15.1.9 Formula for size of memory required during block transfer or array FETCH

To determine the size of the memory required for block transfer or array `FETCH`, use the formulas below.

Condition		Value specified in the <code>PDBLKBUFSIZE</code> operand	
		Omitted or 0	1 or greater
An array-type embedded variable is specified in the <code>INTO</code> clause of the <code>FETCH</code> statement		Formula 1	
An array-type embedded variable is not specified in the <code>INTO</code> clause of the <code>FETCH</code> statement	<code>PDBLK</code> operand is omitted or is set to 1	--	Formula 2
	<code>PDBLK</code> operand is set to 2 or greater	Formula 1	

Legend:

--: Not applicable

Formula 1

$$\uparrow \{ 864 + 16 \times a + (6 \times a + 2 \times d + b) \times c \} \div 1,024 \uparrow \text{ (kilobytes)}$$

a : Number of retrieved items specified in the `SELECT` clause

b : Data length per row in the retrieval results obtained by the `FETCH` statement (sum of the maximum length of each column, in bytes)

c : Value of the `PDBLK` operand or number of arrays

d : Number of selection formulas with `BINARY` type specified in the search item specified in the `SELECT` clause

Formula 2

$$\text{MAX}(X_1, X_2) \text{ (kilobytes)}$$

$$X_1: \uparrow (864 + 22 \times a + 2 \times c + b) \div 1,024 \uparrow$$

X_2 : Value of the `PDBLKBUFSIZE` operand

a : Number of retrieved items specified in the `SELECT` clause

b : Data length per row in the retrieval results obtained by the `FETCH` statement (sum of the length of each column that is actually obtained, in bytes)

c: Number of selection formulas with `BINARY` type specified in the search item specified in the `SELECT` clause

15.1.10 Memory required by in-memory data processing

Use the following formulas to calculate the memory required by in-memory data processing.

Formula

Formula 1 + <i>D</i> × 2 (KB)
Formula 1
$\sum_{i=1}^n \uparrow \{736 + 32 \times A + 48 + 448 \times B + 2,048 + C \times B\} \div 1,024 \uparrow \text{ (KB)}$
<i>n</i> : Number of in-memory RDAREAs <i>A</i> : Number of HiRDB files that constitute the in-memory RDAREAs <i>B</i> : Total number of pages of in-memory RDAREAs <i>C</i> : Page size of in-memory RDAREAs <i>D</i> : Value of Formula 2
Formula 2 (number of shared memory segments used by in-memory data buffer)
$\uparrow \text{value of Formula 1} \div (\text{value of SHMMAX operand} \times 1,024) \uparrow$

Formula 2 calculates the value per RDAREA. Calculate for as many in-memory RDAREAs as there are.
The value found by Formula 2 is used to calculate the `pd_max_resident_rdarea_shm_no` operand or the operating system parameters.

15.2 Estimating the memory size required for a HiRDB parallel server configuration

This section explains how to estimate the size of the memory required for each unit constituting a HiRDB parallel server configuration. The topics covered include:

- Memory allocation
- Calculation of required memory
- Formulas for shared memory used by a unit controller
- Formulas for shared memory used by each server
- Formulas for size of shared memory used by global buffers
- Formulas for size of memory required during SQL execution
- Formula for size of memory required during SQL preprocessing
- Formula for size of memory required during BLOB data retrieval or updating (for front-end servers)
- Formula for size of memory required during block transfer or array FETCH (for front-end servers)
- Formula for size of memory required during BLOB data retrieval or updating (for back-end servers or dictionary servers)

15.2.1 Memory allocation

The following figure shows the memory allocation for each unit of a HiRDB parallel server configuration.

Figure 15–2: Memory allocation for each unit of a HiRDB parallel server configuration

Shared memory						Process private memory	
Unit controller shared memory	Global buffer shared memory	Utility shared memory	Security monitoring information buffer shared memory	Inter-process memory communication shared memory	Trouble-shooting information acquisition shared memory	Process private memory for unit controller processes	Process private memory for server processes
<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">A</div> <div style="border: 1px solid black; padding: 5px;">B</div>	<div style="border: 1px solid black; width: 20px; height: 60px; margin-bottom: 5px;"></div> <div style="text-align: center;">...</div> <div style="border: 1px solid black; width: 20px; height: 60px;"></div>	<div style="border: 1px solid black; width: 20px; height: 60px;"></div>	<div style="border: 1px solid black; width: 20px; height: 60px;"></div>	<div style="border: 1px solid black; width: 20px; height: 60px; margin-bottom: 5px;"></div> <div style="text-align: center;">...</div> <div style="border: 1px solid black; width: 20px; height: 60px;"></div>	<div style="border: 1px solid black; width: 20px; height: 60px;"></div>	All processes of unit controller <div style="border: 1px solid black; width: 20px; height: 20px; display: inline-block;"></div> ... <div style="border: 1px solid black; width: 20px; height: 20px; display: inline-block;"></div>	All processes in server <div style="border: 1px solid black; padding: 2px; display: inline-block;">bes1</div> ... <div style="border: 1px solid black; padding: 2px; display: inline-block;">bes1</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">fes1</div> ... <div style="border: 1px solid black; padding: 2px; display: inline-block;">fes1</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">ds1</div> ... <div style="border: 1px solid black; padding: 2px; display: inline-block;">ds1</div>

Legend:

- A: Part used by a unit controller process
- B: Part used by a server process
- bes: Back-end server
- fes: Front-end server
- ds: Dictionary server

The following table lists details about the shared memory for each unit of a HiRDB parallel server configuration.

Table 15–6: HiRDB parallel server configuration shared memory details per unit

Item	Type of shared memory					
	Unit controller shared memory	Global buffer shared memory	Utility shared memory	Security audit information buffer shared memory	Inter-process memory communication shared memory	Troubleshooting information acquisition shared memory
Purpose	System control	Global buffers	Communication between the unit controller and utilities	Security audit information buffer	Client-server inter-process communication	Acquisition of troubleshooting information
Processes	All HiRDB processes	Back-end servers, dictionary servers	Utility processes	Front-end servers	Front-end servers, client processes	All HiRDB processes
Number of segments	1	<ul style="list-style-type: none"> If the facility for dynamic updating of global buffers is not used: 1 to 512^{#1} If the facility for dynamic updating of global buffers is used: 32-bit mode: 1 to 1,012^{#1} 64-bit mode: 1 to 1,512^{#1} 	1	1	Number of clients connected using the PDIPC=MEMORY environment variable (0-2,000) × 2	1
Maximum value per segment	See Table 15-7 <i>Size of memory required for each unit of a HiRDB parallel server configuration.</i> Value of the shmmax OS parameter must be equal to or greater than the calculation value.	Divide the segment by the SHMMAX operand value. Value of the shmmax OS parameter must be equal to or greater than the SHMMAX operand value.	See Table 15-7 <i>Size of memory required for each unit of a HiRDB parallel server configuration.</i> Value of the shmmax OS parameter must be equal to or greater than the calculation value.	See Table 15-7 <i>Size of memory required for each unit of a HiRDB parallel server configuration.</i> Value of the shmmax OS parameter must be equal to or greater than the calculation value.	See Table 15-7 <i>Size of memory required for each unit of a HiRDB parallel server configuration.</i> Value of the shmmax OS parameter must be equal to or greater than the calculation value.	10 MB
Allocation conditions	None	There must be a global buffer definition	Specify pd_utl_exec_mode=1	Specify the pd_aud_file_name operand as the HiRDB file system area name for the audit trail.	There are clients connected using the PDIPC=MEMORY environment variable	None
Creation timing	At unit activation (including	<ul style="list-style-type: none"> At server activation (including 	When utilities are executed	When a front-end server starts up	When client and server are connected	At unit activation (including

Item	Type of shared memory					
	Unit controller shared memory	Global buffer shared memory	Utility shared memory	Security audit information buffer shared memory	Inter-process memory communication shared memory	Troubleshooting information acquisition shared memory
	standby unit activation when user server hot standby or the rapid system switchover facility is used)	standby unit activation when the rapid system switchover facility is used) <ul style="list-style-type: none"> When <code>pdbufmod -k {add upd}</code> is executed 				standby unit activation when user server hot standby or the rapid system switchover facility is used)
Deletion timing	At next unit activation (including standby unit activation when user server hot standby or the rapid system switchover facility is used)	<ul style="list-style-type: none"> When <code>pdbufmod -k del</code> is executed For normal termination or planned termination: When the server is terminated For forced termination, abnormal termination, or termination of standby unit when the rapid system switchover facility is used: When the unit is next activated 	10 minutes after the utility terminates	When a front-end server quits	When client and server are disconnected	At unit termination
Indication by <code>pdls -d mem</code>	Indicated	Indicated	Indicated	Indicated	Not indicated	Not indicated
SHM-OWNER of <code>pdls -d mem</code>	MANAGER	Server name	UTILITY	AUDDEF	Not indicated	Not indicated
Related operands	<ul style="list-style-type: none"> <code>pd_shmpool_attribute</code> <code>pd_dic_shmpool_size</code> <code>pd_bes_shmpool_size</code> 	<ul style="list-style-type: none"> <code>pd_dbbuf_f_attribute</code> <code>pd_dbbuf_f_modify</code> <code>pdbuffer</code> SHMMAX 	<ul style="list-style-type: none"> <code>pd_utl_exec_mode</code> 	<ul style="list-style-type: none"> Operands related to the security audit facility^{#2} 	<ul style="list-style-type: none"> PDIPC PDSENDMESSAGE PDRECVMESSAGE 	None

Item	Type of shared memory					
	Unit controller shared memory	Global buffer shared memory	Utility shared memory	Security audit information buffer shared memory	Inter-process memory communication shared memory	Troubleshooting information acquisition shared memory
Remarks	--	--	Can be created only when <code>pd_utl_exec_mode=1</code> (when <code>pd_utl_exec_mode=0</code> , the relevant space is allocated in the unit controller shared memory).	--	--	--

Legend:

--: Not applicable.

#1: Number of global buffers allocated per back-end server or dictionary server.

#2: For details, see the manual *HiRDB Version 9 System Definition*.

15.2.2 Calculation of required memory

The size of the memory required for each unit of a HiRDB parallel server configuration is the sum of the items listed in the following table.

For details about the value specified in the `shmmax` operating system parameter (for Solaris, `shmsys:shminfo_shmmax`; for Linux, `SHMMAX`), see *Chapter 20. Specifying OS Parameters*.

If you increase the size of shared memory, it might affect transaction performance by increasing the number of page faults. For each operand, consult the guidelines for your particular system and then specify the most appropriate value.

Table 15–7: Size of memory required for each unit of a HiRDB parallel server configuration

Item		Required memory (KB)
Process private area	Process private area used by all unit controller processes	<ul style="list-style-type: none"> 32-bit mode $J + K \times \text{number of FESs in unit} + L \times (\text{number of BESs in unit} + \text{number of DSs in unit}) + \uparrow \{(64 + 48 \times (v + 1)) \times (\text{value of } pd_max_server_process - w - 3) + (64 + 48 \times (ac + 1)) \times 3 + z\} \div 1,024 \uparrow$ <ul style="list-style-type: none"> 64-bit mode $J + K \times \text{number of FESs in unit} + L \times (\text{number of BESs in unit} + \text{number of DSs in unit}) + \uparrow \{(64 + 64 \times (v + 1)) \times (\text{value of } pd_max_server_process - w - 3) + (64 + 64 \times (ac + 1)) \times 3 + z\} \div 1,024 \uparrow + ab$ <ul style="list-style-type: none"> If a plug-in is used, add: $+ 1,400$ <ul style="list-style-type: none"> If the asynchronous READ facility is used, add: $+ s$ <ul style="list-style-type: none"> If Real Time SAN Replication is used, add: $+ \uparrow (A + B + C) \div 1,024 \uparrow$

Item			Required memory (KB)
Process private area used by each server process ^{#1, #2}			<ul style="list-style-type: none"> If <code>fixed</code> is specified in the <code>pd_process_terminator</code> operand, add: $+ M \times (\text{value of } pd_process_terminator_max - 1)$ <ul style="list-style-type: none"> If you are performing in-memory data processing, add: $+ \uparrow \{T \times (\text{value of } pd_max_bes_process \times 2 + 7) \times \text{number of BESs in unit} \} \div 1,024 \uparrow$ <ul style="list-style-type: none"> If you are changing the maximum number of communication traces stored, add: $+ \uparrow V \div 1,024 \uparrow$
	Front-end server		$(N + h + m + p + q) \times b + y$ <ul style="list-style-type: none"> If you are changing the maximum number of communication traces stored, add: $+ \uparrow W \div 1,024 \uparrow$
	Dictionary server	pd_work_buff_mode=each specified	$\{P + i + m + (a + 9) \times 2 + r + t\} \times b + y + S$ <ul style="list-style-type: none"> If you are changing the maximum number of communication traces stored, add: $+ \uparrow W \div 1,024 \uparrow$
		pd_work_buff_mode=pool specified or omitted	<ul style="list-style-type: none"> 32-bit mode $(P + i + m + a + \uparrow a \div 128 \times 0.1 \uparrow + 11 + n + r + t) \times b + y + S$ <ul style="list-style-type: none"> 64-bit mode $(P + i + m + a + \uparrow a \div 128 \times 0.1 \uparrow + 15 + n + r + t) \times b + y + S$ <ul style="list-style-type: none"> If you are changing the maximum number of communication traces stored, add: $+ \uparrow W \div 1,024 \uparrow$
	Back-end server	pd_work_buff_mode=each specified	$\{Q + g + (a + 9) \times c + i + m + r + t\} \times (b + 3) + y + S$ <ul style="list-style-type: none"> If you are performing in-memory data processing, add: $+ \uparrow \{T \times (b + 3)\} \div 1,024 \uparrow$ <ul style="list-style-type: none"> If you are changing the maximum number of communication traces stored, add: $+ \uparrow W \div 1,024 \uparrow$
		pd_work_buff_mode=pool specified or omitted	<ul style="list-style-type: none"> 32-bit mode $(Q + g + a + \uparrow a \div 128 \times 0.1 \uparrow + 11 + i + m + n + r + t) \times (b \times 3) + y + S$ <ul style="list-style-type: none"> 64-bit mode $(Q + g + a + \uparrow a \div 128 \times 0.1 \uparrow + 15 + i + m + n + r + t) \times (b \times 3) + y + S$ <ul style="list-style-type: none"> If you are performing in-memory data processing, add: $+ \uparrow \{T \times (b + 3)\} \div 1,024 \uparrow$

Item				Required memory (KB)
				<ul style="list-style-type: none"> If you are changing the maximum number of communication traces stored, add: + $\uparrow W \div 1,024 \uparrow$
Shared memory	Space used by the unit controller in the unit controller shared memory			$\uparrow d \div 1,024 \uparrow$
	Space used by each server in the unit controller shared memory ^{#1}			e
	Global buffer shared memory			F
	In-memory data processing shared memory			U
	Utility shared memory			u
	Security audit information buffer shared memory			<p>■ For automatic calculation by the system: $\uparrow 0.3 + \text{MAX}\{(R + 100), (R \times 1.2)\} \times 0.25 \uparrow$</p> <p>■ For user-specified values (specify the <code>pd_audit_def_buffer_size</code> operand): Value specified for <code>pd_audit_def_buffer_size</code></p>
	Inter-process memory communication shared memory ^{#3}			$j \times k$

#1: If the unit contains multiple servers (excluding the system manager), obtain the value for each server.

#2: When using plug-ins, add 300 per server process.

#3: Add this value if you have specified `PDIPC=MEMORY` in the client environment definition. For details about the inter-process memory communication facility and client environment definitions, see the *HiRDB Version 9 UAP Development Guide*. If either the HiRDB server or the HiRDB client is in 32-bit mode, the system allocates the shared memory for the inter-process memory communication facility in the 32-bit address space.

a: Value of `pd_work_buff_size` operand

b: Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand

- For a dictionary server, the value is (value of the `pd_max_dic_process` operand + value of the `pd_max_reflect_process_count` operand).
- For a back-end server, the value is (value of the `pd_max_bes_process` operand + value of the `pd_max_reflect_process_count` operand).
- If the `pd_max_dic_process` or `pd_max_bes_process` operand is omitted, the value is (value of the `pd_max_users` operand + value of the `pd_max_reflect_process_count` operand).

c: Maximum number of work tables

Find the number of work tables for each SQL statement in *Table 15-8 Procedure for obtaining the number of work tables for each SQL statement*. Use the largest number of work tables obtained from *Table 15-8 Procedure for obtaining the number of work tables for each SQL statement* as the maximum number of work tables.

d: Value obtained from *15.2.3 Formulas for shared memory used by a unit controller*.

e: Value obtained from *15.2.4 Formulas for shared memory used by each server*.

f: Value obtained from *15.2.5 Formula for size of shared memory used by global buffers*.

g: Size of memory required during SQL execution

For details about the formula, see *15.2.6 Formulas for size of memory required during SQL execution*.

h: Size of memory required during SQL preprocessing

For details about the formula, see *15.2.7 Formula for size of memory required during SQL preprocessing*.

i: LOB buffer batch input/output work memory

Add 62 KB if LOB global buffer is specified for the LOB RDAREA for the corresponding server (`-b` specified in the `pdbuffer` operand of the system common definition).

j: Maximum number of concurrently executable clients that use the inter-process memory communication facility.

If you are not sure about the value, specify the number of all clients that use the inter-process memory communication facility or the value of the `pd_max_users` operand.

k: Average memory size for data transfer performed by all clients that use the inter-process memory communication facility (value of `PDSENDMEMSIZE` + value of `PDRECVMEMSIZE` in the client environment definition).*m*: Memory requirement for a Java virtual machine

If you use Java stored procedures or Java stored functions, add the size of memory used by the Java virtual machine. This value depends on the Java virtual machine's options (`-Xms`, `-Xmx`, and `-Xmn` options for Hewlett-Packard JRE 1.2.2.04) and version. For details about the memory requirement for your Java virtual machine, see the applicable manual. Following are the guidelines for the memory required for HP-UX:

- Eight MB of memory is required to start a Java virtual machine.
- Add the maximum memory size for the Java virtual machine (value of the `-Xmx` option). Note that some Java virtual machines may use more memory than the size specified in the `-Xmx` option.

n: Work table extended memory size

When the `pd_work_buff_expand_limit` operand is specified, add the work table extended memory size. The work table extended memory size is determined from the following formula:

Work table extended memory size (kilobytes) = work table extended buffer size + \uparrow (work table extended buffer size \div 128) \times 0.1 \uparrow

- Work table extended buffer size (kilobytes) = MAX(0, work table extended buffer size based on hash join, subquery hash execution) + MAX(0, work table extended buffer size based on the increase in the number of work tables)
- Work table extended buffer size based on hash join, subquery hash execution = MIN{ (work table extended buffer size based on hash join, subquery hash execution - value of the `pd_work_buff_size` operand), (value of the `pd_work_buff_expand_limit` operand - value of the `pd_work_buff_size` operand) } \times number of concurrently executing users executing hash join, subquery hash execution

For details about determining the work table extended buffer size when executing hash joins, subquery hash executions, see the *HiRDB Version 9 UAP Development Guide*.

- Work table extended buffer size based on the increase in the number of work tables = MIN{ (number of work tables used \times 128 - value of the `pd_work_buff_size` operand), (value of `pd_work_buff_expand_limit` operand - value of `pd_work_buff_size` operand) } \times (number of users such that the number of work tables is greater than the value of the `pd_work_buff_size` operand \div 128)

Number of work tables used = MAX(number of work table files used per SQL statement, number of work table files used by the `ASSIGN LIST` statement)

For details about determining the number of work table files used per SQL statement and the number of work table files used by the `ASSIGN LIST` statement, see *18.3 Determining the maximum number of files (pdfmkfs -l command)*.

p: Memory requirements required for BLOB data type

For details about the formula, see *15.2.8 Formula for size of memory required during BLOB data retrieval or updating (front-end server)*.

q: Memory requirements required for server-side block transfer or array `FETCH`

For details about the formula, see *15.2.10 Formula for size of memory required during block transfer or array `FETCH` (front-end server)*.

r: Memory requirements required for BLOB data type

For details about the formula, see *15.2.9 Formula for size of memory required during BLOB data retrieval or updating (back-end server or dictionary server)*.

s: Memory size used by asynchronous `READ`

This is applicable when the asynchronous `READ` facility is used; use the following formula (in kilobytes) for the calculation:

(90 +

90

\sum Memory used by the RDAREA for management of the HiRDB file system area)

$i=1$

\times value of `pd_max_ard_process`

For the memory used by the RDAREA for management of the HiRDB file system area, use the largest 90 areas in descending order of the values. If the number of areas used by the server is fewer than 90, assume that amount anyway.

The memory used by the RDAREA for management of the HiRDB file system area (in kilobytes) is calculated from the formula below based on the initial settings:

Note that the parameters at the time of initialization of the area can be checked by running the `pdfstatfs` command with the `-A` option specified.

$\{(Number\ of\ files^{#1} + number\ of\ extensions^{#2}) \div 64\} \times 1.5^{#3}$

#1: Value specified by `pdfmkfs -l`, or displayed in available file count in the execution results of the `pdfstatfs` command.

#2: Value specified by `pdfmkfs -e`, or displayed in available expand count in the execution results of the `pdfstatfs` command.

#3: Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2,048.

t: HiRDB file system memory size

Determine with the following formula (in kilobytes):

347 + Memory used by the work tables for management of the HiRDB file system area + Memory used by the system logs for management of the HiRDB file system area +

90

\sum memory used by the RDAREA for management of the HiRDB file system area

$i=1$

The memory used by the HiRDB file system area for management of work tables and system logs uses the maximum value calculated for the memory used by the HiRDB file system area for management used by the server. For RDAREAs, use the largest 90 areas in descending order of the values. If the number of areas used by the server is fewer than 90, use as many areas as are used for the calculation.

The memory used by the RDAREA for management of the HiRDB file system area (in kilobytes) is calculated with the formula below based on the initial settings:

Note that the parameters at the time of initialization of the area can be checked by running the `pdfstatfs` command with the `-A` option specified.

$\{(Number\ of\ files^{#1} + number\ of\ extensions^{#2}) \div 64\} \times 1.5^{#3}$

#1: Value specified by `pdfmkfs -l`, or displayed in available file count in the execution results of the `pdfstatfs` command.

#2: Value specified by `pdfmkfs -e`, or displayed in available expand count in the execution results of the `pdfstatfs` command.

#3: Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2,048.

u: When value of `pd_utl_exec_mode` is 0: 0

When value of `pd_utl_exec_mode` is 1: $\uparrow \{(b \times 2,000 + 136) \div 1,024\} \uparrow \times 1,024$

v: Value of `pd_module_trace_max` that is valid as the unit control information definition

w: Sum of (maximum number of processes that can be started + 3) for all server processes in the unit

For details about the maximum number of processes that can be started, see the manual *HiRDB Version 9 System Definition*.

y: Sum of the values obtained by the following formula for each server process in the unit:

In the 32-bit mode:

$\uparrow \{(64 + 48 \times (\text{value of } pd_module_trace_max + 1)) \times (\text{maximum number of processes that can be started} + 3)\} \div 1,024 \uparrow$

In the 64-bit mode:

$\uparrow \{(64 + 64 \times (\text{value of } pd_module_trace_max + 1)) \times (\text{maximum number of processes that can be started} + 3)\} \div 1,024 \uparrow$

For details about the maximum number of processes that can be started, see the manual *HiRDB Version 9 System Definition*.

z: Memory size for restarting HiRDB

If this memory size cannot be allocated, HiRDB restart fails. Use the following formula to determine the size (in bytes):

$$(D + E + F) \times \text{number of dictionary servers} + (D + E + F + G) \times \text{number of back-end servers} + \sum H$$

Use the following variables for the formula to calculate the size of memory used by HiRDB to restart:

Variable	Value
<i>D</i>	<ul style="list-style-type: none"> 32-bit mode $246,762 + 4 \times \text{value of } pd_max_rdarea_no$ $+ \{48 \times (\text{value of } pd_max_rdarea_no + \text{number of tables}) + 304\} \times ((\text{value of } pd_max_users^{\#} + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$ <ul style="list-style-type: none"> 64-bit mode $305,274 + 8 \times \text{value of } pd_max_rdarea_no$ $+ \{64 \times (\text{value of } pd_max_rdarea_no + \text{number of tables}) + 512\} \times ((\text{value of } pd_max_users^{\#} + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$ <p>Number of tables: $62 + \text{MAX}\{\text{value of } pd_max_access_tables, 500\}$</p>
<i>E</i>	$b1 \times X + b2 \times Y$ <p><i>b1</i>: When the record length of the server status file $< 4,096$</p> $\text{MAX}((\downarrow(3,400 \div ((\downarrow((\text{record length} - 40) - 308) \div 20) \downarrow))$ $+ (\downarrow(\text{record length} - 40) \div 20) \downarrow) \times (\text{MAX}(\downarrow(4,096 \div \text{record length} \downarrow, 2) - 1))$ $+ 0.7) \downarrow, 1) \times \text{MAX}(\downarrow(4,096 \div \text{record length} \downarrow, 2) \times (\text{record length} - 40))$ <p>When $4,096 \leq \text{record length of server status file} < 12,288$</p> $\text{MAX}(\downarrow(3,400 \div (\downarrow(((\text{record length} - 40) - 308) \div 20) \downarrow) + 0.7) \downarrow, 1)$ $\times (\text{record length} - 40)$ <p>When $12,288 \leq \text{record length of server status file}$</p> $\text{MAX}(\downarrow(3,400 \div (\downarrow(((\text{record length} - 40) - 836) \div 20) \downarrow) + 0.7) \downarrow, 1)$ $\times (\text{record length} - 40)$ <p><i>X</i>: When the number of RDAREAS in server $\leq 3,400$: 1</p> <p>When $3,401 \leq \text{number of RDAREAS in server} \leq 6,800$: 2</p> <p>When $6,801 \leq \text{number of RDAREAS in server}$: 3</p> <p><i>b2</i>: When the record length of the status file for server $< 4,096$</p> $(\downarrow(5,662,310 \div ((\downarrow((\text{record length} - 40) - 308) \div 20) \downarrow))$ $+ (\downarrow(\text{record length} - 40) \div 20) \downarrow) \times (\text{MAX}(\downarrow(4,096 \div \text{record length} \downarrow, 2) - 1))$ $+ 0.7) \downarrow) \times \text{MAX}(\downarrow(4,096 \div \text{record length} \downarrow, 2) \times (\text{record length} - 40))$ <p>When $4,096 \leq \text{record length of server status file} < 12,288$</p> $\downarrow(5,662,310 \div (\downarrow(((\text{record length} - 40) - 308) \div 20) \downarrow) + 0.7) \downarrow$ $\times (\text{record length} - 40)$ <p>When $12,288 \leq \text{record length of server status file}$</p> $\downarrow(5,662,310 \div (\downarrow(((\text{record length} - 40) - 836) \div 20) \downarrow) + 0.7) \downarrow$ $\times (\text{record length} - 40)$ <p><i>Y</i>: When the number of RDAREAS in server $\leq 10,200$: 0</p>

Variable	Value
	When $10,201 \leq \text{number of RDAREAS in server} \leq 5,672,510$: 1 When $5,672,511 \leq \text{number of RDAREAS in server} \leq 11,334,820$: 2 When $11,334,821 \leq \text{number of RDAREAS in server}$: 3
<i>F</i>	If <code>commit</code> is specified in the <code>pd_dbsync_point</code> operand, add: $+ 112 \times ((\text{value of } pd_max_users^{\#} + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$
<i>G</i>	If 1 or a greater value is specified in the <code>pd_inner_replica_control</code> operand, add: <ul style="list-style-type: none"> 32-bit mode $(48 \times \text{value of } pd_max_rdarea_no + 80) \times ((\text{value of } pd_max_users^{\#} + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$ <ul style="list-style-type: none"> 64-bit mode $(64 \times \text{value of } pd_max_rdarea_no + 160) \times ((\text{value of } pd_max_users^{\#} + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$
<i>H</i>	For back-end servers in which the number of HiRDB file system areas that store RDAREAs created in a character special file in a server is 1,001 or more, add: <ul style="list-style-type: none"> 32-bit mode $12,012 \times (\lceil (\text{number of HiRDB file system areas that store RDAREAs created in a character special file} - 1,000) \div 1,000 \rceil)$ <ul style="list-style-type: none"> 64-bit mode $16,016 \times (\lceil (\text{number of HiRDB file system areas that store RDAREAs created in a character special file} - 1,000) \div 1,000 \rceil)$

#

For a dictionary server, use the value of `pd_max_dic_process`. For a back-end server, use the value of `pd_max_bes_process`. However, if both `pd_max_dic_process` and `pd_max_bes_process` are omitted, use the value of `pd_max_users`

A: $425 \times (2 \times b + 7) \times \text{Number of FESs in unit}$

B: $425 \times (2 \times b + 7) \times \text{Number of DSs in unit}$

C: $425 \times (2 \times b + 7) \times \text{Number of BESSs in unit}$

J, K, L, M, N, P, Q: Fixed value

These values depend on the OS being used. The following table presents the values for each OS (in kilobytes):

OS	Value of J	Value of K	Value of L	Value of M	Value of N	Value of P	Value of Q
HP-UX (32-bit mode)	80,000	9,700	12,600	2,800	4,700	3,800	3,600
HP-UX (32-bit-mode POSIX library version)	127,300	10,300	12,900	3,000	5,200	4,300	5,000
HP-UX (64-bit mode)	80,700	10,300	13,400	2,800	5,100	4,300	4,000
HP-UX (IPF)	200,500	15,100	17,900	2,600	7,100	8,100	10,600
AIX (32-bit mode)	67,700	8,400	10,700	2,200	5,100	4,100	4,300
AIX (32-bit-mode POSIX library version)	123,700	16,800	19,200	5,000	7,900	7,300	7,400
AIX (64-bit mode)	123,700	25,000	31,200	6,000	11,900	11,700	13,400
Solaris (32-bit mode)	71,100	6,100	7,600	1,400	2,600	2,100	4,100

OS	Value of J	Value of K	Value of L	Value of M	Value of N	Value of P	Value of Q
Solaris (32-bit-mode POSIX library version)	121,100	6,800	8,200	1,700	2,800	2,300	4,400
Solaris (64-bit mode)	106,900	9,700	11,900	12,100	4,100	3,300	6,400
Linux (32-bit mode)	74,200	6,700	8,800	1,700	2,900	4,300	4,300
Linux (EM64T)	134,600	24,100	31,700	7,600	13,300	15,400	14,700
Linux 6 (64-bit x86_64)	106,400	12,300	14,600	1,900	5,400	5,800	7,200

R: If you are making a rough estimate, this is the number of audit events (the number of executions of CREATE AUDIT). If you are making a detailed estimate, it is the number of entries in the security audit information buffer

S: Memory required when using the facility for acquiring syncpoint output synchronization control information (bytes)
If 1 is specified in the `pd_dbbuff_trace_level` operand and the `pd_dfw_awt_process` operand is not specified, add:

32-bit mode

$320 \times \text{number of global buffers defined in a single server}$

64-bit mode

$640 \times \text{number of global buffers defined in a single server}$

T: If 1 or a greater value is specified in the `pd_max_resident_rdarea_no` operand, add:

$1,648 + 16 \times \text{value of } pd_max_resident_rdarea_no + 16 \times \text{value of } pd_max_resident_rdarea_shm_no$

U: Memory required by in-memory data processing

For the applicable formulas, see *15.2.11 Memory required by in-memory data processing*.

V: Memory required by communication trace processing

32-bit mode

$(16 \times (Z - 1,024) \times 2) \times (\text{value of } pd_max_server_process - w)$

64-bit mode

$(32 \times (Z - 1,024) \times 2) \times (\text{value of } pd_max_server_process - w)$

W: Memory required by communication trace processing

This is either of the following values calculated for each server process within the unit.

32-bit mode

$(16 \times (aa - 1,024) \times 2) \times (\text{maximum number of startup processes} + 3)$

64-bit mode

$(32 \times (aa - 1,024) \times 2) \times (\text{maximum number of startup processes} + 3)$

For details about the maximum number of startup processes, see the manual *HiRDB Version 9 System Definition*.

Z: The value of `pd_pth_trace_max` enabled as the unit control information definition.

The value specified for the operand rounded up to a power of two.

aa: The value of `pd_pth_trace_max` enabled as each server definition.

The value specified for the operand rounded up to a power of two.

ab: Size of memory for signal handler

HP-UX (IPF) version: 768

Otherwise: 0

ac: If the operand `pd_module_trace_max` is specified in the system common definition or the unit control information definition: value of `pd_module_trace_max`

Otherwise: 16,383

Table 15–8: Procedure for obtaining the number of work tables for each SQL statement

SQL statement	Procedure for obtaining the number of work tables
SELECT statement	When none of 1-8 as follows are applicable: 0
INSERT (–SELECT) statement	When any of 1-8 as follows are applicable: Sum of the applicable values from 1-8 <ol style="list-style-type: none"> When multiple tables are joined for retrieval Number of additional work tables = (Number of joined tables - 1) \times 2 + 1 When specifying the ORDER BY clause Number of additional work tables = 2 When specifying the GROUP BY clause Number of additional work tables = Number of GROUP BY clauses specified When specifying the DISTINCT clause Number of additional work tables = Number of DISTINCT clauses specified When specifying the UNION, UNION ALL, or EXCEPT [ALL] clause Number of additional work tables = (Number of UNION or UNION ALL clauses specified) \times 2 + 1 When search condition contains columns with index defined Number of additional work tables = Number of columns with index defined in the search condition When specifying the FOR UPDATE or FOR READ ONLY clause Number of additional work tables = 1 When specifying a subquery (quantified predicate) Number of additional work tables = Number of subqueries specified
UPDATE statement	Number of columns with index defined in the search condition + 1
DELETE statement	
DROP SCHEMA statement	2
DROP TABLE statement	
DROP INDEX statement	
CREATE INDEX statement	
REVOKE statement to revoke access privilege	

15.2.3 Formulas for shared memory used by a unit controller

(1) 32-bit mode HiRDB

The size of memory required for the unit controller in each server machine from startup to termination of the unit is the sum of the items listed as follows:

Ensure that the size of the shared memory within the entire controller does not exceed 2 gigabytes.

Process item	Shared memory calculation formula (bytes)
Scheduler	<p>Value of <code>pd_utl_exec_mode</code> set to 0:</p> $\{ \uparrow(432 + 304 \times n) \div 1,024 \uparrow + 494 + x + z + \uparrow(134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \uparrow \} \times 1,024$ <p>Value of <code>pd_utl_exec_mode</code> set to 1:</p> $\{ \uparrow(432 + 304 \times n) \div 1,024 \uparrow + \uparrow(s \times 2,000 + 136) \div 1,024 \uparrow + y + z + \uparrow(134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \uparrow \} \times 1,024$ <p>x: Unit contains MGR: 37 Unit contains FES: $57 + 1 \times (s + 3) + 14$ Unit contains DS: $102 + 5 \times (t + 3) + 14$</p>

Process item	Shared memory calculation formula (bytes)
	<p>Unit contains BES: $\{192 + 9 \times (u + 3) + 14\} \times (\text{number of BESs} + \beta + \gamma)$</p> <p>y: Unit contains MGR: 0</p> <p>Unit contains FES: $1 \times (s + 3) + 14$</p> <p>Unit contains DS: $5 \times (t + 3) + 14$</p> <p>Unit contains BES: $\{9 \times (u + 3) + 14\} \times (\text{number of BESs} + \beta + \gamma)$</p> <p>z: Unit subject to standby-less system switchover (effects distributed):</p> <p>$\uparrow 64 + \{(\text{Number of BESs in unit} + \text{number of acceptable BESs}^\#) \times 48\} \div 1,024 \uparrow$</p> <p>Other unit: 0</p> <p>n: Number of servers in unit + $\beta + \gamma$ + number of utility servers in unit + 1</p> <p>Number of utility servers in unit: $24 + \alpha$</p> <p>α: If the unit controller has a MGR: 3</p> <p>If the unit controller contains a FES: 3</p> <p>If the unit controller contains a DS: 7</p> <p>If the unit controller contains BESs: $(\text{number of BESs} + \beta) \times 15$</p> <p>s: Value of pd_max_users + value of pd_max_reflect_process_count</p> <p>t: Value of pd_max_dic_process + value of pd_max_reflect_process_count</p> <p>u: Value of pd_max_bes_process + value of pd_max_reflect_process_count</p> <p>β: Unit subject to standby-less system switchover (effects distributed):</p> <p>Number of acceptable BESs[#]</p> <p>Other unit: 0</p> <p>γ: Unit subject to standby-less system switchover (1:1):</p> <p>Number of alternate BESs</p> <p>Other unit: 0</p> <p>#: Value of pd_ha_max_guest_act_servers</p>
Lock server	<p>If a server (FES, BES, or DS) exists in the unit:</p> <p>For the values of the operands that are used for each server to determine the value for guest BESs in the unit subject to standby-less system switchover (effects distributed) (such as pd_lck_pool_size, pd_lck_pool_partition, pd_lck_hash_entry, and pd_max_bes_process), use the maximum value among those specified for all guest BESs in that unit, not the operand values for a particular guest BES. In the unit that is subject to standby-less system switchover (effects distributed), <i>servers in the unit</i> means <i>all host BESs + all guest BESs</i>.</p> <p>In a unit subject to standby-less system switchover (1:1), <i>servers in the unit</i> means <i>all normal BESs + all alternate BESs</i>.</p> <p>y</p> <p>$\sum_{x=1} \{$</p> <p>$320 + 48 + c_x + d_x + 48 + 4,096 + g_x + 48 + i_x$</p> <p>$+ 48 + 12,252 + 48 + n_x + p_x + t_x + u_x + 16$</p> <p>$\} \times \text{value of } pd_lck_pool_partition^\#$</p> <p>#: For a FES, the value of pd_fes_lck_pool_partition</p> <p>x: Server serial number in the unit</p> <p>y: Number of servers in the unit</p> <p>c_x: If pd_lck_hash_entry is omitted or 0 is specified:</p> <p>For a FES with pd_fes_lck_pool_size omitted:</p> <p>$(\downarrow (8 + 4 \times \text{MAX}(\text{largest prime number} < \uparrow (\downarrow (p_x + 3) \times (\text{value of } pd_max_access_tables + 4) \div 6 \downarrow$</p> <p>$\div \text{value of } pd_fes_lck_pool_partition \downarrow \times 6) \div 10 \uparrow, 11,261)) \div 16 \downarrow + 1) \times 16$</p> <p>For a FES with pd_fes_lck_pool_size omitted:</p>

Process item	Shared memory calculation formula (bytes)
	$(\downarrow(8+4 \times \text{MAX}(\text{largest prime number} < \uparrow(\downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 6) \div 10 \uparrow, 11,261)) \div 16 \downarrow + 1) \times 16$ <p>For a BES or DS:</p> $(\downarrow(8+4 \times \text{MAX}(\uparrow((p_x+3) \times 2 \times 5 + \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 6) \div 10 \uparrow, 11,261)) \div 16 \downarrow + 1) \times 16$ <p>When 2 or a larger non-prime number is specified for <code>pd_lck_hash_entry</code>:</p> $(\downarrow(8+4 \times \text{largest prime number} < \text{value of } pd_lck_hash_entry) \div 16 \downarrow + 1) \times 16$ <p>If 1 or a prime number is specified for <code>pd_lck_hash_entry</code>:</p> $(\downarrow(8+4 \times \text{value of } pd_lck_hash_entry) \div 16 \downarrow + 1) \times 16$ <p>d_x: For a FES with <code>pd_fes_lck_pool_size</code> omitted:</p> $(\downarrow \downarrow (p_x+3) \times (\text{value of } pd_max_access_tables + 4) \div 6 \downarrow \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 6) \times 96$ <p>For a FES with <code>pd_fes_lck_pool_size</code> specified:</p> $\downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 6 \times 96$ <p>For a BES or DS:</p> $((p_x+3) \times 2 \times 5 + \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 6) \times 96$ <p>g_x: For a FES:</p> $(p+3) \times 2 \times 256$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $((p+3) \times 2 + s) \times 256$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $((p+3) \times 2 + 32) \times 256$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $((p+3) \times 2 + s) \times 256$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $((p+3) \times 2 + 16) \times 256$ <p>i_x: For a FES with <code>pd_fes_lck_pool_size</code> omitted:</p> $(\downarrow \downarrow (p_x+3) \times (\text{value of } pd_max_access_tables + 4) \div 6 \downarrow \div \text{value of } pd_fes_lck_pool_partition \downarrow) \times 12 \times 64$ <p>For a FES with <code>pd_fes_lck_pool_size</code> specified:</p> $(\downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 8) \text{ rounded up to an even number} \times 64$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8 + (p_x+3) \times 2 \times 2 \times 5 + s \times (\text{value of } pd_max_rdarea_no + 1)) \text{ rounded up to an even number} \times 64$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8 + (p_x+3) \times 2 \times 2 \times 5 + 32 \times (\text{value of } pd_max_rdarea_no + 1)) \text{ rounded up to an even number} \times 64$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8$

Process item	Shared memory calculation formula (bytes)
	$+ (p_x + 3) \times 2 \times 2 \times 5 + s + 4) \text{ rounded up to an even number} \times 64$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8$ $+ (p_x + 3) \times 2 \times 2 \times 5 + 20) \text{ rounded up to an even number} \times 64$ <p>n_x: For a FES:</p> $(p_x + 3) \times 2 \times 48$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $((p_x + 3) \times 2 \times 17 + s) \times 48$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $((p_x + 3) \times 2 \times 17 + 32) \times 48$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $((p_x + 3) \times 2 \times 17 + s) \times 48$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $((p_x + 3) \times 2 \times 17 + 16) \times 48$ <p>p_x: For a FES if the number of FESs in the HiRDB system > 1: $s + 1$ For a FES, if the number of FESs in the HiRDB system = 1: s</p> <p>For a BES with $s > \text{value of } pd_max_bes_process$: s For a BES with $s \leq \text{value of } pd_max_bes_process$: $\text{value of } pd_max_bes_process \text{ operand}$ $+ \text{value of } pd_max_reflect_process_count \text{ operand}$</p> <p>For a DS with $s > \text{value of } pd_max_dic_process$: s For a DS with $s \leq \text{value of } pd_max_dic_process$: $\text{value of } pd_max_dic_process \text{ operand}$ $+ \text{value of } pd_max_reflect_process_count \text{ operand}$</p> <p>$s$: $\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count$</p> <p>$t_x$: For a FES:</p> $48 + (p_x + 3) \times 2 \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $48 + ((p_x + 3) \times 2 + s) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $48 + ((p_x + 3) \times 2 + 32) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $48 + ((p_x + 3) \times 2 + s) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $48 + ((p_x + 3) \times 2 + 16) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>u_x: For a FES with <code>pd_fes_lck_pool_size</code> omitted:</p> $48 + (\downarrow \downarrow (p_x + 3) \times (\text{value of } pd_max_access_tables + 4) \div 6 \downarrow \div \text{value of } pd_fes_lck_pool_partition \downarrow) \times 12$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a FES with <code>pd_fes_lck_pool_size</code> specified:</p> $48 + \downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 8$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8$ $+ (p_x + 3) \times 2 \times 2 \times 5 + s \times (\text{value of } pd_max_rdarea_no + 1))$

Process item	Shared memory calculation formula (bytes)
	<p>rounded up to an even number $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <p>For a BES with the value of $pd_utl_exec_mode$ set to 0 or $s \leq 32$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8$ $+ (p_x + 3) \times 2 \times 2 \times 5 + 32 \times (\text{value of } pd_max_rdarea_no + 1))$ <p>rounded up to an even number $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <p>For a DS with the value of $pd_utl_exec_mode$ set to 1 and $s > 16$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8$ $+ (p_x + 3) \times 2 \times 2 \times 5 + s + 4) \text{ rounded up to an even number}$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For DS with the value of $pd_utl_exec_mode$ set to 0 or $s \leq 16$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 8$ $+ (p_x + 3) \times 2 \times 2 \times 5 + 20) \text{ rounded up to an even number}$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>■ If no server (FES, BES, or DS) exists in the unit</p> <p>8,416</p>
Transaction server	$288 + 32 \times B + 192 \times s \times 2$ <p>If the unit contains FES, add the following value:[#]</p> $+ 1,028 + (420 + 564 + 256 + 384 \times 2) \times (A \times 2 + 7) + 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ C$ <p>If the unit contains BES, add the following value:[#]</p> $+ 1,028 + (420 + 564 + 256 + 384 \times 2)$ $\times (u \times 2 + 7) + 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ D$ <p>If the unit contains DS, add the following value:[#]</p> $+ 1,028 + (420 + 564 + 256 + 384 \times 2)$ $\times (t \times 2 + 7) + 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ E$ <p>s: Value of pd_max_users + value of $pd_max_reflect_process_count$</p> <p>$t$: Value of $pd_max_dic_process$ + value of $pd_max_reflect_process_count$</p> <p>$u$: Value of $pd_max_bes_process$ + value of $pd_max_reflect_process_count$</p> <p>$A$: For a multi-FES system: $s + 1$</p> <p>If not a multi-FES system: s</p> <p>B: Unit subject to standby-less system switchover (effects distributed):</p> <p>Number of host BESs + $pd_ha_max_act_guest_servers$ operand correction value</p> <p>Other unit: Number of servers in the unit</p> <p>C: If the unit meets one condition in <i>Conditions</i> below:</p> $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ <p>If the unit does not match both of the conditions in <i>Conditions</i> below: 0</p> <p>D: If the unit meets one condition in <i>Conditions</i> below:</p> $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$

Process item	Shared memory calculation formula (bytes)
	<p>+ number of FESs in the system) $\times (u \times 2 + 7)$</p> <p>If the unit does not meet both conditions in <i>Conditions</i> below: 0</p> <p><i>E</i>: If the unit meets one condition in <i>Conditions</i> below:</p> <p>$128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$</p> <p>+ number of FESs in the system) $\times (t \times 2 + 7)$</p> <p>If the unit does not meet both conditions in <i>Conditions</i> below: 0</p> <p>Conditions</p> <ul style="list-style-type: none"> • uap is specified in the pd_rpl_reflect_mode operand. • The pdstart -k stls operand is specified for a front-end server in the system. <p>#: Add the value calculated by the above formula once for each server.</p> <p>For units subject to standby-less system switchover (1:1):</p> <p>Number of normal BESs + Number of alternate BESs</p> <p>Other than the above, the number of the servers in the unit.</p>
Timer server	<p>$32 \times (\text{value of pd_max_users} + \text{value of pd_max_reflect_process_count} + 3)$</p> <p>$\times (\text{number of BESs in system} + 1 + \text{number of utility servers in unit} + 1)$</p> <p>+ 1,440</p> <p>Number of utility servers in unit is $23 + \alpha$</p> <p>α: When there is MGR in the unit: 2</p> <p>When there is FES in the unit: 3</p> <p>When there is DS in the unit: 7</p> <p>When there is BES in the unit: Number of BESs $\times b$</p> <p>b: For HP-UX, Solaris, and AIX: 15</p> <p>For Linux: 6</p>
Statistics log server	<p>$384 + 128 \times 16 + 32 + 288 \times 2 + 1,024 + 128 \times 3$</p> <p>+ value of pd_stj_buff_size $\times 1,024 \times 3 + 64 + 4,096 + 8,192$</p>
Process server	<p>$192 + 512 \times \text{MAX}(c, 256) + 80 + 256 + (\text{value of pd_max_server_process} + 50) \times (256 + 144) + 16 + 8$</p> <p>$\times 34 + 16 + 16 + 48 + 48 \times (k + 1)$</p> <p>$c$: $\uparrow(51 + d + e + f + (g \times \text{number of BESs in the unit}^\#) + h + i) \div 16 \uparrow \times 16$</p> <p>$d$: If the unit contains MGR, 59; if not, j.</p> <p>e: If the unit contains DS, 17; if not, 0.</p> <p>f: If the unit contains FES, 11; if not, 0.</p> <p>g: If the unit contains BES, 25; if not, 0.</p> <p>h: If the standby-less system switchover (1:1) facility is used, 9; if not, 0.</p> <p>i: If the unit is subject to standby-less system switchover (effects distributed), 1; if not, 0.</p> <p>j: If manager is specified in the pd_mlg_msg_log_unit operand, 1; if local is specified, 2.</p> <p>k: If the operand pd_module_trace_max is specified in the system common definition or the unit control information definition, the value of pd_module_trace_max; if not, 16,383.</p> <p>#: If the unit is subject to standby-less system switchover (1:1), the value is (number of BESs $\times 2$). If the unit is subject to standby-less system switchover (effects distributed), the value includes the pd_ha_max_act_guest_servers operand correction value.</p>
System manager	<p>$704 + (44 + 4) \times (g + h + i) + (100 + 4) \times \{(p + q + 3) + u \times (15 + 1)\} + (92 + 4) \times c + 40 \times (k + m + n \times o + u) \times 14 + 256 \times m + 128 \times c + 36 \times d + 12 \times e + 96 \times o + v \times (16 \times 35 \times (k + u) + 15 + 36 \times z + 15) + w \times (48 \times B + 15 + 4 \times z + 15 + 4 \times y + 15) + v \times (132 + 15) + 8 + 5,844 + s + s \times o + 16 + 96 \times o + 1,024 + 272 \times A$</p> <p>$c$: Number of units</p> <p>d: Number of -c options specified in pdunit operand</p>

Process item	Shared memory calculation formula (bytes)
	<p>e: Number of <code>pdcltgrp</code> operands specified</p> <p>g: Number of FESs in the system</p> <p>h: Number of BESSs in the system</p> <p>i: Number of DSs in the system</p> <p>j: Number of FESs in unit</p> <p>k: Number of BESSs in unit</p> <p>m: Number of DSs in the unit</p> <p>n: Number of alternate BESSs in the unit</p> <p>o: If the unit is subject to standby-less system switchover (1:1), 1; if not 0</p> <p>p: $i + k + m + n$</p> <p>q: $24 + t + j \times 3 + k \times 15 + m \times 7$</p> <p>$r$: $14 \times (k + m + u) + p + q + u \times 15 + 2 + 38 + 5 + p \times 4$</p> <p>$s$: $224 + 2,052 + 128 \times (r + 3) + v \times (40 \times (k + u) + 72 \times (k + u))$</p> <p>$t$: If the unit contains MGR, 2; if not, 0</p> <p>u: Number of acceptable BESSs (value of the <code>pd_ha_max_act_guest_servers</code> operand)</p> <p>v: If the unit is subject to standby-less system switchover (effects distributed), 1; if not, 0</p> <p>w: If there is a unit that is subject to standby-less system switchover (effects distributed), 1; if not, 0</p> <p>y: Sum of the number of units in the HA group</p> <p>z: Sum of the number of servers in the HA group</p> <p>A: Number of IP addresses for the host specified in the <code>pd_security_host_group</code> operand</p> <p>If the <code>pd_security_host_group</code> operand is not specified, 0</p> <p>B: Number of <code>pdhagroup</code> operands specified in the system common definition</p>
Name server	<p>$\text{MAX}\{65,536, (X + Y + Z)\} + \text{MAX}(16,384, L) + M$</p> <p>$X$: $\uparrow(256 + 16 + 156 \times \text{number of units} + 16 + 16 \times 126) \div 1,024 \uparrow \times 1,024$</p> <p>$Y$: 8,192</p> <p>$Z$: $\uparrow(264 \times (Z_2 + Z_3 + j + 32)) \div 1,024 \uparrow \times 1,024$</p> <p>$Z_2$: $b + 10 + c + 11 \times \text{number of HiRDB servers within local unit} + d + e$</p> <p>$Z_3$: $f + 7 + g + 4 \times \text{number of HiRDB servers within local unit} + h + i$</p> <p>$L$: $\uparrow(224 \times (L_2 + L_3 + L_4)) \div 1,024 \uparrow \times 1,024$</p> <p>$L_2$: $k + 2 \times \text{number of units within system}$</p> <p>$L_3$: Number of BESSs within system + number of FESs within system + number of DSs within system</p> <p>L_4: $15 \times \text{number of HiRDB servers within system}$</p> <p>$M$: Number of HiRDB servers within unit + $z + m \times \text{number of system servers within unit} \times 1,024 \div 1,024$</p> <p>$b$: If the unit has a MGR: 3</p> <p>If the unit does not have a MGR: 0</p> <p>c: For units subject to standby-less system switchover (1:1): 2</p> <p>For other units: 0</p> <p>d: For units subject to standby-less system switchover (effects distributed):</p> <p>$11 \times \text{number of guest BESSs that can be accepted}$</p> <p>For other units: 0</p> <p>e: For units subject to standby-less system switchover (1:1):</p> <p>$6 \times \text{number of HiRDB servers within local unit}$</p> <p>For other units: 0</p> <p>f: If the unit has a MGR: 3</p> <p>If the unit does not have a MGR: 0</p>

Process item	Shared memory calculation formula (bytes)
	<p>g: For units subject to standby-less system switchover (1:1): 2 For other units: 0</p> <p>h: For units subject to standby-less system switchover (effects distributed): $3 \times$ number of guest BESs that can be accepted For other units: 0</p> <p>i: For units subject to standby-less system switchover (1:1): $4 \times$ number of HiRDB servers within local unit For other units: 0</p> <p>j: Number of servers in unit + β + γ + number of utility servers in unit + 2 Number of utility servers in unit: $23 + \alpha$</p> <p>k: If the unit has a MGR: 3 If the unit does not have a MGR: 0</p> <p>m: $38 + n + o$</p> <p>n: If the unit has a MGR and the number of units is 2 or more: 5 If the unit has a MGR and the number of units is 1: 4 If the unit does not have a MGR and the value of <code>pd_mlg_msg_log_unit</code> is local: 4 If the unit does not have a MGR and the value of <code>pd_mlg_msg_log_unit</code> is manager: 3</p> <p>o: $4 \times$ number of HiRDB servers within unit z: $23 + \alpha$ α: If the unit has a MGR: 3 If the unit has a FES: 3 If the unit has a DS: 7 If the unit has a BES: (number of BESs + β) \times 15</p> <p>β: For units subject to standby-less system switchover (effects distributed): Number of guest BESs that can be accepted[#] For other units: 0</p> <p>γ: For units subject to standby-less system switchover (1:1): number of alternate BESs For other units: 0</p> <p>$\#$: Value of <code>pd_ha_max_guest_act_servers</code></p>
Node manager	<p>Unit contains MGR: $\uparrow (1,152 + 432 \times \text{total number of units in the system} + 80 \times \text{total number of servers in the system} + 7,680 + 1,008 + 56 \times C + 240 \times A + 44 \times A + 28 \times A + 16 \times B + 16 \times \text{total number of BESs in the system} + 8 \times \text{total number of units in the system} + 64 \times \text{total number of servers in system} + 32 \times \text{total number of servers in system} + 32) \div 1,024 \uparrow \times 1,024$</p> <p>Unit contains no MGR: $\uparrow (1,008 + 56 \times C + (240 \times A + 44 \times A + 28 \times A) \times F + 16 \times B + 16 \times \text{total number of BESs in the system} + 8 \times \text{total number of units in the system} + 64 \times \text{total number of servers in system} + 32 \times \text{total number of servers in system} + 32) \div 1,024 \uparrow \times 1,024$</p> <p>$A$: Value of <code>pd_utl_exec_mode</code> = 0: 1,024 Value of <code>pd_utl_exec_mode</code> = 1: value of <code>pd_max_users</code> \times total number of BESs in system \times 3 If the unit contains MGR, add: Value of <code>pd_max_users</code> \times 4 + 200 If the unit contains DS, add: Value of <code>pd_max_users</code> \times 3 + 200 If the unit contains BESs, add: Value of <code>pd_max_users</code> \times D If the value of A obtained from the previous formula is not greater than 1,024, use 1,024.</p>

Process item	Shared memory calculation formula (bytes)
	<p><i>B</i>: <i>pdcltgrp</i> operand not specified: 0 <i>pdcltgrp</i> operand specified: Number of <i>pdcltgrp</i> operands specified + 1</p> <p><i>C</i>: Number of servers in the unit + <i>E</i> <i>D</i>: Number of BESs in unit + <i>E</i></p> <p><i>E</i>: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit Unit subject to standby-less system switchover (effects distributed): Number of acceptable BESs Other unit: 0</p> <p><i>F</i>: Unit subject to standby-less system switchover (1:1): 2 Other unit: 1</p>
I/O server	<p>$\uparrow (28 + (\uparrow (32 + A) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128$</p> <p>If the unit is not subject to standby-less system switchover (effects distributed):</p> <p><i>A</i>: $3,248 + (14 + 16) \times 972 + 1 \times 276 + (534 \times 276)^{\#1}$ $+ \{(534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972) \times \text{number of BESs}\}^{\#2}$ $+ \{534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972\}^{\#3}$ $\#1$ Add this value if FESs exist. $\#2$ Add this value if BESs exist. $\#3$ Add this value if DSs exist.</p> <p>In a unit subject to standby-less system switchover (1:1), double the value obtained in the above formula.</p> <p>If the unit is subject to standby-less system switchover (effects distributed):</p> <p><i>A</i>: $\uparrow 48 + 24 \times \text{number of BESs}^{\#4} \div 16 \uparrow \times 16$ $+ \uparrow (3,248 + 16 \times 972 + 534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972) \div 16 \uparrow \times$ $16 \times \text{number of BESs}^{\#4}$ $+ \uparrow (3,248 + (14 + 16) \times 972 + 1 \times 276) \div 16 \uparrow \times 16$ $\#4$ Includes the value of <i>pd_ha_max_act_guest_servers</i>.</p>
Log server	<p>$32 + 48 + 128 \times 37$</p> <p>$+ \{$ $384 + 128 \times 7 + 1,024 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of } pd_log_rec_leng^{\#} \uparrow$ $\times \text{value of } pd_log_rec_leng^{\#}$ $+ 64 + 4,096 \times 2 + (736 + 512) \times B$ $+ \uparrow \{(B + 1) \div 12\} \uparrow \times 8,320$ $+ 128 \times \text{value of } pd_log_write_buff_count^{\#}$ $+ (\text{value of } pd_log_write_buff_count^{\#} + A)$ $\times \uparrow \{\text{value of } pd_log_max_data_size^{\#} + (68 + 44 + 96 + 160)\} \div 4,096 \uparrow$ $\times 4,096 + C$ $\} \times \text{number of servers in the unit} + D + 128 \times \text{number of FESs in the unit}$ Add this if the <i>pd_max_reflect_process_count</i> operand is specified. $(128 + 704) \times (\text{number of BESs in the unit} + D)$ <i>A</i>: 16 <i>B</i>: Number of <i>pdlogadfg -d sys</i> operands specified[#] <i>C</i>: 0 <i>D</i>: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit Unit subject to standby-less system switchover (effects distributed): <i>pd_ha_max_act_guest_servers</i> operand correction value [#]: Of the values specified for all servers in the unit, specify the maximum value. If the unit is subject to standby-less system switchover (1:1), specify the maximum value of all the values specified for all servers and the alternate BESs in the unit.</p>

Process item	Shared memory calculation formula (bytes)
	<p>If the unit is subject to standby-less system switchover (effects distributed), specify the maximum value of all the values specified for all servers in the unit and all BESs in the HA group.</p>
Synchronization point dump server	$\{$ $\uparrow (368 + 1,456 \times 2) \div 1,024 \uparrow \times 1,024$ $+ \uparrow \{(96 + 80 + 208 + 208) + 192 \times (\text{number of } \text{pdlogadfg} \text{ -d spd operands specified}^\#)$ $+ 416 \times (\text{number of } \text{pdlogadpf} \text{ -d spd operands specified}^\#) + 1,023\} \div 1,024 \uparrow \times 1,024$ $\} \times (\text{total number of servers} + A)$ <p>A: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit</p> <p>Unit subject to standby-less system switchover (effects distributed):</p> <p>pd_ha_max_act_guest_servers operand correction value</p> <p>#: Of the values specified for all servers in the unit, specify the maximum value. If the unit is subject to standby-less system switchover (1:1), specify the maximum value of all the values specified for all servers and the alternate BESs in the unit.</p> <p>If the unit is subject to standby-less system switchover (effects distributed), specify the maximum value of all the values specified for all servers in the unit and all BESs in the HA group.</p>
Common to all units	$a + \{b + 64 + (s + 3) \times c + 64 + 48 + d + e\}$ $\times (\text{total number of FESs, BESs, and DSs in unit} + i)$ $+ (g \times (\text{total number of BESs and DSs in unit} + i)) + f$ $+ (\text{value of } \text{pd_max_server_process} \times 2 + 100) \times (48 + 16) + 32$ $+ (\text{value of } \text{pd_max_server_process} \times 2 + 100 + 384) \times 32 + 32 + h + j + w$ $+ (\text{value of } \text{pd_max_server_process} + 127 + y) \times 32 + 32$ <p>If you are using the standby-less system switchover (effects distributed) facility, add:</p> $(\uparrow (28 + (\uparrow (56 + 72,584) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128)$ $a: 26,640 + v \times 4 \times 34$ $b: 2,988$ $c: 1,956$ $d: 32 \times 32$ $e: 64 + 64 \times \{(s + 3) \times 2$ $+ \text{MAX}(5, \downarrow [s + 3] \div 10 \downarrow) + 7\}$ $f: 512 \times (13 + \text{total number of FESs, BESs, and DSs in unit} \times 3) \times 2$ $g: \{(96 + \text{value of } \text{pd_lck_until_disconnect_cnt} \times 112 + 4,095) \div 4,096\}$ $\times 4,096 \times 2$ $h: \uparrow (\text{number of port numbers specified with } \text{pd_registered_port} \times 16 + 32 + 1,023)$ $\div 1,024 \uparrow \times 1,024$ <p>If pd_registered_port is omitted: 0</p> <p>i: Unit subject to standby-less system switchover (1:1): Number of alternate BESs</p> <p>Unit subject to standby-less system switchover (effects distributed): pd_ha_max_act_guest_servers operand correction value</p> $j: p \times (\text{number of FESs in unit}) + q \times (\text{number of BESs in unit} + i) + r \times (\text{number of DSs in unit})$ $k: \text{value of } \text{pd_fes_lck_pool_partition} \text{ in FES}$ $m: \text{value of } \text{pd_lck_pool_partition} \text{ in BES}$ $n: \text{value of } \text{pd_lck_pool_partition} \text{ in DS}$ $o: (s + 3) \times 2 + \text{MAX}\{5, \downarrow (s + 3) \div 10 \downarrow\} + 7$ $p: \text{If } k \text{ is 2 or more: } 32 + (8 + 8 \times k) \times o$ <p>Otherwise: 0</p> $q: \text{If } m \text{ is 2 or more: } 32 + (8 + 8 \times m) \times o$ <p>Otherwise: 0</p> $r: \text{If } n \text{ is 2 or more: } 32 + (8 + 8 \times n) \times o$

Process item	Shared memory calculation formula (bytes)
	<p>Otherwise: 0</p> <p>s: value of pd_max_users + value of $pd_max_reflect_process_count$</p> <p>$t$: value of $pd_max_dic_process$ + value of $pd_max_reflect_process_count$</p> <p>$u$: value of $pd_max_bes_process$ + value of $pd_max_reflect_process_count$</p> <p>$s$ is t for a DS and u for a BES. If $pd_max_dic_process$ or $pd_max_bes_process$ is omitted, use s.</p> <p>v: If \forall is specified for pd_dbbuff_modify and the unit has a BES or DS: $512 + (\text{maximum value in unit of the value of } pd_max_add_dbbuff_shm_no \text{ in the BES or DS definition})$</p> <p>If N is specified for pd_dbbuff_modify and the unit has a BES or DS: 512</p> <p>Otherwise: 16</p> <p>w: 144</p> <p>y: Add the value calculated by the following formula once for each server.</p> <p>If the unit has a FES, add:</p> $(z + 3) \times 2$ <p>If the unit has a BES^{#1}, with $pd_utl_exec_mode$ set to 1 and $s > 32$, add:</p> $(z + 3) \times 34 + s$ <p>If the unit has a BES^{#1}, with $pd_utl_exec_mode$ set to 0 or $s \leq 32$, add:</p> $(z + 3) \times 34 + 32$ <p>If the unit has a DS, with $pd_utl_exec_mode$ set to 1 and $s > 16$, add:</p> $(z + 3) \times 34 + s$ <p>If the unit has a DS, with $pd_utl_exec_mode$ set to 0 or $s \leq 16$, add:</p> $(z + 3) \times 34 + 16$ <p>z: For a FES if the number of FESs in the HiRDB system > 1: $s + 1$</p> <p>For a FES if the number of FESs in the HiRDB system = 1: s</p> <p>For a BES with $s > \text{value of } pd_max_bes_process^{#2}$: s</p> <p>For a BES with $s \leq \text{value of } pd_max_bes_process^{#2}$: u</p> <p>For a DS with $s > \text{value of } pd_max_dic_process$: s</p> <p>For a DS with $s \leq \text{value of } pd_max_dic_process$: t</p> <p>#1</p> <p>In a unit subject to standby-less system switchover (1:1), servers include all normal BESs and all alternate BESs.</p> <p>In a unit subject to standby-less system switchover (effects distributed) facility, servers include all host BESs and all guest BESs.</p> <p>#2</p> <p>To make the calculation for guest BESs in the unit subject to standby-less system switchover (effects distributed), use the maximum value among those specified for all guest BESs in that unit, not the operand values for a particular guest BES.</p>
Transaction log server	$\{1,024 + 512 \times A\} \times (\text{number of servers in unit} + H)$ $+ \{$ $128 \times B + 128$ $+ [F + \uparrow(128 + 256 + 8 + 224) \div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng^{\#}$ $+ \uparrow(\text{value of } pd_log_max_data_size^{\#} + 68 + 44 + 96 + 160)$ $\div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng^{\#}]$ $\times D + E + (48 + 8) \times B \times 2$ $\} \times (\text{number of BESs and DSs in unit} + H)$ $+ \{$ $584 \times B + 128 \times B + 64 \times B \times C + 128 + F$ $+ \uparrow(128 + 256 + 8 + 224) \div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng^{\#}$ $+ \uparrow(\text{value of } pd_log_max_data_size^{\#} + 68 + 44 + 96 + 160)$ $\}$

Process item	Shared memory calculation formula (bytes)
	$\div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng$ $+ E + (48 + 8) \times (B \times 2 + 2)$ $\} \times (\text{number of servers in unit} + H)$ <p>A: 2</p> <p>B: $7 + J \times 2$</p> <p>C: $\text{number of BESs in the entire system} \times 4 + \text{number of DSs in the entire system} \times 2$ $+ \text{number of FESs in the entire system}$</p> <p>D: If the value of <code>pd_log_rollback_buff_count</code> is 0: 8 Otherwise: <i>value of <code>pd_log_rollback_buff_count</code></i></p> <p>E: 0</p> <p>F: 60</p> <p>H: In a unit subject to standby-less system switchover (1:1): number of alternate BESs in unit When the unit is subject to standby-less system switchover (effects distributed): corrected value of <code>pd_ha_max_act_guest_servers</code> operand</p> <p>J: Maximum value of <i>s</i>, <i>t</i>, and <i>u</i> on the servers in the unit</p> <p><i>s</i>: <i>value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code></i></p> <p><i>t</i>: <i>value of <code>pd_max_dic_process</code> + value of <code>pd_max_reflect_process_count</code></i></p> <p><i>u</i>: <i>value of <code>pd_max_bes_process</code> + value of <code>pd_max_reflect_process_count</code></i></p> <p># Of the values specified for all servers in the unit, specify the maximum value. If the unit is subject to standby-less system switchover (1:1), specify the maximum value of all the values specified for all servers and the alternate BESs in the unit. If the unit is subject to standby-less system switchover (effects distributed), specify the maximum value of all the values specified for all servers in the unit and all BESs in the HA group.</p>
Status server	$\uparrow 64 \div 32 \uparrow \times 32 \times (\text{number of servers in unit} + A)$ <p>A: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit Unit subject to standby-less system switchover (effects distributed): <code>pd_ha_max_act_guest_servers</code> operand correction value</p>
Audit trail management server	$\uparrow A \div 1,024 \uparrow \times 1,024$ <p>A: 640 if the <code>pd_aud_file_name</code> operand is omitted $640 + (304 \times 200) + B + C$ if the <code>pd_aud_file_name</code> operand is specified</p> <p>B: 0 if the <code>pd_aud_async_buff_size</code> operand value is 0 The following value if the <code>pd_aud_async_buff_size</code> operand value is 4,096 or greater: For Linux: $(160 \times \text{value of } pd_aud_async_buff_count \text{ operand})$ $+ \{(\uparrow \text{value of } pd_aud_async_buff_size \text{ operand} \div 4,096 \uparrow \times 4,096)$ $\times \text{value of } pd_aud_async_buff_count \text{ operand}\} + 4,096$ For a system other than Linux: $(160 \times \text{value of } pd_aud_async_buff_count \text{ operand})$ $+ \{(\uparrow \text{value of } pd_aud_async_buff_size \text{ operand} \div 4,096 \uparrow \times 4,096)$ $\times \text{value of } pd_aud_async_buff_count \text{ operand}\}$</p> <p>C: If <i>Y</i> is specified for the <code>pd_aud_auto_loading</code> operand and the unit has a MGR: $256 \times (\text{total number of units in system} + 1) + 240$ Otherwise: 0</p> <p>If the unit uses the standby-less system switchover facility, the size of memory required for security audit at the target unit must be added to the size of memory for the local unit.</p>

(2) 64-bit mode

The size of memory required for the unit controller in each server machine from startup to termination of the unit is the sum of the following items:

Process item	Shared memory calculation formula (bytes)
Scheduler	<p>Value of <code>pd_utl_exec_mode</code> set to 0:</p> $\{ \uparrow (432 + 304 \times n) \div 1,024 \uparrow + 494 + x + z + \uparrow (134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \uparrow \} \times 1,024$ <p>Value of <code>pd_utl_exec_mode</code> set to 1:</p> $\{ \uparrow (432 + 304 \times n) \div 1,024 \uparrow + \uparrow (s \times 2,000 + 136) \div 1,024 \uparrow + y + z + \uparrow (134 + \text{value of } pd_trn_rcvmsg_store_buflen) \div 1,024 \uparrow \} \times 1,024$ <p><i>x</i>: Unit contains MGR: 37 Unit contains FES: $57 + 1 \times (s + 3) + 14$ Unit contains DS: $102 + 5 \times (t + 3) + 14$ Unit contains BES: $\{192 + 9 \times (u + 3) + 14\} \times (\text{number of BESs} + \beta + \gamma)$</p> <p><i>y</i>: Unit contains MGR: 0 Unit contains FES: $1 \times (s + 3) + 14$ Unit contains DS: $5 \times (t + 3) + 14$ Unit contains BES: $\{9 \times (u + 3) + 14\} \times (\text{number of BESs} + \beta + \gamma)$</p> <p><i>z</i>: Unit subject to standby-less system switchover (effects distributed): $\uparrow 64 + \{(\text{Number of BESs in unit} + \text{number of acceptable guest BESs}^\#) \times 48\} \div 1,024 \uparrow$ Other unit: 0</p> <p><i>n</i>: Number of servers in unit + $\beta + \gamma$ + number of utility servers in unit + 1 Number of utility servers in unit: $24 + \alpha$ α: If the unit has a MGR: 3 If the unit has a FES: 3 If the unit has a DS: 7 If the unit has a BES: $(\text{number of BESs} + \beta) \times 15$</p> <p><i>s</i>: value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code> <i>t</i>: value of <code>pd_max_dic_process</code> + value of <code>pd_max_reflect_process_count</code> <i>u</i>: value of <code>pd_max_bes_process</code> + value of <code>pd_max_reflect_process_count</code></p> <p>β: For units subject to standby-less system switchover (effects distributed): Number of guest BESs that can be accepted[#] For other units: 0</p> <p>γ: For units subject to standby-less system switchover (1:1): Number of alternate BESs For other units: 0</p> <p>[#]: Value of <code>pd_ha_max_guest_act_servers</code></p>
Lock server	<p>If a server (FES, BES, or DS) exists in the unit:</p> <p>For the values of the operands that are used for each server to determine the value for guest BESs in the unit subject to standby-less system switchover (effects distributed) (such as <code>pd_lck_pool_size</code>, <code>pd_lck_pool_partition</code>, <code>pd_lck_hash_entry</code>, and <code>pd_max_bes_process</code>), use the maximum value among those specified for all guest BESs in that unit, not the operand values for a particular guest BES. In a unit that is subject to standby-less system switchover (effects distributed), <i>servers in the unit</i> means <i>all host BESs + all guest BESs</i>.</p> <p>In a unit subject to standby-less system switchover (1:1), <i>servers in the unit</i> means <i>all normal BESs + all alternate BESs</i>.</p> <p><i>y</i></p> $\sum_{x=1} \{ 496 + 80 + c_x + d_x + 64 + 8,192 + g_x + 80 + i_x$

Process item	Shared memory calculation formula (bytes)
	$+ 64 + 16,336 + 64 + n_x + p_x + t_x + u_x + 16$ $\} \times \text{value of } pd_lck_pool_partition^\#$ $\#$: For a FES, the value of $pd_fes_lck_pool_partition$ x : Server serial number in the unit y : Number of servers in the unit c_x : If $pd_lck_hash_entry$ is omitted or if 0 is specified: For a FES with $pd_fes_lck_pool_size$ omitted: $(\downarrow(8+8 \times \text{MAX}(\text{largest prime number} < \uparrow(\downarrow(p_x+3) \times (\text{value of } pd_max_access_tables + 4) \div 4 \downarrow$ $\div \text{value of } pd_fes_lck_pool_partition \downarrow \times 4) \div 10 \uparrow, 11,261))$ $\div 16 \downarrow + 1) \times 16$ For a FES with $pd_fes_lck_pool_size$ specified: $(\downarrow(8+8 \times \text{MAX}(\text{largest prime number} < \uparrow(\downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 4) \div 10 \uparrow$ $, 11,261)) \div 16 \downarrow + 1) \times 16$ For a BES or DS: $(\downarrow(8+8 \times \text{MAX}(\text{largest prime number} < \uparrow((p_x+3) \times 2 \times 5$ $+ \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 4) \div 10 \uparrow$ $, 11,261)) \div 16 \downarrow + 1) \times 16$ If 2 or a larger non-prime number is specified for $pd_lck_hash_entry$: $(\downarrow(8+8 \times \text{largest prime number that is less than the value of } pd_lck_hash_entry) \div 16 \downarrow + 1)$ $\times 16$ If 1 or a prime number is specified for $pd_lck_hash_entry$: $(\downarrow(8+8 \times \text{value of } pd_lck_hash_entry) \div 16 \downarrow + 1) \times 16$ d_x : For a FES with $pd_fes_lck_pool_size$ omitted: $(\downarrow(p_x+3) \times (\text{value of } pd_max_access_tables + 4) \div 4 \downarrow$ $\div \text{value of } pd_fes_lck_pool_partition \downarrow \times 4) \times 128$ For a FES with $pd_fes_lck_pool_size$ specified: $\downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 4 \times 128$ For a BES or DS: $((p_x+3) \times 2 \times 5 + \downarrow \text{value of } pd_lck_pool_size$ $\div \text{value of } pd_lck_pool_partition \downarrow \times 4) \times 128$ g_x : For a FES: $(p+3) \times 2 \times 320$ For a BES with the value of $pd_utl_exec_mode$ set to 1 and $s > 32$: $((p+3) \times 2 + s) \times 320$ For a BES with the value of $pd_utl_exec_mode$ set to 0 or $s \leq 32$: $((p+3) \times 2 + 32) \times 320$ For a DS with the value of $pd_utl_exec_mode$ set to 1 and $s > 16$: $((p+3) \times 2 + s) \times 320$ For a DS with the value of $pd_utl_exec_mode$ set to 0 or $s \leq 16$: $((p+3) \times 2 + 16) \times 320$ i_x : For a FES with $pd_fes_lck_pool_size$ omitted: $(\downarrow((p_x+3) \times (\text{value of } pd_max_access_tables + 4))$ $\div \text{value of } pd_fes_lck_pool_partition \downarrow) \times 8 \times 112$

Process item	Shared memory calculation formula (bytes)
	<p>For a FES with <code>pd_fes_lck_pool_size</code> specified:</p> $(\downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 5 \\ + \downarrow \downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \div 3 \downarrow) \text{ rounded up to an even number } \times 112$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5 \\ + \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow \\ + (p_x + 3) \times 2 \times 2 \times 5 + s \times (\text{value of } pd_max_rdarea_no + 1)) \\ \text{rounded up to an even number } \times 112$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5 \\ + \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow \\ + (p_x + 3) \times 2 \times 2 \times 5 + 32 \times (\text{value of } pd_max_rdarea_no + 1)) \\ \text{rounded up to an even number } \times 112$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5 \\ + \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow \\ + (p_x + 3) \times 2 \times 2 \times 5 + s + 4) \text{ rounded up to an even number } \times 112$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $(\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5 \\ + \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow \\ + (p_x + 3) \times 2 \times 2 \times 5 + 20) \\ \text{rounded up to an even number } \times 112$ <p>n_x: For a FES:</p> $(p_x + 3) \times 2 \times 80$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $((p + 3) \times 2 \times 17 + s) \times 80$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $((p + 3) \times 2 \times 17 + 32) \times 80$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $((p + 3) \times 2 \times 17 + s) \times 80$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $((p + 3) \times 2 \times 17 + 16) \times 80$ <p>p_x: For a FES with the number of FESs in the HiRDB system > 1: $s + 1$</p> <p>For a FES with the number of FESs in the HiRDB system = 1: s</p> <p>For a BES with $s > \text{value of } pd_max_bes_process$: s</p> <p>For a BES with $s \leq \text{value of } pd_max_bes_process$:</p> $\text{value of } pd_max_bes_process \text{ operand} \\ + \text{value of } pd_max_reflect_process_count \text{ operand}$ <p>For a DS with $s > \text{value of } pd_max_dic_process$: s</p> <p>For a DS with $s \leq \text{value of } pd_max_dic_process$:</p> $\text{value of } pd_max_dic_process \text{ operand} \\ + \text{value of } pd_max_reflect_process_count \text{ operand}$ <p>s: $\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count$</p>

Process item	Shared memory calculation formula (bytes)
	<p>t_x: For a FES:</p> $48 + (p_x + 3) \times 2 \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $48 + ((p_x + 3) \times 2 + s) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $48 + ((p_x + 3) \times 2 + 32) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $48 + ((p_x + 3) \times 2 + s) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $48 + ((p_x + 3) \times 2 + 16) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>u_x: For a FES with <code>pd_fes_lck_pool_size</code> omitted:</p> $48 + (\downarrow \downarrow (p_x + 3) \times (\text{value of } pd_max_access_tables + 4) \div 4 \downarrow \div \text{value of } pd_fes_lck_pool_partition \downarrow) \times 8$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a FES with <code>pd_fes_lck_pool_size</code> specified:</p> $48 + (\downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \times 5$ $+ \downarrow \downarrow \text{value of } pd_fes_lck_pool_size \div \text{value of } pd_fes_lck_pool_partition \downarrow \div 3 \downarrow)$ <p>rounded up to an even number $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 32$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5$ $+ \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow$ $+ (p_x + 3) \times 2 \times 2 \times 5 + s \times (\text{value of } pd_max_rdarea_no + 1))$ <p>rounded up to an even number $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <p>For a BES with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5$ $+ \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow$ $+ (p_x + 3) \times 2 \times 2 \times 5 + 32 \times (\text{value of } pd_max_rdarea_no + 1))$ <p>rounded up to an even number $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 1 and $s > 16$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5$ $+ \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow$ $+ (p_x + 3) \times 2 \times 2 \times 5 + s + 4) \text{ rounded up to an even number}$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For a DS with the value of <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$:</p> $48 + (\downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \times 5$ $+ \downarrow \downarrow \text{value of } pd_lck_pool_size \div \text{value of } pd_lck_pool_partition \downarrow \div 3 \downarrow$ $+ (p_x + 3) \times 2 \times 2 \times 5 + 20) \text{ rounded up to an even number}$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>■ If no server (FES, BES, or DS) exists in the unit</p> <p>16,704</p>
Transaction server	$304 + 32 \times B + 192 \times s \times 2$ <p>If the unit contains FES, add the following value: #</p> $+ 1,048 + (416 + 720 + 256 + 392 \times 2) \times (A \times 2 + 7) + 256 \times 2$

Process item	Shared memory calculation formula (bytes)
	$+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2 + \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ C$ If the unit contains BES, add the following value: [#] $+ 1,048 + (416 + 720 + 256 + 392 \times 2) \times (u \times 2 + 7)$ $+ 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2 + \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ D$ If the unit contains DS, add the following value: [#] $+ 1,048 + (416 + 720 + 256 + 392 \times 2) \times (t \times 2 + 7)$ $+ 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2 + \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ E$ s : Value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code> t : Value of <code>pd_max_dic_process</code> + value of <code>pd_max_reflect_process_count</code> u : Value of <code>pd_max_bes_process</code> + value of <code>pd_max_reflect_process_count</code> A : For a multi-FES system: $s + 1$; if not a multi-FES system: s B : Unit subject to standby-less system switchover (effects distributed): Number of host BESs + value of <code>pd_ha_max_act_guest_servers</code> operand Other unit: Number of servers in the unit C : If the unit meets one condition in <i>Conditions</i> below: 0 $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2 + \text{number of FESs in the system}) \times (A \times 2 + 7)$ If the unit does not meet both conditions in <i>Conditions</i> below: 0 D : If the unit meets one condition in <i>Conditions</i> below: $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2 + \text{number of FESs in the system}) \times (u \times 2 + 7)$ If the unit does not meet both conditions in <i>Conditions</i> below: 0 E : If the unit meets one condition in <i>Conditions</i> below: $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2 + \text{number of FESs in the system}) \times (t \times 2 + 7)$ If the unit does not meet both conditions in <i>Conditions</i> below: 0 Conditions: <ul style="list-style-type: none"> • <code>uap</code> is specified in the <code>pd_rpl_reflect_mode</code> operand. • The <code>pdstart -k stls</code> operand is specified for a front-end server in the system. [#] : Add the value calculated by the above formula once for each server. For units subject to standby-less system switchover (1:1): $\text{Number of normal BESs} + \text{Number of alternate BESs}$ Other than the above, the number of the servers in the unit
Timer server	$32 \times (\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count + 3)$ $\times (\text{number of BESs in system} + 1 + \text{number of utility servers in unit} + 1)$ $+ 1,440 + (48 - 32) \times 2$ Number of utility servers in unit is $23 + \alpha$ α : When there is MGR in the unit: 2 When there is FES in the unit: 3

Process item	Shared memory calculation formula (bytes)
	<p>When there is DS in the unit: 7</p> <p>When there is BES in the unit: number of BESs $\times b$</p> <p>b: For HP-UX, Solaris, and AIX: 15</p> <p>For Linux: 6</p>
Statistics log server	$424 + 128 \times 16 + 32 + 288 \times 2 + 1,168 + 144 \times 3$ $+ \text{value of } pd_stj_buff_size \times 1,024 \times 3 + 64 + 4,096 + 8,192$
Process server	$208 + 528 \times \text{MAX}(c, 256) + 80 + 256 + (\text{value of } pd_max_server_process + 50) \times (256 + 160) + 16 + 8$ $\times 34 + 16 + 16 + 64 + 64 \times (k + 1)$ $c: \uparrow(51 + d + e + f + (g \times \text{number of BESs in the unit}^\#) + h + i) \div 16 \uparrow \times 16$ d : If the unit contains MGR, 59; if not, j . e : If the unit contains DS, 17; if not, 0. f : If the unit contains FES, 11; if not, 0. g : If the unit contains BES, 25; if not, 0. h : If the standby-less system switchover (1:1) facility is used, 9; if not, 0. i : If the unit is subject to standby-less system switchover (effects distributed), 1; if not, 0. j : If <code>manager</code> is specified in the <code>pd_mlg_msg_log_unit</code> operand, 1; if <code>local</code> is specified, 2. k : If the operand <code>pd_module_trace_max</code> is specified in the system common definition or the unit control information definition, the value of <code>pd_module_trace_max</code> ; if not, 16,383. m : If the unit has a MGR, 0; otherwise, 1. $\#$: If the unit is subject to standby-less system switchover (1:1), the value is (number of BESs $\times 2$). If the unit is subject to standby-less system switchover (effects distributed), the value includes the <code>pd_ha_max_act_guest_servers</code> operand correction value.
System manager	$736 + (48 + 8) \times (g + h + i) + (108 + 8) \times \{(p + q + 3) + u \times (15 + 1)\} + (104 + 8) \times c + 40 \times (k + m$ $+ n \times o + u) \times 14 + 256 \times m + 128 \times c + 40 \times d + 16 \times e + 96 \times o + v \times (16 \times 35 \times (k + u)$ $+ 15 + 44 \times z + 15) + w \times (48 \times B + 15 + 4 \times z + 15 + 4 \times y + 15) + v \times (144 + 15) + 8 + 5,864 + s$ $+ s \times o + 16 + 96 \times o + 1,024 + 272 \times A$ c : Number of units d : Number of <code>-c</code> options specified in <code>pdunit</code> operand e : Number of <code>pdcltgrp</code> operands specified g : Number of FESs in the system h : Number of BESs in the system i : Number of DSs in the system j : Number of FESs in unit k : Number of BESs in unit m : Number of DSs in the unit n : Number of alternate BESs in the unit o : If the unit is subject to standby-less system switchover, 1; if not, 0 $p: i + k + m + n$ $q: 24 + t + j \times 3 + k \times 15 + m \times 7$ $r: 14 \times (k + m + u) + p + q + u \times 15 + 2 + 38 + 5 + p \times 4$ $s: 256 + 2,052 + 148 \times (r + 3) + v \times (40 \times (k + u) + 72 \times (k + u))$ t : If the unit contains MGR, 2; if not, 0 u : Number of acceptable BESs (value of the <code>pd_ha_max_act_guest_servers</code> operand) v : If the unit is subject to standby-less system switchover (effects distributed), 1; if not, 0 w : If there is a unit that is subject to standby-less system switchover (effects distributed), 1; if not, 0 y : Sum of the units in the HA group z : Sum of the servers in the HA group A : Number of IP addresses for the host specified in the <code>pd_security_host_group</code> operand

Process item	Shared memory calculation formula (bytes)
	<p>If the <code>pd_security_host_group</code> operand is not specified, 0</p> <p>B: Number of <code>pdhagroup</code> operands specified in the system common definition</p>
Name server	<p>$\text{MAX}\{65,536, (X + Y + Z)\} + \text{MAX}(16,384, L) + M$</p> <p>$X$: $\uparrow (272 + 16 + 156 \times \text{number of units} + 16 + 16 \times 126) \div 1,024 \uparrow \times 1,024$</p> <p>$Y$: 8,192</p> <p>$Z$: $\uparrow (264 \times (Z_2 + Z_3 + j + 32)) \div 1,024 \uparrow \times 1,024$</p> <p>$Z_2$: $b + 10 + c + 11 \times \text{number of HiRDB servers within local unit} + d + e$</p> <p>$Z_3$: $f + 7 + g + 4 \times \text{number of HiRDB servers within local unit} + h + i$</p> <p>$L$: $\uparrow (224 \times (L_2 + L_3 + L_4)) \div 1,024 \uparrow \times 1,024$</p> <p>$L_2$: $k + 2 \times \text{number of units within system}$</p> <p>$L_3$: Number of BESs within system + number of FESs within system + number of DSs within system</p> <p>L_4: $15 \times \text{number of HiRDB servers within system}$</p> <p>$M$: Number of HiRDB servers within unit + $z + m \times \text{number of system servers within unit} \times 1,024 \div 1,024$</p> <p>$b$: If the unit has a MGR: 3</p> <p>If the unit does not have a MGR: 0</p> <p>c: For units subject to standby-less system switchover (1:1): 2</p> <p>For other units: 0</p> <p>d: For units subject to standby-less system switchover (effects distributed):</p> <p>$11 \times \text{number of guest BESs that can be accepted}$</p> <p>For other units: 0</p> <p>e: For units subject to standby-less system switchover (1:1):</p> <p>$6 \times \text{number of HiRDB servers within local unit}$</p> <p>For other units: 0</p> <p>f: If the unit has a MGR: 3</p> <p>If the unit does not have a MGR: 0</p> <p>g: For units subject to standby-less system switchover (1:1): 2</p> <p>For other units: 0</p> <p>h: For units subject to standby-less system switchover (effects distributed):</p> <p>$3 \times \text{number of guest BESs that can be accepted}$</p> <p>For other units: 0</p> <p>i: For units subject to standby-less system switchover (1:1):</p> <p>$4 \times \text{number of HiRDB servers within local unit}$</p> <p>For other units: 0</p> <p>j: Number of servers in unit + $\beta + \gamma + \text{number of utility servers in unit} + 2$</p> <p>Number of utility servers in unit: $23 + \alpha$</p> <p>k: If the unit has a MGR: 3</p> <p>If the unit does not have a MGR: 0</p> <p>m: $38 + n + o$</p> <p>n: If the unit has a MGR and the number of units is 2 or more: 5</p> <p>If the unit has a MGR and the number of units is 1: 4</p> <p>If the unit does not have a MGR and the value of <code>pd_mlg_msg_log_unit</code> is <code>local</code>: 4</p> <p>If the unit does not have a MGR and the value of <code>pd_mlg_msg_log_unit</code> is <code>manager</code>: 3</p> <p>o: $4 \times \text{number of HiRDB servers within unit}$</p> <p>$z$: $23 + \alpha$</p> <p>α: If the unit has a MGR: 3</p>

Process item	Shared memory calculation formula (bytes)
	<p>If the unit has a FES: 3</p> <p>If the unit has a DS: 7</p> <p>If the unit has a BES: $(\text{number of BESs} + \beta) \times 15$</p> <p>$\beta$: For units subject to standby-less system switchover (effects distributed):</p> <p>Number of guest BESs that can be accepted[#]</p> <p>For other units: 0</p> <p>γ: For units subject to standby-less system switchover (1:1): number of alternate BESs</p> <p>For other units: 0</p> <p>#: Value of <code>pd_ha_max_guest_act_servers</code></p>
Node manager	<p>Unit contains MGR:</p> $\uparrow (1,312 + 464 \times \text{total number of units in system} + 96 \times \text{total number of servers in system} + 10,240 + 1,200 + 72 \times C + 240 \times A + 44 \times A + 28 \times A + 16 \times B + 16 \times \text{total number of BESs in system} + 8 \times \text{total number of units in system} + 64 \times \text{total number of servers in system} + 32 \times \text{total number of servers in system} + 48 + J + 16 \times K) \div 1,024 \uparrow \times 1,024$ <p>Unit contains no MGR:</p> $\uparrow (1,200 + 72 \times C + (240 \times A + 44 \times A + 28 \times A) \times F + 16 \times B + 16 \times \text{total number of BESs in system} + 8 \times \text{total number of units in system} + 64 \times \text{total number of servers in system} + 32 \times \text{total number of servers in system} + 48) \div 1,024 \uparrow \times 1,024$ <p>A: Value of <code>pd_utl_exec_mode</code> = 0: 1,024</p> <p>If <code>pd_utl_exec_mode</code> = 1: (value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>) \times total number of BESs in system \times 3</p> <p>If the unit has a MGR, add: (value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>) \times 4 + 200</p> <p>If unit has a DS, add: (value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>) \times 3 + 200</p> <p>If unit has a BES, add: (value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>) \times D</p> <p>If the value of A calculated in the above formulas does not exceed 1,024, substitute 1,024 for A.</p> <p>B: <code>pdcltgrp</code> operand not specified: 0</p> <p><code>pdcltgrp</code> operand specified: Number of <code>pdcltgrp</code> operands specified + 1</p> <p>C: Number of servers in the unit + E</p> <p>D: Number of BESs in unit + E</p> <p>E: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit</p> <p>Unit subject to standby-less system switchover (effects distributed): Number of acceptable BESs</p> <p>Other unit: 0</p> <p>F: For units subject to standby-less system switchover (1:1): 2</p> <p>For other units: 1</p> <p>J: If the <code>pd_system_expand_unit</code> operand is specified: 16</p> <p>If the <code>pd_system_expand_unit</code> operand is omitted: 0</p> <p>K: If the <code>pd_system_expand_unit</code> operand is specified: The number of units specified in the <code>pd_system_expand_unit</code></p> <p>If the <code>pd_system_expand_unit</code> operand is omitted: 0</p>
I/O server	$\uparrow (56 + (\uparrow (56 + A) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128$ <p>If the unit is not subject to standby-less system switchover (effects distributed):</p>

Process item	Shared memory calculation formula (bytes)
	<p> $A: 3,248 + (14 + 16) \times 972 + 1 \times 276 + (534 \times 276)^{\#1}$ $+ \{(534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972) \times \text{number of BESs}\}^{\#2}$ $+ \{534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972\}^{\#3}$ $+ (48 - 32) \times 3$ #1 Add this value if FESs exist. #2 Add this value if BESs exist. #3 Add this value if DSs exist. </p> <p>In a unit subject to standby-less system switchover (1:1), double the value obtained in the above formula.</p> <p>If the unit is subject to standby-less system switchover (effects distributed):</p> <p> $A: \uparrow 64 + 24 \times \text{number of BESs}^{\#4} \div 16 \uparrow \times 16$ $+ \uparrow (3,296 + 16 \times 972 + 534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972) \div 16 \uparrow \times$ $16 \times \text{number of BESs}^{\#4}$ $+ \uparrow (3,296 + (14 + 16) \times 972 + 1 \times 276) \div 16 \uparrow \times 16$ </p> <p>#4 Includes the value of <code>pd_ha_max_act_guest_servers</code>.</p>
Log server	<p> $32 + 48 + 128 \times 37$ $+ \{$ $432 + 128 \times 7 + 1,168 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of } pd_log_rec_leng^{\#} \uparrow$ $\times \text{value of } pd_log_rec_leng^{\#}$ $+ 64 + 4,096 \times 2 + (768 + 512) \times B$ $+ \uparrow \{(B + 1) \div 12\} \uparrow \times 8,320$ $+ 144 \times \text{value of } pd_log_write_buff_count^{\#}$ $+ (\text{value of } pd_log_write_buff_count^{\#} + A)$ $\times \uparrow \{\text{value of } pd_log_max_data_size^{\#} + (68 + 44 + 96 + 160)\} \div 4,096 \uparrow \times 4,096$ $+ C$ $\} \times \text{number of servers in the unit} + D + 128 \times \text{number of FESs in the unit}$ Add this if the <code>pd_max_reflect_process_count</code> operand is specified. $(128 + 704) \times (\text{number of BESs in the unit} + D)$ A: 16 B: Number of <code>pdlogadfg -d sys</code> operands specified[#] C: 0 D: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit Unit subject to standby-less system switchover (effects distributed): <code>pd_ha_max_act_guest_servers</code> operand correction value #: Of the values specified for all servers in the unit, specify the maximum value. If the unit is subject to standby-less system switchover (1:1), specify the maximum value of all the values specified for all servers and the alternate BESs in the unit. If the unit is subject to standby-less system switchover (effects distributed), specify the maximum value of all the values specified for all servers in the unit and all BESs in the HA group. </p>
Synchronization point dump server	<p> $\{$ $\uparrow (384 + 1,536 \times 2) \div 1,024 \uparrow \times 1,024$ $+ \uparrow \{(128 + 80 + 240 + 240) + 192 \times (\text{number of } pdlogadfg -d spd \text{ operands specified}^{\#})$ $+ 416 \times (\text{number of } pdlogadpf -d spd \text{ operands specified}^{\#}) + 1,023\} \div 1,024 \uparrow \times 1,024$ $\} \times (\text{total number of servers} + A)$ A: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit Unit subject to standby-less system switchover (effects distributed): <code>pd_ha_max_act_guest_servers</code> operand correction value </p>

Process item	Shared memory calculation formula (bytes)
	<p>#: Of the values specified for all servers in the unit, specify the maximum value. If the unit is subject to standby-less system switchover (1:1), specify the maximum value of all the values specified for all servers and the alternate BESs in the unit.</p> <p>If the unit is subject to standby-less system switchover (effects distributed), specify the maximum value of all the values specified for all servers in the unit and all BESs in the HA group.</p>
Common to all units	$a + \{b + 80 + (s + 3) \times c + 64 + 48 + d + e\}$ $\times (\text{total number of FESs, BESs, and DSs in unit} + i)$ $+ (g \times (\text{total number of BESs and DSs in unit} + i)) + f$ $+ (\text{value of } pd_max_server_process \times 2 + 100) \times (64 + 16) + 32$ $+ (\text{value of } pd_max_server_process \times 2 + 100 + 384) \times 32 + 32 + h + j + w$ $+ (\text{value of } pd_max_server_process + 127 + y) \times 48 + 32$ <p>If you are using the standby-less system switchover (effects distributed) facility, add:</p> $(\uparrow(56 + (\uparrow(56 + 88,560) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128)$ $a: 36,048 + v \times 4 \times 34$ $b: 3,480$ $c: 2,760$ $d: 48 \times 32$ $e: 80 + 96 \times \{(s + 3) \times 2$ $+ \text{MAX}(5, \downarrow[s + 3] \div 10 \downarrow) + 7\}$ $f: 512 \times (13 + (\text{total number of FESs, BESs, and DSs in unit} + i) \times 3) \times 2$ $g: \{(128 + \text{value of } pd_lck_until_disconnect_cnt \times 112 + 4,095) \div 4,096\}$ $\times 4,096 \times 2$ $h: \uparrow(\text{number of port numbers specified with } pd_registered_port \times 16 + 32 + 1,023)$ $\div 1,024 \uparrow \times 1,024$ <p>If <code>pd_registered_port</code> is omitted: 0</p> <p><i>i</i>: Unit subject to standby-less system switchover (1:1): Number of alternate BESs Unit subject to standby-less system switchover (effects distributed): <code>pd_ha_max_guest_servers</code> operand correction value</p> $j: p \times (\text{number of FESs in unit}) + q \times (\text{number of BESs in unit} + i) + r \times (\text{number of DSs in unit})$ <p><i>k</i>: value of <code>pd_fes_lck_pool_partition</code> in FES</p> <p><i>m</i>: value of <code>pd_lck_pool_partition</code> in BES</p> <p><i>n</i>: value of <code>pd_lck_pool_partition</code> in DS</p> $o: (s + 3) \times 2 + \text{MAX}\{5, \downarrow(s + 3) \div 10 \downarrow\} + 7$ <p><i>p</i>: If <i>k</i> is 2 or more: $32 + (8 + 16 \times k) \times o$ Otherwise: 0</p> <p><i>q</i>: If <i>m</i> is 2 or more: $32 + (8 + 16 \times m) \times o$ Otherwise: 0</p> <p><i>r</i>: If <i>n</i> is 2 or more: $32 + (8 + 16 \times n) \times o$ Otherwise: 0</p> <p><i>s</i>: value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code></p> <p><i>t</i>: value of <code>pd_max_dic_process</code> + value of <code>pd_max_reflect_process_count</code></p> <p><i>u</i>: value of <code>pd_max_bes_process</code> + value of <code>pd_max_reflect_process_count</code></p> <p><i>s</i> is <i>t</i> for a DS and <i>u</i> for a BES. If <code>pd_max_dic_process</code> or <code>pd_max_bes_process</code> is omitted, use <i>s</i>.</p> <p><i>v</i>: If <i>Y</i> is specified for <code>pd_dbbuff_modify</code> and the unit has a BES or DS: $512 + (\text{maximum value in unit of the value of } pd_max_add_dbbuff_shm_no \text{ in the BES or DS definition}) + \text{value of } pd_max_resident_rdarea_shm_no$</p> <p>If <i>N</i> is specified for <code>pd_dbbuff_modify</code> and the unit has a BES or DS: $512 + \text{value of } pd_max_resident_rdarea_shm_no$</p> <p>Otherwise: 16</p>

Process item	Shared memory calculation formula (bytes)
	<p>w: HP-UX (IPF) 64-bit mode: 256 Otherwise: 144</p> <p>y: Add the value calculated using the following formula once for each server. If the unit has a FES, add: $(z + 3) \times 2$ If the unit has a BES^{#1}, with <code>pd_utl_exec_mode</code> set to 1 and $s > 32$, add: $(z + 3) \times 34 + s$ If the unit has a BES^{#1}, with <code>pd_utl_exec_mode</code> set to 0 or $s \leq 32$, add: $(z + 3) \times 34 + 32$ If the unit has a DS, with <code>pd_utl_exec_mode</code> set to 1 and $s > 16$, add: $(z + 3) \times 34 + s$ If the unit has a DS, with <code>pd_utl_exec_mode</code> set to 0 or $s \leq 16$, add: $(z + 3) \times 34 + 16$</p> <p>z: For a FES if the number of FESs in the HiRDB system > 1: $s + 1$ For a FES if the number of FESs in the HiRDB system = 1: s For a BES with $s > \text{value of } pd_max_bes_process^{#2}$: s For a BES with $s \leq \text{value of } pd_max_bes_process^{#2}$: u For a DS with $s > \text{value of } pd_max_dic_process$: s For a DS with $s \leq \text{value of } pd_max_dic_process$: t</p> <p>#1 In a unit subject to standby-less system switchover (1:1), servers include all normal BESs and all alternate BESs. In a unit subject to the standby-less system switchover (effects distributed) facility, servers include all host BESs and all guest BESs.</p> <p>#2 To make the calculation for guest BESs in the unit subject to standby-less system switchover (effects distributed), use the maximum value among those specified for all guest BESs in that unit, not the operand values for a particular guest BES.</p>
Transaction log server	$\{1,168 + 688 \times A\} \times (\text{number of servers in unit} + H)$ $+ \{$ $128 \times B + 144$ $+ [G + \uparrow(128 + 256 + 8 + 224) \div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng^{\#}$ $+ \uparrow(\text{value of } pd_log_max_data_size^{\#} + 68 + 44 + 96 + 160)$ $\div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng^{\#}]$ $\times D + E + (48 + 8) \times B \times 2$ $\} \times (\text{number of BESs and DSs in unit} + H)$ $+ \{$ $600 \times B + 128 \times B + 64 \times B \times C + 144 + G$ $+ \uparrow(128 + 256 + 8 + 224) \div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng^{\#}$ $+ \uparrow(\text{value of } pd_log_max_data_size^{\#} + 68 + 44 + 96 + 160)$ $\div \text{value of } pd_log_rec_leng^{\#} \uparrow \times \text{value of } pd_log_rec_leng^{\#}$ $+ E + (48 + 8) \times (B \times 2 + 2)$ $\} \times (\text{number of servers in unit} + H)$ <p>A: 2 B: $7 + J \times 2$ C: $\text{number of BESs in the entire system} \times 4 + \text{number of DSs in the entire system} \times 2$ $+ \text{number of FESs in the entire system}$</p>

Process item	Shared memory calculation formula (bytes)
	<p><i>D</i>: If the value of <code>pd_log_rollback_buff_count</code> is 0: 8 Otherwise: <i>value of pd_log_rollback_buff_count</i></p> <p><i>E</i>: 0</p> <p><i>G</i>: 64</p> <p><i>H</i>: In a unit subject to standby-less system switchover (1:1): number of alternate BESs in unit When the unit is subject to standby-less system switchover (effects distributed): corrected value of <code>pd_ha_max_act_guest_servers</code> operand</p> <p><i>J</i>: Maximum value of <i>s</i>, <i>t</i>, and <i>u</i> on the servers in the unit</p> <p><i>s</i>: <i>value of pd_max_users</i> + <i>value of pd_max_reflect_process_count</i></p> <p><i>t</i>: <i>value of pd_max_dic_process</i> + <i>value of pd_max_reflect_process_count</i></p> <p><i>u</i>: <i>value of pd_max_bes_process</i> + <i>value of pd_max_reflect_process_count</i></p> <p>#</p> <p>Of the values specified for all servers in the unit, specify the maximum value. If the unit is subject to standby-less system switchover (1:1), specify the maximum value of all the values specified for all servers and the alternate BESs in the unit. If the unit is subject to standby-less system switchover (effects distributed), specify the maximum value of all the values specified for all servers in the unit and all BESs in the HA group.</p>
Status server	<p>$\uparrow 64 \div 32 \uparrow \times 32 \times (\text{number of servers in unit} + A)$</p> <p><i>A</i>: Unit subject to standby-less system switchover (1:1): Number of alternate BESs in the unit Unit subject to standby-less system switchover (effects distributed): <code>pd_ha_max_act_guest_servers</code> operand correction value</p>
Audit trail management server	<p>$\uparrow A \div 1,024 \uparrow \times 1,024$</p> <p><i>A</i>: 704 if the <code>pd_aud_file_name</code> operand is omitted 704 + (320 \times 200) + <i>B</i> + <i>C</i> if the <code>pd_aud_file_name</code> operand is specified</p> <p><i>B</i>: 0 if the <code>pd_aud_async_buff_size</code> operand value is 0</p> <p>The following value if the <code>pd_aud_async_buff_size</code> operand value is 4,096 or greater: For Linux version: (176 \times <i>value of pd_aud_async_buff_count operand</i>) + {(\uparrow <i>value of pd_aud_async_buff_size operand</i> \div 4,096 \uparrow \times 4,096) \times <i>value of pd_aud_async_buff_count operand</i>} + 4,096</p> <p>For non-Linux versions: (176 \times <i>value of pd_aud_async_buff_count operand</i>) + {(\uparrow <i>value of pd_aud_async_buff_size operand</i> \div 4,096 \uparrow \times 4,096) \times <i>value of pd_aud_async_buff_count operand</i>}</p> <p><i>C</i>: If <i>Y</i> is specified for the <code>pd_aud_auto_loading</code> operand and the unit has a MGR: 256 \times (<i>total number of units in system</i> + 1) + 256 Otherwise: 0</p> <p>If the unit uses the standby-less system switchover facility, the size of memory required for security audit at the target unit must be added to the size of memory for the local unit.</p>

15.2.4 Formulas for shared memory used by each server

(1) Formula for the shared memory used by a front-end server

Following is the formula for calculating the size of the shared memory that is used by a front-end server. For the variables used in this formula, see (4) below.

- **32-bit mode**

$$\begin{aligned}
& 40 + b \times 1.3 + c + d + k + 1.6 + x + y + 4 \\
& + \{[(a + 12) \div 13] \times 1.1 + [(a + 62) \div 63] + 3.7\} \times (e + 3) \\
& + \{ \\
& \quad \uparrow\uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4 \\
& \quad + 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\} \\
& \quad + 4 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} \\
& \quad + 32 + 4 + \{28 \times (f + 1) \times g\} + 20,000 \\
& \quad + \uparrow \{(c \div 8) + 7\} \div 64 \uparrow \times 8 + \uparrow \{(k \div 8) + 7\} \div 64 \uparrow \times 8 \\
& \quad + \text{MAX}\{a \times (e + 3), c \div 8\} \times 104 + \text{MAX}\{a \times (e + 3), k \div 8\} \\
& \quad \times 24 \\
& \quad + \uparrow \{(x \div 4) + 7\} \div 64 \uparrow \times 8 \\
& \quad + \uparrow \{(y - (s \times 592 + t \times 916 + u \times 172)) \div 2\} + 7\} \div 64 \uparrow \\
& \quad \times 8 \\
& \quad + \text{MAX}\{13 \times (e + 3), x \div 4\} \times 88 \\
& \quad + 60 \times \text{MAX}\{21 \times (e + 3), (y - (s \times 592 + t \times 916 + u \\
& \quad \times 172)) \div 2\} \\
& \quad + 44 + 256 + 1,024 \\
& \quad \} \div 1,024 + A + 7 \\
& I \\
& + \sum (J_i) \\
& i=1 \\
& + 6.5 \times 1,024 \times f
\end{aligned}$$

- Add this value if you specified INITIAL in the pd_def_buf_control_area_assign operand or omitted this operand.
 $+ \{[(a + 12) \div 13] \times 1.1 + [(a + 62) \div 63] + 3.7\} \times (e + 7) \text{ (KB)}$

• **64-bit mode**

$$\begin{aligned}
& 40 + b \times 1.3 + c + d + k + 1.6 + x + y + 5 \\
& + \{[(a + 12) \div 13] \times 1.2 + [(a + 62) \div 63] \times 1.5 + 4.1\} \times (e + 3) \\
& + \{ \\
& \quad \uparrow\uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4 \\
& \quad + 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\} \\
& \quad + 4 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} \\
& \quad + 48 + 8 + \{40 \times (f + 1) \times g\} + 20,000 \\
& \quad + \uparrow \{(c \div 8) + 7\} \div 64 \uparrow \times 8 + \uparrow \{(k \div 8) + 7\} \div 64 \uparrow \times 8 \\
& \quad + \text{MAX}(a \times (e + 3), c \div 8) \times 104 + \text{MAX}\{a \times (e + 3), k \div 8\} \times 40 \\
& \quad + \uparrow \{(x \div 4) + 7\} \div 64 \uparrow \times 8 \\
& \quad + \uparrow \{(y - (s \times 600 + t \times 936 + u \times 184)) \div 2\} + 7\} \div 64 \uparrow \times 8 \\
& \quad + \text{MAX}\{13 \times (e + 3), x \div 4\} \times 104 \\
& \quad + 72 \times \text{MAX}\{21 \times (e + 3), (y - (s \times 600 + t \times 936 + u \times 184)) \div 2\} \\
& \quad + 72 + 256 + 1,536 \\
& \quad \} \div 1,024 + A + 7 \\
& I \\
& + \sum (J_i) \\
& i=1 \\
& + 6.5 \times 1,024 \times f
\end{aligned}$$

- Add this value when INITIAL is specified in the pd_def_buf_control_area_assign operand or the operand is omitted.

$$+ \{[(a + 12) \div 13] \times 1.2 + [(a + 62) \div 63] \times 1.5 + 4.1\} \times (e + 7) \text{ (KB)}$$

(2) Formulas for the size of the shared memory used by a dictionary server

This subsection lists and describes the formulas used for calculating the shared memory used by a dictionary server.

For 32-bit mode (KB):

Formula 1 + $\uparrow \{ (\uparrow (40 + (\text{value obtained by adding Formulas 2 through 5})) \div 512 \uparrow \times 512) \} \div 1,024 \uparrow$

For 64-bit mode (KB):

Formula 1 + $\uparrow \{ (\uparrow (72 + (\text{value obtained by adding Formulas 2 through 5})) \div 512 \uparrow \times 512) \} \div 1,024 \uparrow$

For the variables used in the formulas, see (4).

Notes

- Add 3 to the formula if `commit` is specified in either the `pd_dbsync_point` operand or the `pd_system_dbsync_point` operand. The default for the `pd_system_dbsync_point` operand is `commit`.

Otherwise, add Formula 5.

- Add Formula 4 if the `pd_dfw_awt_process` operand is specified.
- If you omit the `pd_dic_shmpool_size` operand, the following value is set:

For 32-bit mode:

$\uparrow \{ (\uparrow (40 + (\text{total of Formulas 2 through 5})) \div 512 \uparrow \times 512) \} \div 1,024 \uparrow$

For 64-bit mode:

$\uparrow \{ (\uparrow (72 + (\text{total of Formulas 2 through 5})) \div 512 \uparrow \times 512) \} \div 1,024 \uparrow$

Condition	Shared memory calculation formula (KB)
Formula 1 (KB)	<p>■ 32-bit mode</p> $ \begin{aligned} & b \times 1.3 \\ & + \{ \\ & \quad \uparrow \uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4 \\ & \quad + 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\} \\ & \quad + 8 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} \\ & \quad + 512 \\ & \} \div 1,024 \\ & + 3.5 + \uparrow (224 \times v \times w) \div 1,024 \uparrow + 0.5 \\ & + \uparrow \{ \\ & \quad \uparrow (28 + (\uparrow (32 + ((\uparrow g \div 127 \uparrow + 1) \times 2,048 + 128)) \div 32 \uparrow \times 32)) \\ & \quad \div 128 \uparrow \times 128 \\ & \} \div 1,024 \uparrow \\ & K \\ & + \sum_{i=1} (Li) \end{aligned} $ <p>■ 64-bit mode</p> $ \begin{aligned} & b \times 1.3 \\ & + \{ \\ & \quad \uparrow \uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4 \\ & \quad + 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\} \\ & \quad + 8 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} \\ & \quad + 1,024 \\ & \} \div 1,024 \\ & + 3.5 + \uparrow (224 \times v \times w) \div 1,024 \uparrow + 0.5 \\ & + \uparrow \{ \\ & \quad \uparrow (56 + (\uparrow (56 + ((\uparrow g \div 127 \uparrow + 1) \times 2,048 + 128)) \div 32 \uparrow \times 32)) \end{aligned} $

Condition	Shared memory calculation formula (KB)
	$\div 128 \uparrow \times 128$ $\} \div 1,024 \uparrow$ K $+ \sum_{i=1} (Li)$
Formula 2 (bytes)	<p>■ 32-bit mode</p> $500 \times 1,024$ $+ (\uparrow 372 \times g \div 16 \uparrow \times 16) + 432 \times h + 112 \times 240$ $+ 5,072 \times (e + 15) + 96 \times z$ $+ 32 \times m + 172 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16$ $+ 48 \times p + 36 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\}$ $+ 68 \times G + 152 \times 120 + 80 + 64^{\#1} + 368^{\#2}$ $+ ((\downarrow (\uparrow (g \div 8) \uparrow + 3) \div 4 \downarrow) \times 4) \times m$ <p>■ 64-bit mode</p> $500 \times 1,024$ $+ (\uparrow 472 \times g \div 16 \uparrow \times 16) + 448 \times h$ $+ (\uparrow 136 \times 240 \div 16 \uparrow \times 16)$ $+ 9,424 \times (e + 15) + 144 \times z$ $+ 48 \times m + 336 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16$ $+ 64 \times p + 72 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\}$ $+ 68 \times G + 184 \times 120 + 96 + 64^{\#1} + 448^{\#2}$ $+ ((\downarrow (\uparrow (g \div 8) \uparrow + 7) \div 8 \downarrow) \times 8) \times m$
Formula 3 (bytes)	<p>32-bit mode</p> $(32 + 16 \times z) \times (G + 1) + 16$ <p>64-bit mode</p> $(48 + 32 \times z) \times (G + 1) + 16$
Formula 4 (bytes)	<p>32-bit mode</p> $72 + 52 \times H + 68 \times z$ <p>If you specify 1 in the <code>pd_dbbuff_trace_level</code> operand, add:</p> $+ 320 \times z$ <p>64-bit mode</p> $96 + 56 \times H + 72 \times z$ <p>If you specify 1 in the <code>pd_dbbuff_trace_level</code> operand, add:</p> $+ 640 \times z$
Formula 5 (bytes)	<p>32-bit mode</p> $(32 + 16 \times z) \times 10 + 16$ <p>64-bit mode</p> $(48 + 32 \times z) \times 10 + 16$

#1: Add this if the `pd_max_ard_process` operand is specified with a value of at least 1.

#2: Add this value if the facility for predicting reorganization time is used.

(3) Formulas for the size of the shared memory used by a back-end server

This subsection lists and describes the formulas used for calculating the shared memory used by a back-end server.

For 32-bit mode (KB):

Formula 1 + $\uparrow \{(\uparrow(40 + (\text{value obtained by adding Formulas 2 through 7})) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$

For 64-bit mode (KB):

Formula 1 + $\uparrow \{(\uparrow(72 + (\text{value obtained by adding Formulas 2 through 9})) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$

For details about the variables used in these formulas, see (4) below.

Notes on Formulas 1 through 9

- If any of the following conditions is satisfied, add Formula 3:
 - Y is specified in the `pd_rdarea_open_attribute_use` operand
 - Y is specified in the `pd_lv_mirror_use` operand
 - The rapid switchover facility is used
- If either of the following conditions is satisfied, add Formula 4:
 - `commit` is specified in the `pd_dbsync_point` operand
 - Y is specified in the `pd_shared_rdarea_use` operand
 - Otherwise, add Formula 7.
- If `commit` is specified in the `pd_inner_replica_control` operand, add Formula 5.
- If the `pd_dfw_awt_process` operand is specified, add Formula 6.
- If you omit the `pd_bes_shmpool_size` operand, the following value is set:
 For 32-bit mode:
 $\uparrow \{(\uparrow(40 + (\text{value obtained by adding Formulas 2 through 7})) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$
 For 64-bit mode:
 $\uparrow \{(\uparrow(72 + (\text{value obtained by adding Formulas 2 through 9})) \div 512 \uparrow \times 512)\} \div 1,024 \uparrow$
- If you specify the `pd_max_resident_rdarea_no` operand, add Formula 8.
- If the value of `pd_max_temporary_object_no` is 1 or greater, add Formula 9.

Condition	Shared memory calculation formula
Formula 1 (KB)	<p>32-bit mode</p> $b \times 1.3$ $+ \{$ $\uparrow \uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4$ $+ 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\}$ $+ 8 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} + 512 + 512^{\#1}$ $\} \div 1,024$ $+ \uparrow \{72 + 8 \times v \times (8 + 3 \times w)\} \div 1,024 \uparrow$ $+ \uparrow \{(\uparrow g \div 127 \uparrow + 1) \times 2,048 + 128\} \div 1,024 \uparrow$ $+ \uparrow \{$ $\uparrow(28 + (\uparrow(32 + ((\uparrow g \div 127 \uparrow + 1) \times 2,048 + 128)) \div 32 \uparrow \times 32))$ $\div 128 \uparrow \times 128\}$ $\} \div 1,024 \uparrow$ M $+ \sum_{i=1} (N_i)$ <p>64-bit mode</p> $b \times 1.3$ $+ \{$ $\uparrow \uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4$ $+ 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\}$ $+ 8 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} + 1,024 + 512^{\#1}$

Condition	Shared memory calculation formula
	$\begin{aligned} & \} \div 1,024 \\ & + \uparrow \{72 + 24 \times v \times (2 + w)\} \div 1,024 \uparrow \\ & + \uparrow \{ \\ & \uparrow (56 + ((\uparrow (56 + ((\uparrow g \div 127 \uparrow + 1) \times 2,048 + 128)) \div 32 \uparrow \times 32)) \\ & \div 128 \uparrow \times 128 \\ & \} \div 1,024 \uparrow \\ & M \\ & + \sum_{i=1} (Ni) \end{aligned}$
Formula 2 (bytes)	<p>■ 32-bit mode</p> $\begin{aligned} & 500 \times 1,024 \\ & + (308 + 48^{#1}) \times g + 432 \times h + 112 \times r \\ & + 5,072 \times (e + 15) + 96 \times z \\ & + 32 \times m + 172 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16 \\ & + 48 \times p + 48 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\} \\ & + 68 \times G + 152 \times F + 80 + 32 \times g + 64^{#2} + 96^{#3} \\ & + ((\downarrow (\uparrow (g \div 8) \uparrow + 3) \div 4 \downarrow) \times 4) \times m \end{aligned}$ <p>■ 64-bit mode</p> $\begin{aligned} & 500 \times 1,024 \\ & + (400 + 56^{#1}) \times g + 448 \times h + 136 \times r \\ & + 9,424 \times (e + 15) + 144 \times z \\ & + 48 \times m + 336 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16 \\ & + 64 \times p + 96 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\} \\ & + 68 \times G + 184 \times F + 96 + 48 \times g + 64^{#2} + 128^{#3} \\ & + ((\downarrow (\uparrow (g \div 8) \uparrow + 7) \div 8 \downarrow) \times 8) \times m \end{aligned}$
Formula 3 (bytes)	<p>32-bit mode</p> $\{ \{ ((\uparrow \uparrow g \div 8 \uparrow \div 4 \uparrow) \times 4) + 8 \} \times (a \times [[e + 3] + e + 15]) \}$ <p>64-bit mode</p> $\{ \{ ((\uparrow \uparrow g \div 8 \uparrow \div 8 \uparrow) \times 8) + 8 \} \times (a \times [e + 3] + e + 15) \}$
Formula 4 (bytes)	<p>32-bit mode</p> $(32 + 16 \times z) \times (e \times 2 + 7 + 1) + 16$ <p>64-bit mode</p> $(48 + 32 \times z) \times (e \times 2 + 7 + 1) + 16$
Formula 5 (bytes)	$56 \times E + 16$
Formula 6 (bytes)	<p>32-bit mode</p> $72 + 52 \times H + 68 \times z$ <p>If you specify 1 in the <code>pd_dbbuff_trace_level</code> operand, add:</p> $+ 320 \times z$ <p>64-bit mode</p> $96 + 56 \times H + 72 \times z$ <p>If you specify 1 in the <code>pd_dbbuff_trace_level</code> operand, add:</p> $+ 640 \times z$
Formula 7 (bytes)	<p>32-bit mode</p> $(32 + 16 \times z) \times 10 + 16$

Condition	Shared memory calculation formula
	64-bit mode $(48 + 32 \times z) \times 10 + 16$
Formula 8 (bytes)	$16 + 112 + (48 + 48 \times Q) + (48 + 32 \times R)$
Formula 9 (bytes)	$16 + 80 \times S$

#1: Add this value if neither `pd_max_list_user` nor `pd_max_list_count` operand is 0.

#2: Add this if the value of the `pd_max_ard_process` operand is at least 1.

#3: Add this if the value of the `pd_max_reflect_process_count` operand is at least 1.

(4) Variables used in the formulas

a: Value of `pd_max_access_tables` operand

b: Value of `pd_sql_object_cache_size` operand

c: Value of `pd_table_def_cache_size` operand

d: Value of `pd_auth_cache_size` operand

e: Value of `pd_max_users` operand^{#1}

f: Total number of back-end servers

g: Value of `pd_max_rdarea_no` operand

h: Value of `pd_max_file_no` operand

k: Value of `pd_view_def_cache_size` operand

m: Number of global buffers for index

If *Y* is specified in the `pd_dbbuff_modify` operand, add the `pd_max_add_dbbuff_no` operand value of the server definition to the number of `pdbuffer` statements related to the server.

p: Value of `pd_lck_until_disconnect_cnt` operand

q: $\text{MIN}(e + 3, p)$

r: Value of `pd_assurance_index_no` operand

s: Number of plug-ins installed

t: Total number of plug-in functions used with DML^{#2}

u: Total number of parameters for the plug-in functions used with DML^{#2}

v: Value of `pd_max_list_users` operand

w: Value of `pd_max_list_count` operand

x: Value of `pd_type_def_cache_size` operand

y: Value of `pd_routine_def_cache_size` operand

z: Total number of global buffers (number of `pdbuffer` operands specified)

If *Y* is specified in the `pd_dbbuff_modify` operand, add the `pd_max_add_dbbuff_no` operand value of the server definition to the number of `pdbuffer` statements related to the server.

A: Value of `pd_registry_cache_size` operand

E: Value of the `pd_inner_replica_control` operand

F: Value of the `pd_assurance_table_no` operand

G: Maximum number of transactions in the server ($2 \times e + 7$)

H: Value of the `pd_dfw_awt_process` operand

I : Total number of `pdplgprm` operands specified in the front-end server

J_i : Size of the shared memory specified in the i^{th} `pdplgprm` operand specified in the front-end server

K : Total number of `pdplgprm` operands specified in the dictionary server

L_i : Size of the shared memory specified in the i^{th} `pdplgprm` operand specified in the dictionary server

M : Total number of `pdplgprm` operands specified in the back-end server

N_i : Size of the shared memory specified in the i^{th} `pdplgprm` operand specified in the back-end server

Q : Value of the `pd_max_resident_rdarea_no` operand

R : Value of the `pd_max_resident_rdarea_shm_no` operand

S : Value of the `pd_max_temporary_object_no` operand

U : Number of tables that will use the free space reuse facility

#1: For a dictionary server, use the value of the `pd_max_dic_process` operand. For a back-end server, use the value of the `pd_max_bes_process` operand. If the `pd_max_dic_process` or `pd_max_bes_process` operand is omitted, use the value of the `pd_max_users` operand.

#2: You can use the following SQL statement to obtain the total number of plug-in functions and the total number of parameters for the plug-in functions used with DML:

```
SELECT COUNT(*), SUM(N_PARAM) FROM MASTER.SQL_PLUGIN_ROUTINES
WHERE PLUGIN_NAME = 'plug-in-name'
AND (TIMING_DESCRIPTOR = 'ADT FUNCTION'
     OR TIMING_DESCRIPTOR IS NULL
     OR TIMING_DESCRIPTOR = 'BEFORE_INSERT'
     OR TIMING_DESCRIPTOR = 'AFTER_INSERT'
     OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE'
     OR TIMING_DESCRIPTOR = 'AFTER_UPDATE'
     OR TIMING_DESCRIPTOR = 'BEFORE_DELETE'
     OR TIMING_DESCRIPTOR = 'AFTER_DELETE'
     OR TIMING_DESCRIPTOR = 'BEFORE_PURGE_TABLE'
     OR TIMING_DESCRIPTOR = 'AFTER_PURGE_TABLE'
     OR TIMING_DESCRIPTOR = 'INDEX_SEARCH'
     OR TIMING_DESCRIPTOR = 'INDEX_COUNT'
     OR TIMING_DESCRIPTOR = 'INDEX_INSERT'
     OR TIMING_DESCRIPTOR = 'INDEX_BEFORE_UPDATE'
     OR TIMING_DESCRIPTOR = 'INDEX_AFTER_UPDATE'
     OR TIMING_DESCRIPTOR = 'INDEX_DELETE'
     OR TIMING_DESCRIPTOR = 'PURGE_INDEX'
     OR TIMING_DESCRIPTOR = 'INDEX_MAINTENANCE_DEFERRED'
     OR TIMING_DESCRIPTOR = 'BEFORE_INSERT_DC'
     OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE_DC'
     OR TIMING_DESCRIPTOR = 'BEFORE_DATA_CHECK'
     OR TIMING_DESCRIPTOR = 'AFTER_DATA_CHECK')
```

15.2.5 Formula for size of shared memory used by global buffers

(1) When the standby-less system switchover (effects distributed) facility is not used

The size of the shared memory used by global buffers is calculated for each dictionary server and back-end server, using Formula 1. If the calculations are made for each server machine, the values can differ depending on the options specified in the `pdbuffer` statement, as shown in the following table.

Table 15–9: Calculation conditions depending on the options specified in the `pdbuffer` statement (when the standby-less system switchover (effects distributed) facility is not used)

pdbuffer statement option	Calculation condition
-r	If the server has an RDAREA for which -r is specified, that RDAREA is used in the calculation.
-i	If the server has an RDAREA that stores an index for which -i is specified, that server is used in the calculation.

pdbuffer statement option	Calculation condition
-b	If the server has an RDAREA for which -b is specified, that server is used in the calculation.
-o	If there are any RDAREAs in the server that are not specified with <code>pdbuffer -r</code> , they are used in the calculation.

If γ is specified in the `pd_dbbuff_modify` operand, add Formula 2. The total value determined from Formulas 1 and 2 is the required shared memory area for the server's global buffers.

If you specify `fixed` for the `pd_dbbuff_attribute` operand, pages are locked in real memory, which reduces the real memory portion of the virtual memory by the same size. In addition, the same size is allocated from virtual memory, which consists of the remaining real memory and swap area.

If the `pdbuffer` operand is omitted, HiRDB calculates the shared memory area automatically, so it need not be estimated.

Formulas	Shared memory calculation formula (KB)
Formula 1	<p>32-bit mode</p> $n \sum_{i=1} \{ \uparrow \{ 752 + 64 + (296 + 64^{\#1}) \times (Pi + 4) + (124 + 80^{\#2} + 96 \times A \times Mi) \times Ui \} \div 4,096 \uparrow \times 4,096 + Si \times \{ Pi + 4 + (Ui \times Mi \times A) \} \} \div 1,024$ <p>64-bit mode</p> $n \sum_{i=1} \{ management\ region\ part + data\ storage\ part \} \div 1,024$ <p><i>management region part:</i></p> $\uparrow \{ 944 + 64 + (480 + 112^{\#1}) \times (Pi + 4) + (176 + 96^{\#2} + 136 \times A \times Mi) \times Ui \} \div 4,096 \uparrow \times 4,096$ <p><i>data storage part:</i></p> $Si \times \{ Pi + 4 + (Ui \times Mi \times A) \}$
Formula 2	<p>32-bit mode</p> $\{ \uparrow (\uparrow (s \times 1,024 \div 2) \div 8 \uparrow + 112) \div 2,048 \uparrow \times 2,048 \times \uparrow a \div (s \times 1,024) \uparrow \} \div 1,024$ <p>64-bit mode</p> $\{ \uparrow (\uparrow (s \times 1,024 \div 2) \div 8 \uparrow + 144) \div 2,048 \uparrow \times 2,048 \times \uparrow a \div (s \times 1,024) \uparrow \} \div 1,024$

n : Number of global buffer pools

i : Global buffer pool definitions to be calculated

P : Number of global buffer sectors

A : If the asynchronous READ facility is used, 2; if it is not used, 1

M : Maximum number of batch input pages

If at least 1 is specified in the `pd_max_ard_process` operand, this is twice the specified value.

U : Maximum number of concurrently executable prefetch operations

S : Maximum page length of the RDAREAs allocated to global buffer

s : Value of `SHMMAX`

a: Total from Formula 1

#1: Add this value in the case of a global buffer for LOB.

#2: Add this value if at least 1 is specified in the `pd_max_ard_process` operand.

(2) When the standby-less system switchover (effects distributed) facility is used

When the standby-less system switchover (effects distributed) facility is used, the size of the shared memory used by global buffers is obtained for each unit using the formula. If the calculations are made for each unit, the values can differ depending on the options specified in the `pdbuffer` statement, as shown in the following table.

Table 15–10: Calculation conditions depending on the options specified in the `pdbuffer` statement (when the standby-less system switchover (effects distributed) facility is used)

pdbuffer statement option	Calculation condition
<code>-r, -b</code>	If the unit has an RDAREA for which <code>-r</code> is specified and which belongs to the same HA group, that RDAREA is used in the calculation.
<code>-i</code>	If an RDAREA that contains the index specified with <code>-i</code> belongs to the same HA group, that RDAREA is used in the calculation.
<code>-o</code>	If there are any RDAREAs in the same HA group that are not specified with <code>pdbuffer -r</code> , they are used in the calculation.

When `fixed` is specified for the `pd_dbbuff_attribute` operand, pages are locked in real memory, which reduces the real memory portion of the virtual memory by the same size. In addition, the same size is allocated from virtual memory, which consists of the remaining real memory and swap area.

If the `pdbuffer` operand is omitted, HiRDB calculates the shared memory area automatically, so it need not be estimated.

Shared memory calculation formula (KB)
32-bit mode $\sum_{i=1}^n \{ (96 + ((752 \times (A + B)) + (288 \times (F + (8 \times (A + B))))) + 8 \times F \times (A + B) + 16) + H + D) + 2,048 + G + (E \times F + (8 \times (A + B))) \} \div 1,024$ 64-bit mode $\sum_{i=1}^n \{ \text{management region part} + \text{data storage part} \} \div 1,024$ <p><i>management region part:</i></p> $((144 + ((944 \times (A + B)) + (464 \times (F + (8 \times (A + B))))) + (16 \times F \times (A + B))) + 16 + H + D)$ <p><i>data storage part:</i></p> $2,048 + G + (E \times F + (8 \times (A + B)))$

n: Number of global buffer pools allocated to this unit

i: Global buffer pool definitions to be calculated

A: Number of host BESSs

B: Maximum number of acceptable guest BESSs

C: Number of batch input pages (value specified in `pdbuffer -p`)

D: Add this value if the prefetch facility is used (`pdbuffer -m` specified):

In the 32-bit mode:

$$2 \times (((80 \times U \times C) + (80 \times U) + (124 \times U) + (8 \times U \times C)) \times (A + B))$$

In the 64-bit mode:

$$2 \times (((112 \times U \times C) + (96 \times U) + (176 \times U) + (16 \times U \times C)) \times (A + B))$$

E: The value depends on the options specified in the `pdbuffer` statement. The following table lists and describes the options and formulas:

pdbuffer statement option	Formula for the maximum value
-r, -b	(MAX (buffer size (value of <code>pdbuffer -l</code>), MAX (page size of the specified RDAREA that belongs to the same HA group)))
-i	(MAX (buffer size (value of <code>pdbuffer -l</code>), MAX (page size of the RDAREA that stores the index specified with -i and that belongs to the same HA group)))
-o	(MAX (buffer size (value of <code>pdbuffer -l</code>), MAX (page size of the RDAREA in the same HA group that is not specified with <code>pdbuffer -r</code>)))

F: Number of buffer sectors (value of `pdbuffer -n`)

G: Add this value if the prefetch facility is used (`pdbuffer -m` specified):

$$2 \times ((E \times U \times C) \times (A + B))$$

H: Add this value if LOB RDAREA is specified (`pdbuffer -b` specified):

In the 32-bit mode:

$$64 \times (F + (8 \times (A + B)))$$

In the 64-bit mode:

$$112 \times (F + (8 \times (A + B)))$$

U: Maximum concurrent prefetch count (value of `pdbuffer -m`)

15.2.6 Formulas for size of memory required during SQL execution

(1) Procedure for obtaining the size of the memory required during execution of rapid grouping facility

If `PDSQLOPTLVL` is specified in the client environment definition, `pd_optimize_level` is specified in the system common definition or front-end server definition, or this operand is omitted, executing an SQL statement that satisfies the applicable conditions will activate the rapid grouping facility. In such a case, HiRDB allocates process private area on the basis of the value of the `PDAGGR` operand in the client environment definition. The size of the memory can be obtained from the following formula (in bytes). The size of the memory required during execution of rapid grouping facility should be calculated for the server machine defining the back-end server only.

Formula

$$e + \uparrow d \div 4 \uparrow \times 4 + \uparrow (17 + 4 \times a + 4 \times b + c + d) \div 4 \uparrow \times 4 \times (N + 1) \text{ (bytes)}$$

a: Number of columns subject to grouping

b: Number of operations by set functions

Each of `COUNT`, `SUM`, `MAX`, and `MIN` is counted as 1.

Each of `AVG (COUNT)` and `AVG (SUM)` is counted as 2.

c: Length of rows subject to grouping (see Table 15-11 Length of column subject to grouping and length of operation area for set functions)

d: Length of operation area for set functions (see Table 15-11 Length of column subject to grouping and length of operation area for set functions)

e : 32-bit mode: $\text{MAX}(4 \times N \times 24, 16,408)$
 64-bit mode: $\text{MAX}(8 \times N \times 40, 32,808)$

N : Value of the PDAGGR operand in the client environment definition

Table 15–11: Length of column subject to grouping and length of operation area for set functions

Column's data type	Column length	Length of operation area for set function ^{#1}
INTEGER	4	6
SMALLINT	2	4 ^{#2}
DECIMAL (p, s)	$\uparrow(p+1) \div 2 \uparrow$	$\uparrow(p+7) \div 2 \uparrow$ ^{#3}
FLOAT	8	10
SMALLFLT	4	6
INTERVAL YEAR TO DAY	5	8
INTERVAL HOUR TO SECOND	4	6
CHAR (n)	n	$n+3$
VARCHAR (n)	$n+2$	$n+5$
NCHAR (n)	$2 \times n$	$2 \times n+2$
NVARCHAR (n)	$2 \times n+2$	$2 \times n+4$
MCHAR (n)	n	$n+3$
MVARCHAR (n)	$n+2$	$n+5$
DATE	4	6
TIME	3	6
BLOB (n)		--
BINARY (n)	$n+2$	$n+5$

Legend:

--: Not applicable

#1

If the set function is COUNT, the length of the operation area is always 6 regardless of the data type.

#2

If the set function is AVG or SUM, the length of the operation area is 6.

#3

If the set function is AVG or SUM, the length of the set function operation region is the following value:

If the set function value type is DECIMAL and precision is 29 digits: 18

If the set function value type is DECIMAL and precision is 38 digits: 23

For details about the data type rules of set functions, see *Set functions* in the manual *HiRDB Version 9 SQL Reference*.

(2) Procedure for obtaining the size of the memory required when data suppression by column is specified

The following formula can be used to obtain the size of the memory (in bytes) required to access a table for which data suppression by column is specified (table for which SUPPRESS is specified in the column definition of CREATE TABLE).

Formula

$$a + 128 \text{ (bytes)}$$

a : Sum of the lengths of columns in the table for which data suppression by column is specified

(3) Procedure for obtaining the size of the memory required during hash join and subquery hash execution

If you specify the `PDADDITIONALOPTLVL` operand in the client environment definition or the `pd_additional_optimize_level` operand in the HiRDB system definition, the SQL extension optimizing option becomes available. If you specify an application of hash join, subquery hash execution (`APPLY_HASH_JOIN`) with this SQL extension optimizing option, the system allocates the following size of process private area when a table join or subquery SQL statement is executed:

Formula**32-bit mode**

$$a$$

$$\sum_{i=1} (13 \times 1,024 + 6 \times 1,024 \times b + c)$$

64-bit mode

$$a$$

$$\sum_{i=1} (13 \times 1,024 + 7 \times 1,024 \times b + c) \text{ (bytes)}$$

a : Maximum number of hash joins in the `SELECT` statement

For details about the maximum number of hash joins in the `SELECT` statement, see the *HiRDB Version 9 UAP Development Guide*.

b : Obtain the hash join processing to be applied on the basis of the number of hash table rows, then determine the this value from the following table:

Guidelines for the number of hash table rows	Hash join processing to be applied		Value of b
1,500 or less	Batch hash join		0.5
$1,500 \times (\text{packet split count} \div 3)$ or less	Packet split Hash join	1-level packet split	1
$1,500 \times (\text{packet split count} \div 3)^2$ or less		2-level packet split	2
Greater than $1,500 \times (\text{packet split count} \div 3)^2$		3-level packet split	3

Number of hash table rows: For join, it is the inner table count; for subquery, it is the subquery search count excluding the predicates that contain external reference rows in the search condition.

Packet split count: $\text{MIN} \{ \downarrow (\text{size of hash table} \div 2) \div \text{page length of hash table} \downarrow, 64 \}$

Hash table size: Value of the `pd_hash_table_size` specified in the HiRDB system definition or the value of the `PDHASHTBLSIZE` operand specified in the client environment definition.

Page length of hash table: Select the page length of hash table corresponding to c (maximum length of hash table row) from the following table:

Maximum length of hash table row	Page length of hash table (bytes)
0 to 1,012	4,096
1,013 to 2,036	8,192
2,037 to 4,084	16,384
4,085 to 16,360	32,768

Maximum length of hash table row	Page length of hash table (bytes)
16,361 to 32,720	$\uparrow (\text{maximum length of hash table row} + 48) \div 2,048 \uparrow \times 2,048$

c: Maximum length of hash table row

For details about the length of a hash table row, see the *HiRDB Version 9 UAP Development Guide*.

(4) Procedure for obtaining the size of the memory required when the snapshot method is used

If the `pd_pageaccess_mode` operand is omitted, or if `SNAPSHOT` is specified, then the page access method for data retrieval uses the snapshot method when an SQL statement for which the snapshot method is applicable is executed. At this time, memory in the process private area is allocated automatically, as shown below, based on the page size of the table or index storage RDAREA.

Formula

$$a \times 2 \text{ (bytes)}$$

a: Maximum page length in the RDAREA where the relevant table or index is stored

However, LOB RDAREAs are excluded.

(5) Determining the size of the memory required to retrieve the first *n* records

If the function for retrieving *n* rows of search results from the top is used, you can retrieve *n* rows from the top of the search results (or from the location resulting from skipping as many rows from the top as specified by the user as an offset).

If the number of rows specified in the `LIMIT` clause is 1 or greater and the value of (number of offset rows + number of rows specified in the `LIMIT` clause) is 32,767 or less, as many rows are retained in memory as will fit in (number of offset rows + number of rows specified in the `LIMIT` clause). The size of the process private area to be allocated can be determined by the formula shown below. If the value of (number of offset rows + number of rows specified in the `LIMIT` clause) is 32,768 or greater, see *Chapter 18. Determining Work Table File Size* because a work table is created.

Formula

$$\{100 + (a + 2) \times (\text{number of offset rows} + \text{number of rows specified in the LIMIT clause})\} \times b \text{ (bytes)}$$

a: Row length

The row length cannot exceed 32,720 bytes. The row length is calculated with the following formula:

m

$$\sum_{i=1} (Ai) + 2 \times m + 4 + c \text{ (bytes)}$$

i=1

m: Number of rows specified in the selection formula, `GROUP BY` clause, or `ORDER BY` clause

Add 1 if the `FOR UPDATE` clause is specified. However, if `ROW` is specified in the selection formula, this is the total number of rows in the table.

Ai: Data length of the *i*th column of the records stored in the first *n* records of the allocation area

For details about column data length, see *Table 16-1 List of data lengths*, and determine the length beginning by assigning the defined length to *d*.

The data length is set to 12 bytes for a column whose data type is BLOB, character string whose defined length is 256 bytes or greater (including national and mixed character strings), or BINARY that does not belong to any of the following:

- Columns specified in a selection formula with the `DISTINCT` clause specified
- A query specification selection formula using a concatenation operation based on `UNION [ALL]`
- Columns specified in the `ORDER BY` clause

Also, if the `FOR UPDATE` clause is specified and 1 is added for *m*, use 12 bytes for *Ai*.

c : 8

However, in the following cases, use 0.

- There is an exclusive lock in the EX mode on the retrieval table
- WITHOUT LOCK is specified
- The rapid grouping facility is specified
- Multiple tables are combined

b : Number of maintenance areas for the first n records

The number of maintenance areas for the first n records is calculated with the following formula:

$$1 + \text{number of UNION [ALL] clause specifications}$$

(6) Determining the size of the memory required for executing SQL statements specifying an index-type plug-in function as a search condition

To determine the size of memory that is allocated in the process private area when an SQL statement specifying an index-type plug-in function as a search condition is executed, use the following formula:

Formula

$$a \times 500 + (20 + 6) \times 800 + 16 \text{ (bytes)}$$

a : Row length. To determine the row length, use the following formula:

m

$$\sum_{i=1}^m (Ai) + 4 \times (m + 2) + 12 + 4 + 8 \text{ (bytes)}$$

$i=1$

m : Number of columns specified in the selection formula, join condition, GROUP BY clause, or ORDER BY clause

If you specified the FOR UPDATE clause, add 1. If ROW is specified in the selection formula, the total number of rows in the table is assumed.

Ai : Length of column data i in the row to be retrieved

For details about column data length, see *Table 16-1 List of data lengths*, and determine the length beginning by assigning the defined length to d .

A length of 12 bytes is assumed for a column with BLOB data or character string data with a defined length of 256 bytes or greater (including national character data and mixed character string data) that is none of the following:

- Column specified in the join condition (join column)
- Column specified in the selection formula with the DISTINCT clause specified
- Column specified in the selection formula in a subquery of a quantified predicate
- Column specified in the selection formula in a subquery of an IN predicate
- Selection formula in the subquery that is the target of Set Operation due to UNION [ALL] or EXCEPT [ALL]
- Column specified in the ORDER BY clause

If the FOR UPDATE clause is specified, Ai corresponding to 1 that was added to m is 12 bytes.

(7) Determining the size of the memory required to use the facility for output of extended SQL error information

When the facility for output of extended SQL error information is used, a process private area is allocated in the following cases:

(a) When the OPEN statement is executed

Formula

32-bit mode

$$(16 + 16 \times m) + a \text{ (bytes)}$$

64-bit mode

$$(24 + 24 \times m) + a \text{ (bytes)}$$

a : Total data length of ? parameters or embedded variables

m

$$a = \sum_{i=1} (a_i)$$

$i=1$

m : Number of ? parameters or embedded variables in the SQL statement

a_i : Data length of the i^{th} ? parameter or the embedded variable

For details about the data length, see *Table 15-5 Data length of embedded variables and ? parameters*.

(b) When the PREPARE statement of the definition SQL is executed

Formula

$$SQL \text{ statement length} + 20 \text{ (bytes)}$$

(8) Determining the size of memory required for defining substructure indexes or for updating tables in which a substructure index is defined

(a) If a substructure index is defined

Use the following formula to calculate the process private area used when a substructure index is defined with the CREATE INDEX statement of definition SQL.

Formula

$$(\text{index key length}^{\#} \times 100 + 64) \text{ (bytes)}$$

$\#$

The maximum definition length of the substructure index defined in the table.

(b) If a table in which a substructure index is defined is updated

Use the following formula to calculate the amount of process private area used for updating a table in which a substructure index has been defined with the INSERT, UPDATE or DELETE statement of data manipulation SQL.

Formula

$$(\text{index key length}^{\#1} \times 100 + 64 + 128) + \sum (\text{index key length} + 128)^{\#2} \text{ (bytes)}$$

$\#1$

The maximum definition length of the substructure index defined in the table.

$\#2$

The number of substructure indexes that specify USING UNIQUE TAG.

(9) Determining the size of the memory required to execute data manipulation SQL statements on compressed columns

If SQL statement execution, data storage processing, or extraction processing involves compressed columns, HiRDB allocates a process private area whose memory size is as shown below.

Formula

$$\text{MIN}(\text{split compression size, definition length of compressed column})^{\#} \times C + L \text{ (bytes)}$$

C: If any of the following conditions is true, 2; if not, 1:

- The SUBSTR function is used.
- The POSITION function is used.
- Backward deletion/updating of data is performed.

L: Page length of the RDAREA containing the compressed table to be processed by the SQL statement
If multiple RDAREAs are processed, use the maximum page length.

#

Use the maximum value for all the compressed columns subject to SQL statement processing.

(10) Determining the size of the communication memory required for executing SQL statements using a HiRDB parallel server configuration

When SQL statements are executed, the system allocates a process private area of the following size for communication between BESs or for FES-BES communication:

Formula

$$4 \times 1,024 \times 3$$

$$\times (\text{maximum number of tables specified in one SQL statement} \times \text{number of BESs with tables of maximum partition size}$$

$$+ \text{number of floating servers}^{\#} \times \text{number of BESs in system}) \times \text{value of pd_max_users (bytes)}$$

#

For SQL statements that use floating servers, specify the maximum number of tables specified in the SQL statement. Specify 0 for SQL statements that do not use floating servers. For details about SQL statements that use floating servers, see *Allocating floating servers* in the *HiRDB Version 9 UAP Development Guide*.

15.2.7 Formula for size of memory required during SQL preprocessing

(1) Size of memory required when no stored procedure is used

If no stored procedure is used, the following formula can be used to obtain the size of the memory that is allocated during SQL preprocessing (KB).

Formula

$$\uparrow \{$$

$$(2,539 + Si \times 70 + Pi \times 20 + Ti \times 980 + Ci \times 68 + Wi \times 818 + Ki \times 416 + Li \times 5$$

$$+ Di \times 116 + Ari \times 108 + Gi \times 44 + Ori \times 10 + Sli \times 40 + Upi \times 96 + Fi \times 90$$

$$+ Ti \times Cwi \times 48 + \text{MAX}(Pi, Wpi) \times 52 + \text{MAX}(Ti, Sli - 1) \times 96$$

$$+ \text{MAX}(Ti \times 2, Wi) \times 24 + \text{MAX}(Ti \times 3, Wi) \times 24$$

$$+ \text{MAX}\{\text{MAX}(Ti, Ori + Gi + Si + Fi), Sli - 1\} \times 24$$

$$\} \times 1.2 \div 1,024 \uparrow \times CLS \text{ (KB)}$$

Si: Number of items to be retrieved in SQL statements

Pi: Number of embedded variables, ? parameters, or SQL parameters in SQL statements

Ti: Number of table names in SQL statements

Ci: Number of column names in SQL statements

Wi: Number of predicates used in Boolean operators (AND and OR) in SQL statements

Ki: Number of literals in SQL statements

Li: Total length of literals in SQL statements (bytes)

Di: Total number of storage conditions defined in SQL statements

Ari: Number of arithmetic operations and concatenation operations in SQL statements

Gi: Number of columns specified in `GROUP BY` clause of SQL statements

Ori: Number of column specification or sort item specification numbers in `ORDER BY` clause of SQL statements

Fi: Total number of set functions and scalar functions in SQL statements

Sli: Number of queries specified in SQL statements

Upi: Number of columns to be updated in SQL statements

Cwi: Number of `WHENS` in `CASE` expression of SQL statements

Wpi: Number of variables corresponding to `WITH` clause of SQL statements

CLS: Number of areas generated per access path in an SQL object[#]

[#]: The following formula can be used to obtain the number of areas where one access path is generated in an SQL object.

Formula

When `SELECT_APSL` is applied[#]

$$a + b \times 4 + c + d + e \times 2$$

When `SELECT_APSL` is not applied[#]

$$a + b + c + d + e$$

a: Number of front-end servers

Specify 1 for the number of front-end servers.

b: Number of tables

Use the following formula to obtain the number of tables:

Number of base tables + number of correlation names

c: Number of set operation servers

If a set function is specified, specify 1; otherwise, specify 0.

d: Number of queries specifying `GROUP BY`, `DISTINCT`, or `ORDER BY` clause

e: Number of join servers

Use the following formula to obtain the number of join servers:

b - number of queries in SQL statement

[#]: The access path display utility (`pdvwopt`) can be used to determine whether `SELECT_APSL` is applied. For details about the access path display utility (`pdvwopt`), see the manual *HiRDB Version 9 Command Reference*.

(2) Procedure for obtaining the size of the memory required when using stored procedures

If stored procedures are used, the size of the memory (in KB) to be allocated during SQL preprocessing is the value obtained from the formula shown in (1) above plus the length of the procedure control object for each stored procedure. For the formula for obtaining the length of a procedure control object, see the section on the `pd_sql_object_cache_size` operand of the system common definition. For details about the length of the procedure control object per stored procedure, see *Formula for determining the size of the routine control object of a routine* in the manual *HiRDB Version 9 System Definition*.

15.2.8 Formula for size of memory required during BLOB data retrieval or updating (front-end server)

Use the following formula to determine the size of the memory required during BLOB data retrieval or updating.

Formula

$$a + b + 7 \text{ (KB)}$$

a: Maximum value from the following formula for BLOB input variables or output variables specified in one SQL statement.

$$\uparrow \left\{ \begin{array}{l} c \\ \sum_{i=1}^c (\text{actual length of BLOB input variable } i^{\#1} \times 2 + 58) + \\ d \\ \sum_{j=1}^d (\text{specified length of BLOB output variable } j^{\#2} + 26) \end{array} \right\} \div 1,024 \uparrow$$

#1: This is the actual length of BLOB data passed as embedded variables from the UAP to the HiRDB server.

#2: This is the declared length of the UAP embedded BLOB data type variables received from the UAP and returned from HiRDB to the UAP. If it is an INSERT or SELECT statement, the BLOB type reflected from the SELECT side is an output variable.

b: Maximum value from the following formula for a combination of SQL statements performing join retrieval with simultaneously open cursors:

$$256 \times \text{number of concurrently open cursors}$$

c: Number of input variables

d: Number of output variables

15.2.9 Formula for size of memory required during BLOB data retrieval or updating (back-end server or dictionary server)

Use the following formula to determine the size of the memory required during BLOB data retrieval or update.

Formula

$$a + b \text{ (KB)}$$

a: Maximum value from the following formula for BLOB input variables or output variables specified in one SQL statement:

$$\uparrow \left\{ \begin{array}{l} c \\ \sum_{i=1}^c (\text{actual length of BLOB input variable } i^{\#} + 118 + 70 \times \text{number of output variables}) \end{array} \right\} \div 1,024 \uparrow$$

#: This is the actual length of BLOB data passed as embedded variables from the UAP to the HiRDB server.

b: The result of the following formula is the largest memory value for a combination of SQL statements that search BLOB data while there are simultaneously open cursors.

$$d \sum_{i=1}^d \{280 + 184 \times (\text{number of tables specified in } SQLi + 1)\}$$

c: Number of input variables

d : Number of cursors

15.2.10 Formula for size of memory required during block transfer or array FETCH (front-end server)

To determine the size of the memory required for block transfer or array `FETCH`, use the following formulas:

Condition		Value specified in the <code>PDBLKBUFSIZE</code> operand	
		Omitted or 0	1 or greater
An array-type embedded variable is specified in the <code>INTO</code> clause of <code>FETCH</code> statement		Formula 1	
An array-type embedded variable is not specified in the <code>INTO</code> clause of <code>FETCH</code> statement	<code>PDBLK</code> operand is omitted or set to 1	--	Formula 2
	<code>PDBLK</code> operand is set to 2 or greater	Formula 1	

Legend:

--: Not applicable

Formula 1

$$\uparrow \{864 + 16 \times a + (6 \times a + 2 \times d + b) \times c\} \div 1,024 \uparrow \text{ (KB)}$$

a : Number of retrieval items specified in the `SELECT` clause

b : Data length per row in the retrieval result obtained by the `FETCH` statement (sum of the maximum length of each column in bytes)

c : Value specified by the `PDBLK` operand or the number of array columns

d : Number of selection formulas with `BINARY` type specified in the search item specified in the `SELECT` clause

Formula 2

$$\text{MAX}(X_1, X_2) \text{ (kilobytes)}$$

$$X_1: \uparrow (864 + 22 \times a + 2 \times c + b) \div 1,024 \uparrow$$

X_2 : Value of the `PDBLKBUFSIZE` operand

a : Number or retrieved items that is specified in the `SELECT` clause

b : Data length per row in the retrieval results obtained by the `FETCH` statement (sum of the length of each column that is actually obtained, in bytes)

c : Number of selection formulas with `BINARY` type specified in the search item specified in the `SELECT` clause

15.2.11 Memory required by in-memory data processing

Use the following formulas to calculate the memory required by in-memory data processing.

For a HiRDB parallel server configuration, calculate the `RDAREA` used as in-memory separately for each server machine.

Formula

$$\text{Formula 1} + D \times 2 \text{ (KB)}$$

Formula 1

n

$$\sum_{i=1}^n \{ 736 + 32 \times A + 48 + 448 \times B + 2,048 + C \times B \} \div 1,024 \uparrow \text{ (KB)}$$

- n*: Number of in-memory RDAREAs
- A*: Number of HiRDB files that constitute an in-memory RDAREAs
- B*: Total number of pages of in-memory RDAREAs
- C*: Page size of in-memory RDAREAs
- D*: Value of Formula 2

Formula 2 (number of shared memory segments used by in-memory data buffer)

$$\uparrow \text{value of Formula 1} \div (\text{value of SHMMAX operand} \times 1,024) \uparrow$$

The value found by Formula 2 is used to calculate the `pd_max_resident_rdarea_shm_no` operand or the operating system parameters.

16

Determining RDAREA Size

This chapter explains how to determine the size of each type of RDAREA.

16.1 Determining the size of a user RDAREA

This section explains how to determine the size of a user RDAREA.

16.1.1 Calculating the size of a user RDAREA

(1) Formula for calculating the size of a user RDAREA

The following formula is used to calculate the size of a user RDAREA.

Formula

Size of user RDAREA (bytes)

$$= \text{Page length of the user RDAREA}^{\#1} \times \text{total number of pages in the user RDAREA}^{\#2}$$

#1: This is the page length specified in the `create rdarea` statement for the database initialization utility or database structure modification utility.

#2: See (2) as follows.

(2) Formula for calculating the total number of pages in a user RDAREA

The following formula is used to calculate the total number of pages in a user RDAREA.

Total number of pages in a user RDAREA (pages)

$$= \text{total number of pages in the directory page part} + \text{total number of pages in the data page part}$$

(a) Formula for calculating the total number of pages in the directory page part

Total number of pages in the directory page part (pages) =

$$6 \times (n + 1) + \lceil 20,480 \div P \rceil \times 2$$

n

$$+ \sum_{i=1}^n \{ \lceil d_i \div b \rceil + \lceil d_i \div f \rceil \}$$

$i=1$

n : Number of HiRDB files that constitute the user RDAREA

P : Page length of the user RDAREA (bytes)

$$b: \lfloor (P - 20) \div (\lceil S \div 32 \rceil \times 8 + 56) \rfloor$$

$$f: \lfloor (125 \times P) \div (16 \times b) \rfloor \times b$$

d_i : Number of segments for each HiRDB file specified with the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`)

S : Number of pages for one segment (segment size) specified with the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`)

(b) Formula for calculating the total number of pages in the data page part

Total number of pages in the data page part (pages) =

e

$$\sum_{i=1}^e \{ \lceil (\alpha_i \div S) \rceil \times S \}$$

$i=1$

e

$$+ \sum_{i=1}^k \{ \lceil (\beta_i \div S) \rceil \times S \}$$

e : Total number of pages stored in the user RDAREA

k : Total number of indexes stored in the user RDAREA

S : Number of pages for one segment (segment size) specified with the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`)

α_i : Number of pages required to store a column other than `BINARY` columns defined as branching in each table
See 16.1.2 *Calculating the number of table storage pages*.

β_i : Number of pages required to store `BINARY` columns defined as branching in each table
See 16.1.2 *Calculating the number of table storage pages*.

γ_i : Number of pages required to store each index
See 16.1.3 *Calculating the number of index storage pages*.

16.1.2 Calculating the number of table storage pages

The procedure used to calculate the number of pages required to store a table depends on whether `FIX` is specified for the table in `CREATE TABLE`. The procedures are explained in (1) and (2). The variables used by the formulas in (1) and (2) are explained in (3), and examples of calculating the number of pages needed to store a table are presented in (7). Estimating the RDAREA size when the rebalancing facility is used is explained in (6).

If a table is row-partitioned, the number of pages for the table in each storage RDAREA must be obtained.

(1) `FIX` not specified

When `FIX` is not specified, the following formula is used to calculate the number of pages needed to store a table.

Formula

Number of table storage pages =

number of pages that store columns other than `BINARY` columns defined as branching
+ number of pages that store `BINARY` columns defined as branching (pages)

- Number of pages that store columns other than `BINARY` columns defined as branching
-

$$\left\lceil \frac{(P + \text{SPN1} + \sum_{i=1}^n PS_i) \times g}{g - \left\lfloor \frac{g \times h}{100} \right\rfloor} \right\rceil$$

- Number of pages that store `BINARY` columns defined as branching
-

SPN2

(a) How to obtain the value of P

Use the following formula to obtain the value of P . The denominator enclosed in parentheses indicates the number of rows stored per page; its minimum and maximum values are 1 and 255, respectively.

Formula

$$P = \left(\frac{a}{\left(\frac{b \times (100 - c)}{100} - 48 - Z \right)^{\#1} \times \frac{\sum_{i=1}^f d_i}{2} \times 2 + 8 + 2 \times f} \right)$$

#1: Determine the values of b and c so that the condition below is satisfied.

The value of b must also satisfy the total column length restrictions described in the common rules for `CREATE TABLE` in the manual *HiRDB Version 9 SQL Reference*.

$$\frac{b \times (100 - c)}{100} - 48 - Z \geq \frac{\sum_{i=1}^f d_i}{2} \times 2 + 8 + 2 \times f$$

(b) How to obtain the value of PS_i

Use the following formula to obtain the value of each PS_i and then obtain their sum, where n indicates the number of columns to which *Table 16-2 Data lengths for the variable-length character string type (except abstract data type and repetition columns)* is applicable.

$$PS_i = a \times \frac{e_i}{(b - 62)}$$

(2) FIX specified

When `FIX` is specified, the following formula is used to calculate the number of pages needed to store a table.

Formula

$$\text{Number of stored pages in table} = \frac{Q \times g}{g - \frac{g \times h}{100}} \quad (\text{Unit: pages})$$

(a) How to obtain the value of Q

Use the following formula to obtain the value of Q , in which the denominator enclosed in parentheses indicates the number of rows stored per page; its minimum and maximum values are 1 and 255, respectively.

$$Q = \left(\frac{a}{\left(\frac{b \times (100 - c)}{100} - 48 - Z \right)^{\#1} \times \frac{\sum_{i=1}^f d_i}{2} \times 2 + 6} \right)$$

#1: Determine the values of b and c so that the condition below is satisfied.

The value of b must also satisfy the total column length restrictions described in the common rules for `CREATE TABLE` in the manual *HiRDB Version 9 SQL Reference*.

$$\frac{b \times (100 - c)}{100} - 48 - Z \geq \frac{\sum_{i=1}^f d_i}{2} \times 2 + 6$$

(3) Variables used in formulas

a: Total number of rows stored in table

b: Page length of user RDAREA (bytes)

c: Percentage of unused area specified with CREATE TABLE (%)

If you omit the percentage of unused area, the system assumes 30%.

d_i: Data length of a column (bytes)

Obtain the values for all columns from *Table 16-1 List of data lengths*.

For a column with an abstract data type, see (4) *How to obtain the data lengths of abstract data type columns*.

For a repetition column, see (5) *How to obtain the data lengths of repetition columns*.

e_i: Average column data length (bytes)

- For columns defined as a provided data type, see *Table 16-2 Data lengths for the variable-length character string type (except abstract data type and repetition columns)* and obtain the values only for the columns with data types listed in this table.
- For columns defined as an abstract data type, see *Table 16-3 Data lengths for the variable-length character string type (abstract data type)* and obtain the values only for the columns with data types listed in this table.
- For repetition columns, see *Table 16-4 Data lengths for the variable-length character string type (repetition columns)* and obtain the values only for the columns with data types listed in this table.

f: Total number of columns defined for table

g: Segment size of RDAREA for storing table (pages)

h: Percentage of free pages in segment specified with CREATE TABLE (%)

If you omit the percentage of free pages in segment, the system assumes 10%. Here, free pages refers to unused pages.

z: System common definition

- If ALL is specified for the operand `pd_dbreuse_remaining_entries`, or if the specification is omitted: 0
- If NONE is specified for the operand `pd_dbreuse_remaining_entries`: 510

SPN1: Number of pages that store columns defined as branching (non-BINARY)

For details about branching conditions, see footnote #5 following *Table 16-1 List of data lengths*.

SPN1 =

f

$\sum_{i=1}^{\uparrow} \text{Value of branch } d_i \div (b - 61) \uparrow \times a \times SF$

i=1

SPN2: Number of pages that store BINARY columns defined as branching

For details about branching conditions, see footnote #5 following *Table 16-1 List of data lengths*.

SPN2 = *SPN2A* + *SPN2B* + *SPN2C*

• Number of branch pages that use INSERT SQL

Calculate for the BINARY columns defined as branching.

SPN2A =

k

$\{ \sum_{i=1}^{\downarrow} L_i \div (b - 59) \downarrow \times a + A \} \times SF$

i=1

• Number of branch pages that use `pdload` or `pdorg`

$L_i > (b - 2,853) \div 255$

SPN2B =

$$k$$

$$\uparrow \{ \sum_{i=1}^k (L_i + 11) \times a \} \div (b - 48) \uparrow \times SF$$

$$i=1$$

$$L_i \leq (b - 2,853) \div 255$$

$$SPN2C =$$

$$\uparrow a \div 255 \uparrow \times SF$$

The formula for A is shown below.

$$A = \left(\begin{array}{c} \uparrow \\ \text{MIN} \left(\begin{array}{c} \downarrow \\ 255, \end{array} \right. \right. \\ \left. \left. \begin{array}{c} \downarrow \sum_{i=1}^k \{ L_i - (b - 59) \times \frac{L_i}{b - 59} + 11 \} \downarrow \end{array} \right) \right. \\ \downarrow \\ \uparrow \end{array} \right)$$

k : Number of columns defined as branching

L_i : Actual data length of each column (bytes)

For a compressed column, use the following formula:

$$\text{Data length after compression} + (\uparrow \text{data length before compression} \div \text{split compression size} \uparrow) \times 8$$

SF : 1.3

However, make this value larger than 1.3 when:

- Large numbers of abstract data type columns will be updated
- Executing, on repetition columns, large quantities of updates that increase element data length or updates that increase the number of elements
- Executing large quantities of updates that significantly increase data length on VARCHAR, NVARCHAR, MVARCCHAR, or BINARY type columns
- Executing large quantities of updates that significantly increase data length on BINARY type columns
- Executing, on columns on which data suppression has been executed on individual columns, large quantities of updates that significantly increase data length
- Executing large quantities of updates from NULL value to non-NULL values with data type other than the above

Table 16–1: List of data lengths

Classification	Data type			Data length (bytes)
Numeric data	INTEGER			4
	SMALLINT			2
	LARGE DECIMAL(m,n) ^{#1}			$\downarrow m/2 \downarrow + 1$ ^{#2}
	FLOAT or DOUBLE PRECISION			8
	SMALLFLT or REAL			4
Character data	CHARACTER(n)			n ^{#3}
	VARCHAR(n) (variable-length character string)	$d \leq 255$	Element of repetition column	$d + 2$
			Other	$d + 1$
		$d \geq 256$		6
	VARCHAR(n) with no-split option specified	$n \leq 255$	Attribute of abstract data type	$d + 3$
			Element of repetition column	$d + 2$
			Other	$d + 1$

Classification	Data type				Data length (bytes)	
		$n \geq 256$	Branching ^{#5}		6	
			Not branching ^{#5}	Attribute of abstract data type	$d + 3$	
				Element of repetition column	$d + 2$	
				Other	$d + 3$	
			National character data	NCHAR(n) or NATIONAL CHARACTER(n)		
NVARCHAR(n)	$d \leq 127$	Element of repetition column		$2 \times d + 2$		
		Other		$2 \times d + 1$		
		$d \geq 128$				6
NVARCHAR(n) with no-split option specified	$n \leq 127$	Attribute of abstract data type		$2 \times d + 3$		
		Element of repetition column		$2 \times d + 2$		
		Other		$2 \times d + 1$		
	$n \geq 128$	Branching ^{#5}		6		
		Not branching ^{#5}		Attribute of abstract data type	$2 \times d + 3$	
				Element of repetition column	$2 \times d + 2$	
				Other	$2 \times d + 3$	
Mixed character string data	MCHAR(n)				$n^{#3}$	
	MVARCHAR(n)	$d \leq 255$		Element of repetition column		$d + 2$
			Other		$d + 1$	
			$d \geq 256$			6
	MVARCHAR(n) with no-split option specified	$n \leq 255$	Attribute of abstract data type		$d + 3$	
			Element of repetition column		$d + 2$	
			Other		$d + 1$	
		$n \geq 256$	Branching ^{#5}		6	
			Not branching ^{#5}	Attribute of abstract data type	$d + 3$	
				Element of repetition column	$d + 2$	
				Other	$d + 3$	
	Date data	DATE				4
	Time data	TIME				3
Date interval data	INTERVAL YEAR TO DAY				5	

Classification	Data type		Data length (bytes)
Time interval data	INTERVAL HOUR TO SECOND		4
Timestamp data	TIMESTAMP(<i>n</i>)		$7 + (n \div 2)$
Large object data	BLOB		9
Binary data	BINARY(<i>n</i>)	$n \leq 255$	$d + 3$
		$n \geq 256$	Branching ^{#5}
			Not branching ^{#5}
Binary data	BINARY(<i>n</i>)	$n \leq 255$	$d + 3$
		$n \geq 256$	Branching ^{#5}
			Not branching ^{#5}
	BINARY(<i>n</i>) with compression specified	Branching ^{#5}	15
		Not branching ^{#5}	$\gamma + 9$

d: Actual data length (in characters)

m, n: Positive integer

γ : Data length after compression (number of characters)

#1: This is a fixed decimal number consisting of a total of *m* digits and *n* decimal places. If *m* is omitted, 15 is assumed.

#2: If the SUPPRESS DECIMAL table option is specified in the table definition, the data length will be $\downarrow k \div 2 \downarrow + 2$, where *k* is the number of significant digits during storage (excluding leading zeros). If the condition shown as follows is satisfied, SUPPRESS DECIMAL should not be used (*a* in the condition is the total data lengths of the columns in the table when SUPPRESS DECIMAL or column data suppression is not used):

$$32,717 < (a + \text{number of columns in table} \times 2 + 8)$$

#3: If column data suppression is specified and data suppression actually occurs, the value of *n* is *n* - *b* + 4. Data suppression occurs only when column data suppression is specified, the column data ends with the blank character, and this blank character is immediately followed by at least four single-byte blank characters. *b* is the number of blank characters following the last character of the column data.

If column data suppression is specified but data suppression does not actually occur, one byte of information is added to each column. However, if the condition shown below is satisfied, column data suppression should not be specified (*a* in the condition is the total data lengths of the columns in the table when SUPPRESS DECIMAL or column data suppression is not used):

$$32,717 < (a + \text{number of columns in table} \times 2 + 8)$$

#4: If column data suppression is specified and data suppression actually occurs, $2 \times n$ becomes $2 \times n - 2 \times b + 5$. Data suppression occurs only when column data suppression is specified, the column data ends with the blank character, and this blank character is immediately followed by at least three double-byte blank characters. *b* is the number of blank characters following the last character of the column data.

However, if the condition shown below is satisfied, column data suppression should not be specified (*a* in the condition is total data lengths of the columns in the table when SUPPRESS DECIMAL or column data suppression is not used):

$$32,717 < (a + \text{number of columns in table} \times 2 + 8)$$

#5: In general, the calculation assumes that there is no branching. Branch only when the condition shown below is satisfied. For a compressed column, use the data length before compression in the calculation.

$$BL > \text{page length} - 50$$

f

$$BL \text{ (bytes)} = \sum d_i + 2 \times f + 6$$

 $i=1$

If this branch condition is satisfied, recalculate the value of BL assuming that each column branches, starting with the column having the smallest column number until the column no longer satisfies the branch condition.

Table 16–2: Data lengths for the variable-length character string type (except abstract data type and repetition columns)

Data type		Data length (bytes)
VARCHAR (n)	$d \geq 256$	$d + 2$
	No-split option specified	0
NVARCHAR (n)	$d \geq 128$	$2 \times d + 2$
	No-split option specified	0
MVARCCHAR (n)	$d \geq 256$	$d + 2$
	No-split option specified	0

d : Actual data length (in characters)

(4) How to obtain the data lengths of abstract data type columns

Use the following formula to obtain data length di of an abstract data type column.

Formula

 h

$$di = \sum_{k=1} ADT_k + 5$$

 $k=1$

h : Inheritance count for the abstract data type

If there is no inheritance, this value is 1.

If you have specified the UNDER operand in the CREATE TYPE statement to inherit another abstract data type, the highest abstract data type is h and the lowest abstract data type is 1.

ADT_k : Data length of the abstract data type (bytes)

Use the following formula to obtain this value:

 m

$$ADT_k = \sum_{i=1} att_i + 10 + 2 \times m$$

 $i=1$

m : Total number of attributes of the abstract data type

att_j : Data length for each attribute of the abstract data type (bytes)

If there is no inheritance, $m = 1$; therefore, calculate the value of ADT_1 .

For the data lengths for each attribute, see Table 16-1 List of data lengths. If the data type satisfies the condition shown in Table 16-3 Data lengths for the variable-length character string type (abstract data type), calculate the data according to Table 16-3 Data lengths for the variable-length character string type (abstract data type).

Assign the value of the corresponding $atte_j$ to the following formula and add branch row storage pages $ADTLS$ to P :

 h

$$ADTLS = \sum_{l=1} \uparrow atte_j \div (b - 62) \uparrow \times a$$

 $l=1$

When attributes are defined as an abstract data type, use the following formula to obtain their data length:

$$att_j(\text{bytes}) = \sum_{k=1}^h ADT_k + 5$$

Table 16–3: Data lengths for the variable-length character string type (abstract data type)

Data type	Condition	Data length att _j (bytes)	Data length of branch section atte _j (bytes)
VARCHAR (n)	$d \geq 256$	8	$d + 2$
	No-split option specified	$d + 3$	0
NVARCHAR (n)	$d \geq 128$	8	$2 \times d + 2$
	No-split option specified	$2 \times d + 3$	0
MVARCHAR (n)	$d \geq 256$	8	$d + 2$
	No-split option specified	$d + 3$	0

d : Actual data length (in characters)

(5) How to obtain the data lengths of repetition columns

Use the following formula to obtain the data length of a repetition column:

Formula

$$d_i = 4 + (el_i + 1) \times en_i$$

el_i : Data length of a repetition column

Obtain the data length from *Table 16-1 List of data lengths*.

For the variable-length character string type, obtain the data length from *Table 16-4 Data lengths for the variable-length character string type (repetition columns)*.

en_i : Average number of elements for a repetition column

Table 16–4: Data lengths for the variable-length character string type (repetition columns)

Data type	Condition	Data length el _i (bytes)	Data length of branch section es _j (bytes)
VARCHAR (n)	$d \geq 256$	5	$d + 2$
	No-split option specified	$d + 2$	0
NVARCHAR (n)	$d \geq 128$	5	$2 \times d + 2$
	No-split option specified	$2 \times d + 3$	0
MVARCHAR (n)	$d \geq 256$	5	$d + 2$
	No-split option specified	$d + 2$	0

d : Actual data length (in characters)

If a repetition column with the variable-length character string type satisfies the value of el_i shown in *Table 16-4 Data lengths for the variable-length character string type (repetition columns)*, add the value obtained from the following formula to P :

$$\uparrow \sum_{m=1}^m \{es_i \times en_i + 14 \times (en_i - 1)\} \div (b - 62) \uparrow \times a$$

 $i=1$

m : Number of repetition columns with the variable-length character string type that satisfy the condition shown in Table 16-4 Data lengths for the variable-length character string type (repetition columns).

es_i : Average value of the actual data length per element

Apply the data length shown in Table 16-2 Data lengths for the variable-length character string type (except abstract data type and repetition columns).

(6) How to estimate the area when the rebalancing facility is used

If there are partitioning tables that use any one of the HASHA through HASHF hash functions, the data is divided into 1,024 hash element values, each of which is stored in a separate segment.

An average of $(1,024 \div \text{number of partitions})$ hash elements of data is stored in each partitioned RDAREA.

Therefore, at a minimum, enough segments must be allocated to each RDAREA to store the number of elements.

When the rebalancing facility is used, the RDAREA size can be estimated as follows:

1. The total number of segments S_n that will be required is estimated from the number of items of data N , row length L , and page length P .

2. Estimate the number of segments S_{sn} required per RDAREA.

$$S_{sn} = \lceil S_n \div S_{rn} \rceil \times S_{rn}$$

$$S_{rn}: \lceil 1,024 \div D_{vn} \rceil$$

D_{vn} : Number of RDAREA partitions

3. Estimate the number of segments S used per RDAREA, making provision for a surplus.

$$S = \lceil (S_{sn} \times K) \div S_{rn} \rceil \times S_{rn}$$

K : Coefficient (Example: 20% surplus, 1.2)

(7) Examples of calculating the number of table storage pages

(a) Example

Obtain the number of table storage pages for the following STOCK table:

PCODE	PNAME	STANDARD	PRICE	QUANTITY	COST
20180	CLEANER	C20	20000	26	15000
20190	CLEANER	C77	28000	105	23000
20130	REFRIGERATOR	P10	30000	70	25000
20220	TV	K18	35000	12	30000
20200	CLEANER	C89	35000	30	30000
20140	REFRIGERATOR	P23	35000	60	30000
20280	AMPLIFIER	L10	38000	200	33000
20150	REFRIGERATOR	P32	48000	50	43000
20290	AMPLIFIER	L50	49800	260	45000
20230	TV	K20	50000	15	45000
20160	REFRIGERATOR	P35	55800	120	50000

Conditions:

1. Total number of rows stored in the table: 10,000

2. Page length of user RDAREA: 8,192 bytes
3. Percentage of unused space specified with CREATE TABLE: 30%
4. Number of columns: 6
5. Segment size for storing table: 100 pages
6. Percentage of free pages in a segment specified in CREATE TABLE: 40%
7. Columns' data types:

Column	Data type
PCODE	CHARACTER (5)
PNAME	CHARACTER (4)
STANDARD	CHARACTER (3)
PRICE	INTEGER
QUANTITY	INTEGER
COST	INTEGER

8. The operand `pd_dbreuse_remaining_entries` is not specified in the system common definition.

FIX not specified

1. Calculation of row length
 $5(\text{PCODE}) + (2 \times 4)(\text{PNAME}) + 3(\text{STANDARD}) + 4(\text{PRICE}) + 4(\text{QUANTITY}) + 4(\text{COST}) = 28 \text{ bytes}$
2. Calculation of P

$$\begin{array}{c}
 \uparrow \quad \quad \quad 10000 \quad \quad \quad \uparrow \\
 \downarrow \quad \quad \quad \frac{8192 \times (100-30)}{100} \quad \quad \quad \downarrow -48 \\
 \downarrow \quad \quad \quad 14 \times 2 + 8 + 2 \times 6 \quad \quad \quad \downarrow \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad = 85
 \end{array}$$

3. Calculation of the number of table storage pages

$$\begin{array}{c}
 \uparrow \quad \quad \quad 85 \times 100 \quad \quad \quad \uparrow \\
 \downarrow \quad \quad \quad \frac{100 \times 40}{100} \quad \quad \quad \downarrow \\
 \downarrow \quad \quad \quad 100 - \quad \quad \quad \downarrow \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad = 142 \text{ pages}
 \end{array}$$

FIX specified

1. Calculation of row length
 $5(\text{PCODE}) + (2 \times 4)(\text{PNAME}) + 3(\text{STANDARD}) + 4(\text{PRICE}) + 4(\text{QUANTITY}) + 4(\text{COST}) = 28 \text{ bytes}$
2. Calculation of Q

$$\begin{array}{c}
 \uparrow \quad \quad \quad 10000 \quad \quad \quad \uparrow \\
 \downarrow \quad \quad \quad \frac{8192 \times (100-30)}{100} \quad \quad \quad \downarrow -48 \\
 \downarrow \quad \quad \quad 14 \times 2 + 6 \quad \quad \quad \downarrow \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad = 60
 \end{array}$$

3. Calculation of the number of table storage pages

$$\begin{array}{c} \uparrow \\ \hline 60 \times 100 \\ \hline 100 - \frac{100 \times 40}{100} \\ \hline \downarrow \end{array} = \underline{100\text{pages}}$$

16.1.3 Calculating the number of index storage pages

The procedure used to calculate the number of pages required to store an index is explained in (1) as follows. The variables used in the formulas are explained in (2), and examples of calculating the number of pages needed to store an index are presented in (3).

If cluster key is specified with CREATE TABLE, the number of cluster key storage pages should be obtained in the same manner as for the number of index storage pages.

If an index is row-partitioned, the number of pages for the index in each storage RDAREA must be obtained.

Note

When an index page split occurs, the storage ratio of the keys in the index is 50:50, and the index is divided into two indexes. Therefore, if there are many additions or updates to an index, make the estimate for the maximum number of index storage pages double the calculated value. Also, even if there are inserts from the UAP, the index page split of the leaf page that stores the largest key considers the value of the PCTFREE operand.

One method to reduce the frequency of index page splits is unbalanced index splits. For details about index page splits and unbalanced index splits, see the *HiRDB Version 9 System Operation Guide*.

(1) Calculation procedure

The following formula is used to calculate the number of pages needed to store an index.

Formula

$$\begin{array}{l} n \\ \text{Number of index storage pages} = \sum_{i=1} P_i + P_d \end{array}$$

The recursive formula shown in Formula 1 as follows is used to obtain P_i . $P_i + 1$ must be calculated until $P_n = 1$, then the sum of the results must be obtained.

The value of P_d must be obtained if the number of duplicated key values exceeds 200. The formula for obtaining the value of P_d is shown in Formula 2 as follows:

Formula 1

$$\begin{aligned}
 P_1 = & \left(\begin{array}{c} \uparrow \\ \text{MA} \times \left(1 + \frac{c - h}{18 + g + 4 \times d} \left(\frac{a \times (100 - b)}{100} - g - 68 \right) \right) \\ \downarrow \end{array} \right) \\
 + & \left(\begin{array}{c} \uparrow \\ \text{MA} \times \left(1 + \frac{e}{18 + g} \left(\frac{a \times (100 - b)}{100} - g - 68 \right) \right) \\ \downarrow \end{array} \right) \\
 + & \left(\begin{array}{c} \uparrow \\ \text{MA} \times \left(1 + \frac{h}{14 + g} \left(\frac{a \times (100 - b)}{100} - g - 68 \right) \right) \\ \downarrow \end{array} \right) \\
 P_{i+1} = & \left(\begin{array}{c} \uparrow \\ \text{MA} \times \left(1 + \frac{P_i}{14 + g} \left(\frac{a \times (100 - b)}{100} - g - 68 \right) + 1 \right) \\ \downarrow \end{array} \right)
 \end{aligned}$$

Formula 2

$$P_d = \left\{ \left(\begin{array}{c} \uparrow \\ \frac{f}{4} \left(\frac{a \times 95}{100} - 70 \right) \\ \downarrow \end{array} \right) + 1 \right\} \times e$$

Number of duplicated elements per row when the index contains repetition columns

If the index contains repetition columns, the number of duplicated elements per row must not exceed the following value:

$$\text{Number of duplicated elements} = \left\lfloor \left(\left\lfloor a \times 0.95 \right\rfloor - 82 \right) \div 4 \right\rfloor - 1$$

(2) Variables used in formulas

a: Page length of user RDAREA (bytes)

b: Percentage of unused space specified with CREATE TABLE^{#1}(%)

c: Number of keys with up to 200 duplicated key values^{#2, #3, #4}

d: Average number of duplicates for keys with up to 200 duplicated key values^{#3, #5}

e: Number of keys with more than 200 duplicated key values^{#3, #4}

f: Average number of duplicates for keys with more than 200 duplicated key values^{#3, #5}

g: DB storage key length^{#6} (bytes)

h: One of the following:

- For a unique index: Number of keys not containing a null value
For a multicolumn index, the total number of keys not containing a null value in its component columns.
- For other than unique index: 0

#1

If no percentage of unused space is specified, 30% should be used in the calculation. If a cluster key is specified, the percentage of unused space specified with `CREATE TABLE` should be used.

#2

Non-duplicated keys in unique indexes must be included.

#3

Calculate so that the value of $c \times d + e \times f$ is larger than the total number of index keys.

#4

Duplicate keys in unique indexes must be included (keys that are duplicate due to the fact that the key contains a null value).

#5

Round up fractions to an integer value.

#6

See *Table 16-5 List of index key lengths*. The length of the DB storage key can be obtained from the following formula:

- For single column indexes and fixed-size multicolumn indexes
 $\lceil \text{key length} \div 4 \rceil \times 4$
- For variable-size multicolumn indexes with a key length of 255 bytes or less
 $\lceil (\text{key length} + 1) \div 4 \rceil \times 4$
- For variable-size multicolumn indexes with a key length of 256 bytes or more
 $\lceil (\text{key length} + 2) \div 4 \rceil \times 4$

The key length of a multicolumn index is the sum of the key lengths of its component columns based on the key lengths shown in *Table 16-5 List of index key lengths*.

Table 16–5: List of index key lengths

Classifi- cation	Data type	Data length (bytes)					
		Key length less than 256			Key length 256 or greater		
		Single- column index	Multicolumn index		Single- column index	Multicolumn index	
			Fixed length ^{#1}	Variable length ^{#2}		Fixed length ^{#1}	Variable length ^{#2}
Numeric data	INTEGER	4	5	6	N	5	7
	SMALLINT	2	3	4	N	3	5
	LARGE DECIMAL(<i>m,n</i>) ^{#3}	$\lfloor m \div 2 \rfloor$ + 1	$\lfloor m \div 2 \rfloor$ + 2	$\lfloor m \div 2 \rfloor$ + 3	N	$\lfloor m \div 2 \rfloor$ + 2	$\lfloor m \div 2 \rfloor$ + 4
	FLOAT or DOUBLE PRECISION	8	E	E	--	E	E
	SMALLFLT or REAL	4	E	E	--	E	E
Character data	CHARACTER(<i>n</i>)	<i>N</i>	<i>n</i> + 1	<i>n</i> + 2	<i>n</i>	<i>n</i> + 1	<i>n</i> + 3
	VARCHAR(<i>n</i>)	<i>a</i> + 1	N	<i>a</i> + 2	<i>a</i> + 2	N	<i>a</i> + 3

Classification	Data type	Data length (bytes)					
		Key length less than 256			Key length 256 or greater		
		Single-column index	Multicolumn index		Single-column index	Multicolumn index	
			Fixed length#1	Variable length#2		Fixed length#1	Variable length#2
National character data	NCHAR(<i>n</i>) or NATIONAL CHARACTER(<i>n</i>)	$2 \times n$	$2 \times n + 1$	$2 \times n + 2$	$2 \times n$	$2 \times n + 1$	$2 \times n + 3$
	NVARCHAR(<i>n</i>)	$2 \times b + 1$	N	$2 \times b + 2$	$2 \times b + 2$	N	$2 \times b + 3$
Mixed character data	MCHAR(<i>n</i>)	<i>N</i>	<i>n</i> + 1	<i>n</i> + 2	<i>n</i>	<i>n</i> + 1	<i>n</i> + 3
	MVARCHAR(<i>n</i>)	<i>a</i> + 1	N	<i>a</i> + 2	<i>a</i> + 2	N	<i>a</i> + 3
Date data	DATE	4	5	6	--	5	7
Time data	TIME	3	4	5	--	4	6
Date interval data	INTERVAL YEAR TO DAY	5	6	7	--	6	8
Time interval data	INTERVAL HOUR TO SECOND	4	5	6	--	5	7
Timestamp data	TIMESTAMP(<i>p</i>)	$7 + (p \div 2)$	$8 + (p \div 2)$	$9 + (p \div 2)$	--	$8 + (p \div 2)$	$10 + (p \div 2)$

a: Actual data length

b: Actual number of characters

m, *n*, *p*: Positive integer

E: Error occurs during index definition

--: Not applicable

Note

Begin calculation with a key length less than 255 bytes. If it turns out that the key length is 256 bytes or greater, recalculate at a key length of 256 bytes or greater.

#1: Key length of an index that contains only fixed-length component columns.

#2: Key length of an index that contains variable-length component columns.

#3: This is a fixed decimal number consisting of a total of *m* digits and *n* decimal places. If *m* is omitted, 15 is assumed.

Reference note

Non-unique indexes have areas that store index data multiple times in their index data storage areas, so they are that much larger. Unique indexes, on the other hand, have no areas that store duplicated instances. For this reason, unique indexes are smaller than non-unique indexes.

(3) Examples of calculating the number of index storage pages

(a) Example 1

Obtain the number of index storage pages for a unique index (no duplicated key values) of the PCODE column for the following STOCK table:

PNO	PNAME	STANDARD	PRICE	QUANTITY	COST
20180	CLEANER	C20	20000	26	15000

PNO	PNAME	STANDARD	PRICE	QUANTITY	COST
20190	CLEANER	C77	28000	105	23000
20130	REFRIGERATOR	P10	30000	70	25000
20220	TV	K18	35000	12	30000
20200	CLEANER	C89	35000	30	30000
20140	REFRIGERATOR	P23	35000	60	30000
20280	AMPLIFIER	L10	38000	200	33000
20150	REFRIGERATOR	P32	48000	50	43000
20290	AMPLIFIER	L50	49800	260	45000
20230	TV	K20	50000	15	45000
20160	REFRIGERATOR	P35	55800	120	50000

Conditions:

1. Total number of index keys: 10,000
2. Page length of user RDAREA: 8,192 bytes
3. Percentage of unused space specified with CREATE TABLE: 30%
4. Data type of index: CHARACTER
5. Index key length: 5 bytes
6. Number of key duplicates: 1

Formula

$$\text{DB storage key length}(\underline{g}) = \uparrow 5 \div 4 \uparrow \times 4 = 8$$

$$\begin{aligned}
 P_1 = & \begin{array}{c} \uparrow \\ \text{10000-10000} \\ \text{MAX (1 , } \downarrow \frac{8192 \times (100-30)}{100} \downarrow -8-68 \downarrow \text{)} \\ \text{18+8+4} \times 1 \\ \downarrow \end{array} \\
 + & \begin{array}{c} \uparrow \\ 0 \\ \text{MAX (1 , } \downarrow \frac{8192 \times (100-30)}{100} \downarrow -8-68 \downarrow \text{)} \\ \text{18+8} \\ \downarrow \end{array} \\
 + & \begin{array}{c} \uparrow \\ 10000 \\ \text{MAX (1 , } \downarrow \frac{8192 \times (100-30)}{100} \downarrow -8-68 \downarrow \text{)} \\ \text{14+8} \\ \downarrow \end{array} \\
 = & 0 + 0 + 39 = 39 \\
 P_2 = & \begin{array}{c} \uparrow \\ 39 \\ \text{MAX (1 , } \downarrow \frac{8192 \times (100-30)}{100} \downarrow -8-68 \downarrow \text{ + 1)} \\ \text{14+8} \\ \downarrow \end{array} = 1
 \end{aligned}$$

$P_d=0$ (because the number of duplicated key values does not exceed 200)

Number of pages needed to store an Index = $39+1+0=40$ pages

(b) Example 2

Obtain the number of index storage pages for the STOCK table shown in Example 1 when the PNAME column is used as the index (with duplicated key values).

Conditions:

1. Total number of index keys: 10,000
2. Page length of user RDAREA: 8,192 bytes
3. Percentage of unused space specified with CREATE TABLE: 30%
4. Data type of index: NCHAR
5. Index key length: 4 characters (kanji characters)
6. Number of keys with more than 200 duplicated key values: 1
(Average number of duplicates: 250)
7. Number of keys with up to 200 duplicated key values: $(10,000 - 250)/5 = 1,950$ (Average number of duplicates: 5)

Formula

$$\text{DB storage key length}(g) = \lceil (2 \times 4) \div 4 \rceil \times 4 = 8$$

$$\begin{aligned}
 P_1 &= \left\lceil \frac{1950 - 0}{\text{MAX} \left(1, \left\lfloor \frac{8192 \times (100 - 30)}{100} \right\rfloor - 8 - 68 \right)} \right\rceil \\
 &\quad \frac{18 + 8 + 4 \times 5}{1} \\
 + &\left\lceil \frac{1}{\text{MAX} \left(1, \left\lfloor \frac{8192 \times (100 - 30)}{100} \right\rfloor - 8 - 68 \right)} \right\rceil \\
 &\quad \frac{18 + 8}{0} \\
 + &\left\lceil \frac{0}{\text{MAX} \left(1, \left\lfloor \frac{8192 \times (100 - 30)}{100} \right\rfloor - 8 - 68 \right)} \right\rceil \\
 &\quad \frac{14 + 8}{16 + 1 + 0 = 17} \\
 P_2 &= \left\lceil \frac{17}{\text{MAX} \left(1, \left\lfloor \frac{8192 \times (100 - 30)}{100} \right\rfloor - 8 - 68 \right)} \right\rceil + 1 = 1 \\
 P_d &= \left\{ \left\lceil \frac{250}{\left\lfloor \frac{8192 \times 95}{100} \right\rfloor - 70} \right\rceil + 1 \right\} \times 1 = 2
 \end{aligned}$$

Number of pages needed to store an Index = $17 + 1 + 2 = 20$ pages

Obtain the number of index storage pages for the following MEMBERSHIP_TABLE using the SEX and YEAR_JOINED columns as a multicolumn index:

MNO	NAME	AGE	SEX	YEAR_JOINED
0001	Lisa Roberts	18	F	1983
0002	John Anderson	25	M	1967
0003	Jane Wood	24	F	1987
0004	Mark Wood	25	M	1964
...
...

16. Determining RDAREA Size

MNO	NAME	AGE	SEX	YEAR_JOINED
...
1000	Joe Young	30	M	1995

Conditions:

1. Total number of index keys: 10,000
2. Page length of user RDAREA: 8,192 bytes
3. Percentage of unused space specified with CREATE INDEX: 30%
4. Number of members joined in 1964: 1,000
5. Number of members joined in any other year: 200 or fewer
6. Period covered: 31 years from 1965 to 1995
7. The same numbers of male and female members are assumed to have joined each year.
8. Data types of columns:

Column	Data type
MNO	CHARACTER(5)
NAME	NCHAR(4)
AGE	INTEGER
SEX	CHARACTER(4)
YEAR_JOINED	INTEGER

Formula:

1. The number of keys (c) for members who joined within 31 years after 1965 (no more than 200 per year including both male and female members): $c = 31 \times 2 = 62$
2. The average number of duplicates (d) is: $d = (10,000 - 1,000) \div 62 = 146$.
3. Number of keys (e) for members who joined in 1964 (1,000 members including both male and female members): $e = 2$
4. Average number of duplicates (f): $f = 1,000 \div 2 = 500$
5. DB storage key length (g) of the SEX and YEAR_JOINED columns:
 $g = \uparrow(4 + 1 + 5)/4 \uparrow \times 4 = 12$

$$\begin{aligned}
 P_1 &= \begin{array}{c} \uparrow \\ \hline 62-0 \\ \hline \text{MA} \times (1, \downarrow \downarrow \frac{8192 \times (100-30)}{100} \downarrow -12-68 \downarrow) \\ \hline 18+12+4 \times 146 \\ \hline \downarrow \end{array} \\
 &+ \begin{array}{c} \uparrow \\ \hline 2 \\ \hline \text{MA} \times (1, \downarrow \downarrow \frac{8192 \times (100-30)}{100} \downarrow -12-68 \downarrow) \\ \hline 18+12 \\ \hline \downarrow \end{array} \\
 &+ \begin{array}{c} \uparrow \\ \hline 0 \\ \hline \text{MA} \times (1, \downarrow \downarrow \frac{8192 \times (100-30)}{100} \downarrow -12-68 \downarrow) \\ \hline 14+12 \\ \hline \downarrow \end{array} \\
 &= 7 + 1 + 0 = 8 \\
 P_2 &= \begin{array}{c} \uparrow \\ \hline 8 \\ \hline \text{MA} \times (1, \downarrow \downarrow \frac{8192 \times (100-30)}{100} \downarrow -12-68 \downarrow + 1) \\ \hline 14+12 \\ \hline \downarrow \end{array} = 1 \\
 P_d &= \left\{ \begin{array}{c} \uparrow \\ \hline 500 \\ \hline \downarrow \frac{8192 \times 95}{100} \downarrow -70 \\ \hline 4 \\ \hline \downarrow \end{array} \right\} + 1 \times 2 = 4
 \end{aligned}$$

Number of pages needed to store an Index = $8+1+4=13$ pages

16.2 Determining the size of a data dictionary RDAREA

You can use the `create rdarea` statement of the database structure modification utility (`pdmod`) to create the following two types of data dictionary RDAREA:

- Normal data dictionary RDAREA
Specify `datadictionary` or `datadictionary of routines` in the `create rdarea` statement.
- Data dictionary RDAREA for storing database state analyzed tables and database management tables
Specify `datadictionary of dbmanagement` in the `create rdarea` statement.

You must determine the size of each RDAREA of either of these types.

16.2.1 Determining the size of a normal data dictionary RDAREA

(1) Formula for calculating the size of a data dictionary RDAREA

Use the following formula to determine the size of a data dictionary RDAREA that will be created by the `create rdarea` statement with `datadictionary` or `datadictionary of routines` specified:

Formula

Size of data dictionary RDAREA (bytes)

$$= a \times b \times 1.3 + c \times 125 + 1,600,000$$

■ If you are defining an extended system-defined scalar function, add:
+ 1,933,312

a: Page length of the data dictionary RDAREA^{#1}

b: Total number of pages in the data dictionary RDAREA^{#2}

c: Segment size of the data dictionary RDAREA^{#3}

#1: This is the page length specified in the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`).

#2: This is the number of table storage pages + number of index storage pages (see (2) and (3) as follows).

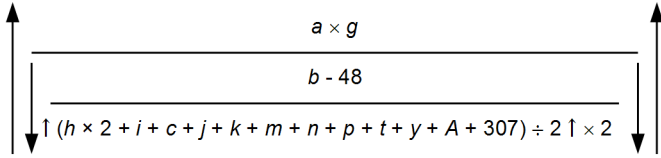
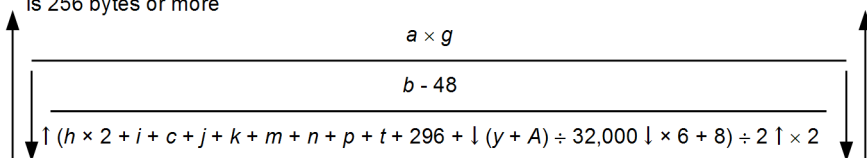
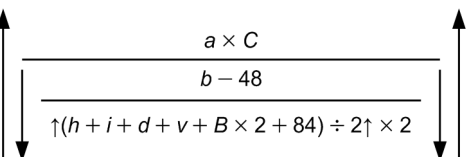
#3: This is the segment size specified in the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`).

(2) Calculating the number of table storage pages

The number of pages required to store tables is the sum of the values obtained from Formulas 1 through 23.

(a) Formula 1

Dictionary table name	Formula
SQL_TABLES	$\uparrow \frac{(h \times 2 + i \times 2 + e + d + f + w + D + 657) \div 2 \uparrow \times 2}{b - 48} \uparrow a$

Dictionary table name	Formula
SQL_COLUMNS	<ul style="list-style-type: none"> When no <code>DEFAULT</code> clause is specified, or when the data length of the default value specified in the <code>DEFAULT</code> clause is 255 bytes or less  When the data length of the default value specified in the <code>DEFAULT</code> clause is 256 bytes or more  $+ a \times g \times \lceil (w + y) \div (b - 61) \rceil$
SQL_DIV_TABLE	

a: Total number of tables

b: Page length of data dictionary RDAREA (bytes)

c: Average length of column names (bytes)

d: Average length of the names of the RDAREAs for storing the tables

e: Average length of the comments in the tables (bytes)

f: Average length of the names of the columns for which table partitioning conditions are specified (bytes)

g: Average number of table columns

h: Average length of the authorization identifiers (bytes)

i: Average length of the table identifiers (bytes)

j: Average length of the comments on the columns (bytes)

k: Average length of the authorization identifiers of the base tables used to create view tables (bytes)

m: Average length of the column names in the base tables used to create view tables (bytes)

n: Average length of the table identifiers of the base tables used to create view tables (bytes)

p: Average length of the names of user-defined data types (bytes)

q: Average number of table row partitioning conditions

t: Average length of `PLUGIN` clauses (bytes)

v: Average length of partitioning key (bytes)

w: Average value for the name of the insert history maintenance column (bytes)

y: Average base length of constants specified in the `DEFAULT` clause (bytes)

For details about how to calculate the base length, see the section on data codes and data lengths set in the SQL descriptor area in the *HiRDB Version 9 UAP Development Guide*.

A: Average length of constants specified in the `DEFAULT` clause (bytes)

If the specified constant is a literal, this is the apparent length of the literal. If there is a possibility that the length of the constant will increase, keep the post-update length in mind when making the calculation. Character literals

are indicated by notations, such as N for National character column literals, M for mixed character column literals, X for hexadecimal character literals, and single quotation marks ('). Constant specification is in bytes.

Example:

'HiRDB': 7 bytes

N' H i R D B ': 13 bytes

X' 4869524442 ': 13 bytes

CURRENT_TIME: 12 bytes

100: 3 bytes

B: Average length of table row partitioning conditions (bytes)

C: Specified number of table storage RDAREAs

D: Average length of names of database areas that store tables (bytes)

(b) Formula 2

Dictionary table name	Formula
SQL_INDEXES	$\begin{array}{c} c \\ a - 48 \\ \uparrow (f \times 2 + b + d + 2 \times k + j + m + n + B + 152) \div 2 \uparrow \times 2 \end{array}$
SQL_INDEX_COLINF	$\begin{array}{c} c \times k \\ a - 48 \\ \uparrow (f + b + d + g + 35) \div 2 \uparrow \times 2 \end{array}$
SQL_DIV_INDEX	$\begin{array}{c} c \times i \\ a - 48 \\ \uparrow (f + b + d + j + 44) \div 2 \uparrow \times 2 \end{array}$
SQL_EXCEPT	$\begin{array}{c} c \times e \\ a - 48 \\ \uparrow \{f + b + d + (k \times 7 - 2) + 33\} \div 2 \uparrow \times 2 \end{array}$
SQL_INDEX_DATATYPE	$\begin{array}{c} t \\ a - 48 \\ \uparrow (f \times 2 + b + d + g + q + r + 43) \div 2 \uparrow \times 2 \end{array}$

Dictionary table name	Formula
SQL_INDEX_FUNCTION	$t \times s$ $a - 48$ $\uparrow (f \times 2 + b + d + g + p + 38) \div 2 \uparrow \times 2$
SQL_INDEX_XMLINF	$y \times v$ $a - 48$ $\uparrow (f + b + d + u + 12 \times w + 81) \div 2 \uparrow \times 2$ $+ y \times v \times \frac{A}{a - 48}$

a: Page length of the data dictionary RDAREA (bytes)

b: Average length of table identifiers (bytes)

c: Total number of indexes

d: Average length of index identifiers (bytes)

e: Average number of index exception key values per index

f: Average length of index identifiers (bytes)

g: Average length of column names (bytes)

i: Average number of table row partitioning conditions

j: Average length of the names of the RDAREAs for storing indexes (bytes)

k: Average number of columns constituting indexes

m: Average length of the index type names (bytes)

n: Average length of `PLUGIN` clause specifications (bytes)

p: Average length of plug-in index application function names (bytes)

q: Average length of abstract data type names (bytes)

r: Average length of attribute names (bytes)

s: Number of application functions per plug-in index

t: Total number of plug-in indexes

u: Average length of substructure path (bytes)

v: Average number of substructure paths constituting a substructure index

w: Average number of binary data items (analysis tree for substructure paths) that have a data length of at least 256 bytes and are branching

For details about the branching conditions of the number of storage pages for binary data, see *Table 16-1 List of data lengths*.

y: Total number of substructure indexes

A: Substructure path analysis tree length (bytes)

The value found using the following formula:

$$S \times 120 + P + L + S \times 4 + 32$$

L: Total length of character string expression of local name that specifies the modifier name of a step expression (bytes)[#]

P: Total length of character string expression of XML name space URI associated with prefix (bytes)[#]

When the prefix is omitted, the default associated XML namespace URI

S: Number of step expression specifications

#: Round up to a power of four.

B: Average length of names of database areas that store indexes (bytes)

(c) Formula 3

Dictionary table name	Formula
SQL_TABLE_PRIVILEGES	
SQL_RDAREA_PRIVILEGES	
SQL_VIEW_TABLE_USAGE	
SQL_VIEWS	
SQL_VIEW_DEF [#]	

a: Page length of the data dictionary RDAREA (bytes)

b: Average length of authorization identifiers (bytes)

c: Average length of table identifiers (bytes)

d: Number of access privileges defined

- If you have granted the privilege to *n* users per table, the value is the number of tables for which the privilege is granted times *n*.
- If you have granted the privilege to PUBLIC, the value is one (user).
- If you have granted the privilege to a group, one group ID is treated as one (user).

e: Total number of RDAREAs

f: Average length of the names of RDAREAs for storing tables (bytes)

g: Average length of SQL statements for defining view tables (bytes)

h: Total number of view definitions

j : Average length of view analysis information (bytes)

For details about the length of view analysis information per table viewed, see *Formulas for determining size of view analysis information buffers (pd_view_def_cache_size)* in the manual *HiRDB Version 9 System Definition*.

#: These tables are used by the system.

(d) Formula 4

Data dictionary table name	Formula
SQL_REFERENTIAL_ CONSTRAINTS	$\uparrow (b+3 \times c+2 \times d+E+F+G+H+186) \div 2 \uparrow \times 2$

$$E: \{e \times h + 2 \times h + (h - 1)\} + 1$$

$$F: \{e \times i + 2 \times i + (i - 1)\} + 1$$

$$G: \{2 \times h + (h - 1)\} + 1$$

$$H: \{2 \times i + (i - 1)\} + 1$$

a : Page length of the data dictionary RDAREA (bytes)

b : Average length of the constraint names (bytes)

c : Average length of the authorization identifiers (bytes)

d : Average length of the table identifiers (bytes)

e : Average length of the column names for which a foreign key has been defined (bytes)

f : Average length of the column names for which a primary key has been defined (bytes)

h : Average number of columns constituting the foreign keys

i : Average number of columns constituting the primary keys

(e) Formula 5

Dictionary table name	Formula
SQL_PHYSICAL_FILES	$\uparrow (c+g+36) \div 2 \uparrow \times 2$
SQL_RDAREAS	$\uparrow (c \times 2 + 101) \div 2 \uparrow \times 2$
SQL_USERS	$\uparrow (d + h + 75) \div 2 \uparrow \times 2$

16. Determining RDAREA Size

- a*: Page length of the data dictionary RDAREA (bytes)
- b*: Total number of RDAREAs
- c*: Average length of RDAREA names (bytes)
- d*: Average length of schema authorization identifiers (bytes)
- e*: Total number of schemas
- f*: Total number of HiRDB files constituting all RDAREAs
- g*: Average length of the names of HiRDB files constituting all RDAREAs (bytes)
- h*: Average length of a password (bytes)

(f) Formula 6

Dictionary table name	Formula
SQL_DIV_TABLE_REGULARSIZE [#]	

- a*: Total number of row-partitioned tables
- b*: Page length of the data dictionary RDAREA (bytes)
- c*: Average length of the names of the RDAREAs for storing tables (bytes)
- d*: Average number of table row partitioning conditions
- e*: Average length of authorization identifiers (bytes)
- f*: Average length of table identifiers (bytes)
- g*: Average length of the condition values for character-string columns for which table partitioning conditions are specified (bytes)
- h*: Average length of the condition values for numeric columns for which table partitioning conditions are specified (bytes)
- [#]: These tables are used by the system.

(g) Formula 7

Dictionary table name	Formula
SQL_TABLE_STATISTICS ^{#1}	
SQL_COLUMN_STATISTICS ^{#1}	

Dictionary table name	Formula
SQL_INDEX_STATISTICS ^{#1}	$\begin{array}{c} \uparrow \\ \overline{\overline{\overline{g}} \\ \overline{b-48} \\ \uparrow (e+f+h+76) \div 2 \uparrow \times 2}} \\ \downarrow \end{array}$

a: Total number of tables for which optimizing information is to be collected

b: Page length of the data dictionary RDAREA (bytes)

c: Average length of column names (bytes)

e: Average length of authorization identifiers (bytes)

f: Average length of table identifiers (bytes)

g: Total number of indexes defined for tables for which optimizing information is collected

h: Average length of index identifiers (bytes)

i: Number of index key column values for tables for which optimizing information is collected^{#2}

#1: These tables are used by the system.

#2: If the number of key column values < 100, then *i* = number of key column values.

If the number of key column values ≥ 100, then *i* = 100.

(h) Formula 8

Dictionary table name	Formula
SQL_DIV_COLUMN	$\begin{array}{c} \uparrow \\ \overline{\overline{\overline{a}} \\ \overline{b-48} \\ \uparrow (e+f+c+d \times 2 + h + 68) \div 2 \uparrow \times 2}} \\ \downarrow \end{array}$

a: Total number of BLOB column storage RDAREAs

b: Page length of the data dictionary RDAREA (bytes)

c: Average length of column names (bytes)

d: Average length of the names of the RDAREAs for storing tables (bytes)

e: Average length of authorization identifiers (bytes)

f: Average length of table identifiers (bytes)

h: Average length of component names (bytes)

(i) Formula 9

Dictionary table name	Formula
SQL_ROUTINES	$\begin{array}{c} \uparrow \\ \overline{\overline{\overline{a}} \\ \overline{b-48} \\ \uparrow (c + d \times 4 + e + k + m + n + p + q + r + 341) \div 2 \uparrow \times 2}} + \frac{174}{\overline{\overline{b-48} \\ 310}} \\ \downarrow \end{array}$

Dictionary table name	Formula
SQL_ROUTINE_RESOURCES	
SQL_ROUTINE_PARAMS	

a: Total number of routines

b: Page length of the data dictionary RDAREA (bytes)

c: Average length of routine names (bytes)

d: Average length of authorization identifiers (bytes)

e: Average length of specified names^{#1} (bytes)

f: Average length of parameter names (bytes)

g: Average length of authorization identifiers of resource^{#2} owners (bytes)

h: Average length of resource^{#2} names (bytes)

i: Average number of resources^{#2} per routine

j: Average number of parameters per routine

k: Average length of abstract data type names (bytes)

m: Average length of user-defined data type names (bytes)

n: Average length of external routine names (bytes)

p: Average length of Java class name (bytes)

q: Average length of Java archive name (bytes)

r: Average length of data type name for Java return value (bytes)

s: Average length of name with Java parameter data type (bytes)

t: Average length of column names specified by old or new value correlation names (bytes)

#1: Indicates *authorization-identifier.routine-identifier*.

#2: Resources include tables and indexes.

(j) Formula 10

Dictionary table name	Formula
SQL_DATATYPES	

Dictionary table name	Formula
SQL_DATATYPE_DESCRIPTOR	$\lceil \{ c + f + g + j \times (k + m) + 152 \} \div 2 \rceil \times 2$

a: Total number of user-defined data types

b: Page length of the data dictionary RDAREA (bytes)

c: Average length of attribute or field names (bytes)

d: Average length of user-defined data type comments (bytes)

e: Average number of attributes per data type

f: Average length of authorization identifiers (bytes)

g: Average length of data type identifiers (bytes)

h: Average length of authorization identifiers for supertype abstract data types (bytes)

i: Average length of data identifiers for supertype abstract data types (bytes)

j: Number of attributes defined for user-defined data types

k: Average length of authorization identifiers of abstract data types for attributes defined for user-defined data types (bytes)

m: Average length of data identifiers of abstract data types for attributes defined for user data types (bytes)

(k) Formula 11

Dictionary table name	Formula
SQL_PLUGINS	$\lceil (m + d \times 2 + n + o + p + 63) \div 2 \rceil \times 2$
SQL_PLUGIN_ROUTINES	$\lceil (c + d + e + m + q + r + s + 47) \div 2 \rceil \times 2$
SQL_PLUGIN_ROUTINE_PARAMS	$\lceil (d \times 2 + e + m + f + n + t + u + v + 140) \div 2 \rceil \times 2$

a: Average number of plug-in routines per plug-in

b: Page length of the data dictionary RDAREA (bytes)

c: Average length of routine names (bytes)

d: Average length of authorization identifiers (bytes)

- e*: Average length of specified names[#] (bytes)
f: Average length of parameter names per plug-in (bytes)
i: Average number of resources per routine
k: Total number of plug-ins
m: Average length of plug-in names (bytes)
n: Average length of abstract data type/index type names (bytes)
o: Average length of plug-in library path names (bytes)
p: Average length of plug-in comments (bytes)
q: Average length of timing indicators (bytes)
r: Average length of operation qualifiers (bytes)
s: Average length of operation qualifier code (bytes)
t: Average length of parameter qualifier information (bytes)
u: Average length of bind operation name (bytes)
v: Average length of parameter modifier information code (bytes)
[#]: This means the *authorization-identifier.routine-identifier*.

(I) Formula 12

Dictionary table name	Formula
SQL_INDEX_TYPES	
SQL_INDEX_TYPE_FUNCTION	

- a*: Total number of index types
b: Page length of the data dictionary RDAREA (bytes)
c: Average length of authorization identifiers (bytes)
d: Average length of index type identifiers (bytes)
e: Average length of abstract data type names (bytes)
f: Number of application functions per index type

(m) Formula 13

Dictionary table name	Formula
SQL_INDEX_RESOURCES	
SQL_TYPE_RESOURCES	
SQL_TABLE_RESOURCES	

- a*: Total number of plug-in indexes
b: Page length of the data dictionary RDAREA (bytes)
c: Average length of authorization identifiers (bytes)
d: Average length of index type identifiers (bytes)
e: Average length of abstract data type names (bytes)
g: Total number of attributes defined for abstract data types
h: Total number of abstract data types defined as subtypes
i: Total number of abstract data types
j: Average length of table identifier (bytes)

(n) Formula 14

Dictionary table name	Formula
SQL_IOS_GENERATIONS	

- a*: Number of files that compose the replica RDAREAs
b: Data dictionary RDAREA page length (bytes)
c: Average length of names of HiRDB files that compose the RDAREAs (bytes)

(o) Formula 15

Dictionary table name	Formula
SQL_TRIGGERS	
SQL_TRIGGER_ACTCOND [#]	
SQL_TRIGGER_COLUMNS	
SQL_TRIGGER_DEF_SOURCE	
SQL_TRIGGER_USAGE	

a: Data dictionary RDAREA page length (bytes)

b: Total number of trigger definitions

c: Average length of trigger authorization identifiers (bytes)

d: Average length of trigger names (bytes)

e: Average length of authorization identifiers of tables defined by triggers (bytes)

f: Average length of names of tables defined by triggers (bytes)

g: Average length of old value correlation names (bytes)

h: Average length of new value correlation names (bytes)

i: Average length of specified names of trigger action procedures (bytes)

j: Average length of SQL statements when triggers were defined (bytes)

k: Number of triggers defined with UPDATE statement as the triggering event

m: Average length of column names specified in trigger events (bytes)

n: Average number of columns specified as trigger events

p: Number of resources in trigger action retrieval conditions

q: Average length of authorization identifiers of resources in trigger action retrieval conditions (bytes)

r: Average length of resource table names used in trigger retrieval action conditions (bytes)

s : Average length of specified names used in trigger action retrieval conditions (bytes)

t : Length of analysis tree for trigger action condition (bytes)

To find the value, use the following formula. All variables in this formula are specifications in the WHEN search conditions.

$$S \times 36 + T + U \times 48 + V \times 128 \\ + F1 \times 420 + F2 \times 132 + F3 \times 124 + F4 \times 296 + F5 \times F4 \times 132 \\ + A \times 140 + B \times 200 + 1,000$$

A : Number of component specification attributes

B : Number of abstract data types that appear in value expressions

$F1$: Number of system-defined scalar functions

$F2$: Number of arguments of system-defined scalar functions

$F3$: Number of user-defined functions

$F4$: Number of potential user-defined functions

$F5$: Number of arguments of user-defined functions

S : Total number of Boolean operators, arithmetic operators, constants and system-embedded scalar functions

T : Total length of constants (data types parsed by HiRDB) (bytes)

U : Number of value expressions in scalar function VALUE, CASE expressions, and CAST specifications

V : Number of column specifications

#: This is a table used by the system.

(p) Formula 16

Dictionary table name	Formula
SQL_PARTKEY	
SQL_PARTKEY_DIVISION	
SQL_DIV_TYPE	

a : Number of matrix partitions created

b : Data dictionary RDAREA page length (bytes)

c : Average number of table row partitioning conditions

d : Average length of column names specified in table row partitioning conditions (bytes)

f : Average length of authorization identifiers (bytes)

g : Average length of table identifiers (bytes)

h : Average length of table row partitioning conditions (bytes)

16. Determining RDAREA Size

n : Number of matrix partitioning tables that combine hash partitioning and key range partitioning with boundary values specified

(q) Formula 17

Dictionary table name	Formula
SQL_AUDITS	$\frac{b}{a - 48} \times \frac{\uparrow(c + d + e + f + g + h + 2 \times 10 + 10 + 8) \div 2 \uparrow \times 2}{1}$

a : Data dictionary RDAREA page length (bytes)

b : Number of HiRDB files that compose duplicated RDAREAs

c : Average length of event type names (bytes)

d : Average length of event subtype names (bytes)

e : Average length of object type name (bytes)

f : Average length of object owner name (bytes)

g : Average length of object name (bytes)

h : Average length of event executor name (bytes)

(r) Formula 18

Dictionary table name	Formula
SQL_AUDITS_REGULARIZE [#]	$\frac{b}{a - 48} \times \frac{\uparrow(c + d + e + f + 2 \times 7 + 214 + 8) \div 2 \uparrow \times 2}{1}$

a : Data dictionary RDAREA page length (bytes)

b : Number of HiRDB files that compose duplicated RDAREAs

c : Average length of object type name (bytes)

d : Average length of object owner name (bytes)

e : Average length of object name (bytes)

f : Average length of event executor name (bytes)

[#]: This is a table used by the system.

(s) Formula 19

Data dictionary table name	Formula
SQL_KEYCOLUMN_USAGE	$\frac{b}{a - 48} \times \frac{\uparrow(c + d + e + f + g + 22) \div 2 \uparrow \times 2}{1}$

a : Page length of the data dictionary RDAREA (bytes)

- b*: Total number of constraint definitions
c: Average length of the constraint authorization identifiers (bytes)
d: Average length of the constraint names (bytes)
e: Average length of the authorization identifiers for a table for which constraints have been defined (bytes)
f: Average length of the names of tables for which constraints have been defined (bytes)
g: Average length of the constraint type names (bytes)

(t) Formula 20

Data dictionary table name	Formula
SQL_TABLE_CONSTRAINTS	
SQL_CHECKS	
SQL_CHECK_COLUMNS	

- a*: Page length of the data dictionary RDAREA (bytes)
b: Total number of constraint definitions
c: Average length of the constraint authorization identifiers (bytes)
d: Average length of the constraint names (bytes)
e: Average length of the authorization identifiers for tables for which constraints have been defined (bytes)
f: Average length of the names of tables for which constraints have been defined (bytes)
g: Average length of the constraint type names (bytes)
h: Average length of SQL statements during check constraint definition (bytes)
i: Number of check constraint definitions
j: Average length of the column names specified for the columns for which check constraints have been defined (bytes)
k: Average number of columns specified for the columns for which check constraints have been defined
m: Average number of binary data items (check constraint search conditions and analysis tree for check constraints) whose length is 256 or greater and that are subject to branching.
 For details about the conditions for branching the number of binary storage pages, see *16.1.2 Calculating the number of table storage pages*.

X : Length of analysis tree for check constraint (bytes)

Value obtained from the formula shown below, where all the variables are specified in the check constraint search conditions:

In 32-bit mode: $S \times 36 + T + (U1 + U2 + U3) \times 48 + V \times 128 + 1,000$

In 64-bit mode: $S \times 72 + T + (U1 + U2 + U3) \times 96 + V \times 184 + 1,400$

S : Total number of Boolean operators, arithmetic operators (+, -, *, /, and ||), and system built-in scalar functions

T : Total length of literals (data type handled by HiRDB) (bytes)

$U1$: Number of CASE expressions and value expressions in CAST specification

$U2$: Number of value expressions in scalar functions (VALUE, SUBSTR, BIT_AND_TEST, POSITION, TIMESTAMP, VARCHAR_FORMAT, TIMESTAMP_FORMAT)

$U3$: Number of datetime formats

V : Number of column specifications

(u) Formula 21

Data dictionary table name	Formula
SQL_SYSPARAMS	2

(v) Formula 22

Data dictionary table name	Formula
SQL_PUBLICVIEW_ SAME_USERS	

a : Page length of the data dictionary RDAREA (bytes)

b : Total number of public view table definitions

c : Average number of duplicate names for each public view table[#] (bytes)

d : Average length of the table identifiers of public view tables (bytes)

e : Average length of the authorization identifiers (bytes)

$\#$: Average number of rows with the same TABLE_NAME column value in the SQL_TABLES table per public view table identifier

(w) Formula 23

Dictionary table name	Formula
SQL_SEQUENCES	

a : Page length of data dictionary RDAREAs (bytes)

b : Total number of sequence generator definitions

c : Average length of sequence generator identifiers (bytes)

- d*: Average length of authorization identifiers (bytes)
e: Average length of sequence generator start values (bytes)
f: Average length of sequence generator maximum values (bytes)
g: Average length of sequence generator minimum values (bytes)
h: Average length of sequence generator increment values (bytes)
i: Average length of RDAREA names (bytes)

(3) Calculating the number of index storage pages

The following formula is used to calculate the number of pages required to store indexes.

Formula

Number of pages needed to store indexes

= number of index storage pages for dictionary tables[#] + 12

[#]: See 16.1.3 *Calculating the number of index storage pages* to calculate the number of index storage pages for dictionary tables; the following conditions apply:

1. The variables listed in Table 16-6 *Variables used in the formulas for obtaining the number of index storage pages* must be used.
2. 30 must be used as the value for variable *b* (percentage of unused space specified with `CREATE INDEX`).
3. 12 must be used as the value of variables *e* (number of keys with at least $dx + 1$ duplicated key values) and *f* (average number of key duplicates).

Table 16-6: Variables used in the formulas for obtaining the number of index storage pages

Table name	Type	Key length ^{#3} (Variable <i>g</i> ^{#1})	Number of key types (Variable <i>c</i> ^{#1})	Average number of key duplicates (Variable <i>d</i> ^{#1})
SQL_PHYSICAL_FILES	1	8	Number of servers	Average number of HiRDB files in server
	2	$g + 1$	Number of RDAREAs	Average number of HiRDB files constituting an RDAREA
SQL_RDAREAS	3	$g + 1$	Number of RDAREAs	1
	4	4		
SQL_TABLES	5	$d + e + 2$	Total number of tables + 80	1
	6	4		
SQL_COLUMNS	7	$d + e + f + 3$	$a \times b$	1
	8	$d + e + 6$		
	9	4		<i>B</i>
SQL_INDEXES	10	$d + e + 2$	<i>a</i>	$\uparrow h \div a \uparrow$
	11	$d + i + 2$	<i>h</i>	1
	12	4		
SQL_USERS	13	$d + 1$	Number of authorization identifiers	1
SQL_RDAREA_PRIVILEGES	14	$d + 1$	Number of unique authorization identifiers specified in the <code>USER USED BY</code> operand of the database initialization utility (<code>pdinit</code>)	Average number of RDAREA access privileges used per user

16. Determining RDAREA Size

Table name	Type	Key length ^{#3} (Variable g ^{#1})	Number of key types (Variable c ^{#1})	Average number of key duplicates (Variable d ^{#1})
	15	$g + 1$	Number of RDAREAs specified in the USER USED BY operand of the database initialization utility (pdinit)	Average number of users for each RDAREA
SQL_TABLE_PRIVILEGES	16	$d + 1$	a	$\uparrow y \div a \uparrow$
	17	$2 \times d + e + 3$	y	1
SQL_DIV_TABLE	18	$d + e + 6$	Total number of table partitions	1
	19	$g + 1$	Number of unique RDAREAs specified when partitioning table	Average number of tables stored in RDAREAs
	20	4		
SQL_DIV_TABLE_REGULARSIZE	21	$d + e + 6$	Total number of table partitions	1
	22	4	Number of row-partitioned tables	Average number of table partitions
SQL_INDEX_COLINF	23	$d + e + 6$	Number of unique tables with index definitions	Average number of columns constituting an index
	24	$d + i + 6$	Number of columns constituting an index	1
SQL_TABLE_STATISTICS	25	$d + e + 2$	Total number of tables (including dictionary tables)	1
SQL_COLUMN_STATISTICS	26	$d + e + f + 3$	h	1
SQL_INDEX_STATISTICS	27	$d + e + 2$	Total number of tables for which an index is defined	$\uparrow h \div a \uparrow$
	28	$d + i + 2$	h	1
SQL_VIEW_TABLE_USAGE	29	$d + e + 2$	z	1
	30	$d + e + 2$	Total number of base tables for which a view is defined	Average number of view definitions for a table
	31	4	z	1
SQL_VIEWS	32	$d + e + 2$	z	1
	33	4		
SQL_VIEW_DEF	34	$d + e + 2$	z	1
	35	10		
SQL_REFERENTIAL_CONSTRAINTS	39	$d + e + 2$	Total number of referential constraints	1
	40	$d + e + 2$	Total number of referencing tables	Average number of referencing tables per table
	41	$d + e + 2$	Total number of referenced table	Average number of referenced tables per table

Table name	Type	Key length ^{#3} (Variable g ^{#1})	Number of key types (Variable c ^{#1})	Average number of key duplicates (Variable d ^{#1})
SQL_EXCEPT	86	$d + e + 2$	Number of indexes for which an exception value is specified	Number of indexes with an exception value specified for a single table
	87	$d + i + 2$		1
	88	4	Number of unique tables with an index for which an exception value is specified	Number of indexes for which an exception value is specified for a single table
SQL_DIV_INDEX	36	$d + e + 6$	Number of row-partitioned indexes \times number of partitions	Average number of partitions per table
	37	$d + i + 2$	Total number of row-partitioned indexes	1
SQL_DIV_COLUMN	38	$d + e + f + 3$	Number of LOB column definitions	Average number of partitions per table
	52	$d + e + 9$	Number of LOB attribute definitions	1
SQL_ROUTINES	43	$d + \text{MAX}(q, 7)$	$p + 174$	1
	44	$d + \text{MAX}(t, 18)$	$u + 65$	1
	45	4	$p + 174$	1
	53	$d + UDT$	Number of abstract data types + 1 (NULL value)	Average number of routines per abstract data type + number of NULL
SQL_ROUTINE_RESOURCES	46	$d + q$	$p \times s$	Average number of resources used per routine
	47	$d + t$		
	48	$d + q$		
	49	4		
SQL_ROUTINE_PARAMS	50	$d + \text{MAX}(q, 8)$	$p \times r + 347$	Average number of parameters per routine
	51	$d + \text{MAX}(t, 19)$	Number of routines	Average number of parameters per specific name (if less than 3, use 3)
	106	$e + 4 + 2$	Number of trigger SQL objects $\times r + 1$ (NULL values)	Average number of parameters per specified name (if less than 3, use 3) + number of NULL values
SQL_DATATYPES	54	$d + UDT$	Number of abstract data types	1
	55	4		
	56	$d + UDT$	Number of abstract data types that have subtypes + 1 (NULL value)	Average number of subtypes per abstract data type + number of NULL values
SQL_DATATYPE_DESCRIPTOR	57	$d + UDT + ATT$	$NUDT \times NATT$	1
	58	4	Number of abstract data types	Average number of attributes per abstract data type
SQL_TABLE_RESOURCES	59	$d + e$	Total number of tables that use user-defined data types	Average number of user-defined data types used per table
	60	$d + UDT$	UDT	Average number of tables used per user-defined data type
	61	4		

Table name	Type	Key length ^{#3} (Variable g ^{#1})	Number of key types (Variable c ^{#1})	Average number of key duplicates (Variable d ^{#1})
SQL_PLUGINS	62	$d + PLG$	Number of plug-ins	1
	63	4		
	64	$\{(d + UDT) \div IXT\} + 2$	Number of data type plug-ins + number of index-type plug-ins	
SQL_PLUGIN_ROUTINES	65	t	$NPLG \times NFPLG$	1
	66	$PLG + TMD + 2$	Number of operations	Number of plug-ins
	67	$PLG + 4$		
	68	$POPR + 1$		
SQL_PLUGIN_ROUTINE_PARAMS	69	$t + PRM$	$NPLG \times NPPAR$	1
	70	PLG	$NPLG$	Average number of parameters per plug-in
	71	$t + 4$	$NPLG \times NPPAR$	1
SQL_REGISTRY_CONTEXT	72	$CNM + 1$	Number of contexts	1
SQL_REGISTRY_KEY	73	$KNM + 6$	Number of keys	1
SQL_INDEX_TYPES	74	$d + IXT$	Number of index types to be created	1
	75	4		
SQL_INDEX_RESOURCES	76	$d + IXT$	Number of plug-in indexes	Average number of index definitions that use index type
	77	4		
SQL_INDEX_DATATYPE	78	$d + e$	Number of table definitions for which plug-in index is defined	Average number of plug-in indexes for the same table
	79	$d + i$	Number of plug-in indexes	1
SQL_INDEX_FUNCTION	80	$d + e$	Number of table definitions for which plug-in index is defined	Average number of plug-in indexes per table \times average number of functions to which plug-in index is applied
	81	$d + i$	Number of plug-in indexes	Average number of functions to be applied per plug-in index
SQL_TYPE_RESOURCES	82	$d + e$	Number of user-defined data types that use a user-defined data type	Average number of user-defined data types that are specified as an attribute of a user-defined data type
	83	$d + UDT$	Number of user-defined data types	Average number of user-defined data types that use a user-defined data type
	84	4		
SQL_INDEX_TYPE_FUNCTION	85	$d + IXT$	Number of index types	Average number of functions to be applied per index
SQL_TRIGGERS	90	$d + e + 2 + 16$	Number of triggers	1
	91	$d + TRIG + 2$		
	92	$d + t + 2$		
	93	4		

Table name	Type	Key length ^{#3} (Variable g ^{#1})	Number of key types (Variable c ^{#1})	Average number of key duplicates (Variable d ^{#1})
SQL_TRIGGER_ACTCOND	94	$d + TRIG + 2 + 4$	Number of triggers with action conditions	1
	95	$d + e + 2$	Number of tables that define triggers	Average number of triggers per table
SQL_TRIGGER_COLUMNS	96	$d + TRIG + 2$	Number of triggers that use the UPDATE statement as the triggering event	Average column length specified per trigger
	97	$d + e + f + 3$	Number of columns specified that use the UPDATE statement as the triggering event	1
SQL_TRIGGER_DEF_SOURCE	98	$d + TRIG + 2 + 4$	Number of triggers	1
	99	$d + e + 2$	Number of tables defined with triggers	Average number of triggers per table
SQL_TRIGGER_USAGE	100	$d + TRIG + 2$	Number of triggers that reference resources while performing the trigger action search condition	Average number of resources referenced per trigger
	101	$d + e + 2$	Number of tables that define triggers that reference resources while performing the trigger action search conditions	Average number of resources referenced per table
	102	$d + e + (t \text{ or } e) + 2$	Number of resources used	1
	103	8		
SQL_PARTKEY	104	$d + e + f + 3$	Number of matrix partitioning tables \times number of partitioning keys	1
SQL_PARTKEY_DIVISION	105	$d + e + 6$	Number of matrix partitioning tables \times number of boundary values \times number of partitioning keys	1
SQL_AUDITS	107	$ETP + EST + 2$	Number of monitored events	1
	108	$OTP + OSC + ONM + 3$	Number of monitored objects	Average number of events monitored per object
	109	d	Number of monitored users	Average number of events monitored per user
SQL_AUDITS_REGULARIZE	110	$OTP + OSC + ONM + 3$	Number of monitored objects	Average number of events monitored per object
	111	d	Number of monitored users	Average number of events monitored per user
SQL_KEYCOLUMN_USAGE	112	$d + CNS + 2$	Number of constraints	1
	113	$d + e + 2$	Number of tables for which constraints have been defined	Average number of constraints per table

16. Determining RDAREA Size

Table name	Type	Key length ^{#3} (Variable g ^{#1})	Number of key types (Variable c ^{#1})	Average number of key duplicates (Variable d ^{#1})
SQL_TABLE_CONSTRAINTS	114	$d + CNS + 2$	Number of constraints	1
	115	$d + e + 2$	Number of tables for which constraints have been defined	Average number of constraints per table
SQL_CHECKS	116	$d + CNS + 2$	Number of check constraints	1
	117	$d + e + 2$	Number of tables for which check constraints have been defined	Average number of check constraints per table
SQL_CHECK_COLUMNS	118	$d + CNS + 2$	Number of check constraints	Average number of columns specified per check constraint
	119	$d + e + f + 3$	Number of columns used in the check constraints	Average number of duplicate columns used in check constraints per table
SQL_SYSPARAMS	121	8	2	1
SQL_PUBLICVIEW_SAMPLE_USERS	124	$d + e + 2$	Number of public view tables \times number of authorization identifiers	1
SQL_INDEX_XMLINF	125	$d + e + 2$	NPSIT	Average number of substructure paths making up an index
	126	$d + i + 7$	NPSS	1
SQL_SEQUENCES	127	$d + SEQN + 2$	Number of sequence generators in system	1
	128	4	Number of sequence generators in system	1

a: Total number of tables

b: Average number of table columns

c: Page length of the data dictionary RDAREA (bytes)

d: Average length of authorization identifiers (bytes)

e: Average length of table identifiers (bytes)

f: Average length of column names (bytes)

g: Average length of RDAREA names (bytes)

h: Total number of indexes

i: Average length of index identifiers (bytes)

n: Average length of HiRDB file names constituting RDAREAs (bytes)

p: Number of routines to be created

q: Average length of routine names (bytes)

r: Average number of parameters per routine

s: Average number of resources used per routine

t: Average length of specific names^{#2} (bytes)

u: Total number of specific names^{#2}

y: Number of access privileges defined

If the privilege is granted to *n* users per table, then the value of this variable would be the number of tables $\times n$.

z: Total number of view definitions
NUDT: Number of user-defined data types to be created
UDT: Average length of user-defined data type names (bytes)
NATT: Average number of attributes per user-defined data type
ATT: Average length of attributes with user-defined data type (bytes)
PLG: Average length of plug-in names (bytes)
NPLG: Number of plug-ins to be created
IXT: Average length of index type names (bytes)
NFPLG: Average number of plug-in functions
POPR: Average length of plug-in function names (bytes)
NPPAR: Average number of parameters per plug-in function
PRM: Average length of parameter names per plug-in function (bytes)
TMD: Average length of timing description (bytes)
CNM: Average length of context name (bytes)
KNM: Average length of key name (bytes)
TRIG: Average length of trigger names (byte)
ETP: Average length of event type names (bytes)
EST: Average length of event subtype names (bytes)
OTP: Average length of object type names (bytes)
OSC: Average length of object owner names (bytes)
ONM: Average length of object names (bytes)
CNS: Average length of constraint names (bytes)
NPSIT: Number of tables that define substructure indexes
NPSS: Number of substructure paths making up a substructure index
SEQN: Average length of sequence generator identifiers (bytes)
#1: Variables shown in (2) *Variables used in formulas* in 16.1.3 *Calculating the number of index storage pages*.
#2: Indicates *authorization-identifier.routine-identifier*.
#3: The key length is rounded up in units of four bytes. Use the following formula to obtain the key length:
 • $\lceil key\ length \rceil \div 4 \lceil \rceil \times 4$

16.2.2 Determining the size of a data dictionary RDAREA for storing database state analyzed tables and database management tables

You can use the following formula to determine the size of a data dictionary RDAREA for which `datadictionary` of `dbmanagement` is specified in the `create rdarea` statement:

Formula

Size of data dictionary RDAREA = $(\lceil (a \times 1.3) \rceil \div b \lceil \rceil) \times b \times 4,096$
 (bytes)

a: Total number of pages in the data dictionary RDAREA^{#1}

b: Segment size of the data dictionary RDAREA^{#2}

#1

Number of table storage pages + *Number of index storage pages*. For details, see (1) and (2) below.

#2
Segment size specified in the `create rdarea` statement of the database structure modification utility (`pdmod`).

(1) How to determine the number of table storage pages

The number of table storage pages is the sum of Formula 1 and Formula 2:

Formula 1	<div><div></div><div>$\frac{(a+c+e+g) \times (b+1) + 120}{96} \times 30$</div><div></div></div>
Formula 2	<div><div></div><div>$\frac{(a+c+e+g) \times (b+1) + 120}{9} \times 30$</div><div></div></div>

- a*: Number of tables to be created + 61
- b*: Average number of partitions in the table storage RDAREA
If the RDAREA is not partitioned, the value is 1. The average value is rounded up.
- c*: Number of indexes to be created + 124
- e*: Total number of BLOB columns defined for the tables to be created + 3
- g*: Total number of BLOB attributes defined for the tables to be created

(2) How to determine the number of index storage pages

Use the following formula to determine the number of index storage pages:

Number of pages for index storage for the data dictionary tables [#] × 2		
#: See 16.1.3 <i>Calculating the number of index storage pages</i> to determine the number of pages for index storage for the data dictionary tables. The following is the calculation condition:		
1. The ratio of unused space specified in <code>CREATE INDEX</code> is 0.		
2. Use the following variables in the formula:		
Key length (variable <i>g</i> [#])	Number of key types (variable <i>c</i> [#])	Average number of duplicate keys (variable <i>d</i> [#])
12	$(a + c + e + g) \times (b + 1) + 120$	30

- a*: Number of tables to be created
 - b*: Average number of partitions in the table storage RDAREA
 - c*: Number of indexes to be created
 - e*: Total number of BLOB columns defined for the tables to be created
 - g*: Total number of BLOB attributes defined for the tables to be created
- There is no need to add variable *h*, since no unique index is defined in the database state analyzed table or the database management table.
- #: Variables shown in (2) *Variables used in formulas* in 16.1.3 *Calculating the number of index storage pages*.

16.3 Determining the size of the master directory RDAREA

The following formula is used to determine the size of the master directory RDAREA.

Formula

Size of master directory RDAREA (bytes)

$$= \{ \begin{aligned} & \uparrow (a + 2) \div 800 \uparrow \times 51 + \uparrow (b + 120) \div 6,000 \uparrow \times 51 + \uparrow (c + 240) \div 6,000 \uparrow \times 51 \\ & + \uparrow (d + 240) \div 64,000 \uparrow \times 51 + \uparrow e \div 64,000 \uparrow \times 51 + \uparrow f \div 50 \uparrow \times 51 + 2 + 6 \times n \\ & \}^{\#1} \times 4,096^{\#2} \end{aligned}$$

a: Total number of data dictionary RDAREAs + total number of user RDAREAs

When the inner replica facility is used, also calculate the total number of replica RDAREAs.

b: Total number of tables to be defined

c: Total number of indexes to be defined

d: Total number of view tables to be defined

e: Total number of data types and index types to be defined

f: Total number of original RDAREAs and replica RDAREAs

If the inner replica facility is not being used: 0

The information on the original RDAREAs is not deleted even if all replica RDAREAs are deleted (this information is not deleted until the original RDAREAs are deleted).

n: Number of HiRDB files that constitute the master directory RDAREA

#1: Indicates the total number of pages in the master directory RDAREA.

#2: Indicates the page length of the master directory RDAREA.

16.4 Determining the size of the data directory RDAREA

The following formula is used to determine the size of the data directory RDAREA.

Formula

Size of data directory RDAREA (bytes)

$e \ f$

$$= \left\{ \left(\sum_{i=1}^e (g_i + \sum_{j=1}^f p_j + 86) \right) \div 3,000 \uparrow \times 51 + 6 \times n + 1 \right\}^{\#1} \times 4,096^{\#2}$$

$i=1 \ j=1$

$$g_i = \left\lceil (5 \times a_i + 2 \times b_i + 2 \times c_i + 48) \div 32 \right\rceil$$

$$p_j = \left\lceil (d_j + 12) \div 16 \right\rceil$$

a_i : Number of columns constituting indexes

b_i : Number of RDAREAs storing indexes

c_i : Number of RDAREAs storing tables for which an index is defined

d_j : Number of RDAREAs storing tables

e : Total number of indexes to be defined

f : Total number of tables to be partitioned

n : Number of HiRDB files that constitute the data directory RDAREA

$\#1$: Indicates the total number of pages in the data directory RDAREA.

$\#2$: Indicates the page length of the data directory RDAREA.

16.5 Determining the size of a data dictionary LOB RDAREA

(1) Estimating the size of the data dictionary LOB RDAREA for storing sources

The following formula is used to determine the size of the data dictionary LOB RDAREA for storing sources.

Formula

Size of data dictionary LOB RDAREA for source storage (bytes)

= {

a

[$\sum_{i=1}^a S_i \div 64,000 \uparrow \times 96 + 7 + 3 \times (a - 1)]^{\#1}$

i=1

+

b

[$\sum_{j=1}^b (C_j + 1,024) \div 8,192 \uparrow]^{\#2}$

j=1

}^{#3} $\times 8,192^{\#4}$

a: Number of HiRDB files that constitute the data dictionary LOB RDAREA for storing sources

b: Total number of procedures (CREATE PROCEDURE), functions and procedures in abstract data types (each FUNCTION (excluding plug-in functions) and PROCEDURE), and user-defined functions (CREATE FUNCTION)

S_i: Number of segments for each HiRDB file specified with the create rdarea statement of the database initialization utility or database structure modification utility.

C_j: Length of each procedure (length of each CREATE PROCEDURE), function and procedure in abstract data types (length of each FUNCTION (excluding plug-in functions) and PROCEDURE), and user-defined function (length of CREATE FUNCTION)

#1: Total number of pages in the directory pages part.

#2: Total number of pages in the data pages part.

#3: Indicates the total number of pages in the data dictionary LOB RDAREA for storing sources.

#4: Indicates the page length of the data dictionary LOB RDAREA for storing sources.

(2) Estimating the size of the data dictionary LOB RDAREA for storing objects

The following formula is used to estimate the size of the data dictionary LOB RDAREA for storing objects:

Formula

Size of data dictionary LOB RDAREA for object storage (bytes)

= {

a

[$\sum_{i=1}^a S_i \div 64,000 \uparrow \times 96 + 7 + 3 \times (a - 1)]^{\#1}$

i=1

+

b

[$\sum_{j=1}^b (C_j + 1,024) \div 8,192 \uparrow]^{\#2}$

j=1

}^{#3} $\times 8,192^{\#4} + 500,000^{\#5}$

a : Number of HiRDB files that constitute the data dictionary LOB RDAREA for storing objects

b : Total number of procedures (CREATE PROCEDURE), functions and procedures in abstract data types (FUNCTION (excluding plug-in functions), PROCEDURE), user-defined functions (CREATE FUNCTION), and trigger definitions (CREATE TRIGGER).

S_i : Number of segments for each HiRDB file specified with the create rdarea statement of the database initialization utility (pdinit) or database structure modification utility (pdmod).

C_j : $QO_i + PR$ (The formula for QO_i is shown in (3), and the formula for PR is shown in (4) The variables used in these formulas are shown in (5)).

#1: Total number of pages in the directory pages part.

#2: Total number of pages in the data pages part.

#3: Indicates the total number of pages in the data dictionary LOB RDAREA for storing objects.

#4: Indicates the page length of the data dictionary LOB RDAREA for storing objects.

#5: This is added when an abstract data type or plug-in function is used.

(3) QO_i (SQL object size) formula

QO_i (bytes) =

a

$\sum \{$

$i=1$

$1,840 + 46 \times RCN + 298 \times Si + 20 \times Pi + 1,138 \times Ti + 76 \times Ti \times Di + 80 \times Ci + 40 \times Li + 534 \times Wi$

$+ 20 \times Ki + Li + 8 \times TCI + 656 \times Di + 48 \times nFF + 100 \times nFP + 148 \times nFC + 696 \times nPFF$

$+ 16 \times (nAT + nPAT) + 20 \times nCAT + 28 \times (nAF + nCAF) + 20 \times (nAA + nPAA + nCAA)$

$+ 1,057 \times nSPA + 120 \times nSPP + 287 \times nSFF + 8 \times nSFP + 813 \times nJFC + 20 \times nJFP$

$[+ 1,057 \times nTR + 120 \times (nTSN + nTSO) + 20 \times (nTCN + nTCO)]^{\#1}$

$[+ 760 + 376 \times RCC + 1,880 \times RCT]^{\#2}$

$[+ 32 \times Si + 16]^{\#3}$

$[+ \uparrow (42 \times SiT) + \{ 52 + 152 \times (SiTA + SiSA + SiNA) \times (SiT + SiS + SiN) \} \uparrow]^{\#4}$

$\}$

a : Number of SQL statements in stored procedures

#1: Add this formula if you use triggers.

#2: Add this formula if you use referential constraints.

#3: This is the formula for determining the length of the Column Name Descriptor Area (SQLCNDA). Add it for dynamic SQL statements.

#4: This is the formula for determining the length of the Type Name Descriptor Area (SQLTNDA). Add it for dynamic SQL statements.

(4) PR (routine control object size) formula

(a) When defined by the user

If you have defined a stored procedure, stored function, or trigger, use the following formula to determine the size of the routine control object:

PR (bytes) =

a

$\sum \{$

$i=1$

$$\begin{aligned}
&600 + 28 \times sRi + 32 \times (sRUi + sDi) + 56 \times sSXi + sCUi + sSi + sPi + sLA \\
&+ sKi + sL + 80 \times sWi + 24 \times sCM + 32 \times sCCR + 2 \times sDCR + 60 \times sCHD + 72 \times sDHD + 64 \times sHCN \\
&+ 8 \times sCHD \times sHCN + 48 \times nRFF + 100 \times nRFP + 148 \times nRFC + 200 \times nPRFF + 8 \times nPRFP \\
&+ 196 \times nPA + 64 \times nPP + 36 \times nPPI + 20 \times nPPO + 200 \times nPPA + 8 \times nPPP + 20 \times nAR + 48 \times nARA \\
&+ 16 \times nRPAT + 20 \times nCAT + 28 \times (nRPAF + nRCAF) + 20 \times (nRPAA + nRCAA) + 287 \times nRSFF \\
&+ 8 \times nRSFP + 813 \times nPJA + 20 \times nPJP + 813 \times nRJFC + 20 \times nRJFP \\
&[+ 28 \times (nTSN \times 2 + nTSO)]^{\#} \\
&\}
\end{aligned}$$

a: Number of the following SQL statements:

- Procedures (CREATE PROCEDURE)
- Functions and procedures in abstract data types (each FUNCTION (excluding plug-in functions), PROCEDURE)
- User-defined functions (CREATE FUNCTION)
- Trigger definitions (CREATE TRIGGER)

#: Add this formula if you use triggers.

(b) When HiRDB creates automatically

If you specified CASCADE during table definition, use the following formula to determine the size of the routine control object when HiRDB creates triggers for constraint control:

PR (bytes) =

a

$$\sum_{i=1} \{240 + 608 \times RCC + (5,120 + 100 \times RDi + 256 \times RLi) \times RCP \times RCT\}$$

a: Number of the following SQL statements:

- Procedures (CREATE PROCEDURE)
- Functions and procedures in abstract data types (FUNCTION (excluding plug-in functions), PROCEDURE)
- User defined functions (CREATE FUNCTION)
- Trigger definitions (CREATE TRIGGER)

(5) Variables used in the calculation of PR and QO_i

Variable name	Explanation
<i>R_{CN}</i>	Total number of tables and indexes used by SQL objects
<i>S_i</i>	Number of retrieval items in SQL statements (if the columns specified by SQL statements are index columns, the number of those columns)
<i>P_i</i>	Number of embedded variables or parameters in SQL statements
<i>T_i</i>	Number of table names in SQL statements
<i>C_i</i>	Number of column names in SQL statements
<i>TC_i</i>	Number of table composition columns in SQL statements
<i>W_i</i>	Number of logical operators in SQL statements [#]
<i>K_i</i>	Number of literals in SQL statements [#]
<i>L_i</i>	Total length of literals in SQL statements [#] (bytes)

Variable name	Explanation
<i>li</i>	Number of indexes used during SQL statement execution (of the tables specified by SQL statements, the number of indexes specified in retrieval conditions)
<i>Di</i>	Total number of storage conditions defined by tables used in SQL statements (count matrix partitioning tables twice)
<i>SiT</i>	Number of abstract data types in queries in SQL statements
<i>SiS</i>	Number of supertypes of abstract data types in queries in SQL statements
<i>SiN</i>	Number of subtypes of abstract data types in queries in SQL statements
<i>SiTA</i>	Number of attributes of abstract data types in queries in SQL statements
<i>SiSA</i>	Number of supertype attributes of abstract data types in queries in SQL statements
<i>SiNA</i>	Number of component specifications of abstract data types that are query subtypes in SQL statements
<i>nSPA</i>	Number of procedure calls in SQL statements
<i>nSPP</i>	Total number of procedure call parameters in SQL statements
<i>nFF</i>	Number of function calls in SQL statements [#]
<i>nFP</i>	Number of function call parameters in SQL statements [#]
<i>nFC</i>	Total number of function definition candidates among the functions in the SQL statements (to the number of function calls <i>nFF</i> , add the number of function definitions that have subtypes as arguments for which the arguments are abstract data types)
<i>nPFF</i>	Number of plug-in function calls used by SQL objects (number of plug-in function calls in SQL statements + 1 for SELECT and 6 for INSERT, UPDATE, or DELETE)
<i>nSFF</i>	Number of system definition scalar function calls in SQL statements [#]
<i>nSFP</i>	Total number of system definition scalar function arguments in SQL statements [#]
<i>nJFC</i>	Number of external Java function calls in SQL statements
<i>nJFP</i>	Total number of external Java functions in SQL statements
<i>nAT</i>	Number of abstract data types used by component specifications in SQL statements (excluding supertypes and abstract data types that emerge depending on the abstract data type attributes)
<i>nAA</i>	Number of abstract data types used by component specifications in SQL statements (including supertypes and abstract data types that emerge depending on the abstract data type attributes)
<i>nAF</i>	Total number of attributes used by component specifications in SQL statements
<i>nPAT</i>	Number of abstract data types of plug-in function arguments used by SQL objects (excluding supertypes and abstract data types that emerge depending on the abstract data type attributes)
<i>nPAA</i>	Number of abstract data types of plug-in function arguments used by SQL objects (including supertypes and subtypes)
<i>nCAT</i>	Number of constructor function calls in SQL statements
<i>nCAA</i>	Number of constructor function abstract data types in SQL statements (including supertypes)
<i>nCAF</i>	Total number of constructor function abstract data type attributes in SQL statements
<i>nTR</i>	Number of triggers activated by the execution of SQL statements
<i>nTSN</i>	Total number of columns modified by new value correlation names in SQL statements that are triggered by the execution of SQL statements
<i>nTSO</i>	Total number of columns modified by old value correlation names in SQL statements that are triggered by the execution of SQL statements
<i>nTCN</i>	Total number of columns modified by new value correlation names in the trigger action conditions of triggers that are activated by the execution of SQL statements

Variable name	Explanation
<i>nTCO</i>	Total number of columns modified by old value correlation names in the trigger action conditions of triggers that are activated by the execution of SQL statements
<i>RCC</i>	Total number of foreign key component columns and primary key component columns of the tables that reference update-target tables in SQL statements
<i>RCT</i>	Sum of the number of tables that reference update-target tables and the number of tables that are referenced by update-target tables in SQL statements
<i>RCP</i>	Total number of CASCADES specified for referencing action when referencing tables are defined
<i>Rli</i>	Total number of indexes defined for referenced tables with reference specified when referencing tables are defined
<i>RDi</i>	Total number of partition storage conditions defined for referenced tables with reference specified when referencing tables are defined (double the value for matrix partitioning tables)
<i>sRi</i>	Number of SQL parameters in procedures and functions (count SQL parameters specified with INOUT twice)
<i>sRUi</i>	Total number of SQL parameters in procedures and functions (or total number of columns modified by new or old value correlation names in the triggered SQL statements defined by triggers)
<i>sDi</i>	Total number of SQL variables (declare) in procedures, functions, and triggered SQL statements
<i>sSXi</i>	Total number of SQLCODE and SQLCOUNT variables in procedures, functions, and triggered SQL statements
<i>sCUi</i>	Total number of CURRENT_TIME and CURRENT_DATE value functions in procedures, functions, and triggered SQL statements
<i>sSi</i>	Number of data manipulation SQL statements in procedures and triggered SQL statements (excluding cursor declarations: OPEN, FETCH, CLOSE, UPDATE, DELETE, INSERT)
<i>sPi</i>	Number of routine control SQL statements in procedures, functions, and triggered SQL statements (BEGIN, SET, IF, ELSEIF, WHILE)
<i>sLA</i>	Number of labels in procedures, functions, and triggered SQL statements
<i>sKi</i>	Number of literals in procedures, functions, and triggered SQL statements (excluding data manipulation SQL literals described in procedures and triggered SQL statements)
<i>sL</i>	Total length of constants in procedures, functions, and triggered SQL statements (excluding data manipulation SQL literals described in procedures and triggered SQL statements)
<i>sWi</i>	Number of conditional predicates in procedures, functions, and triggered SQL statements
<i>sCM</i>	Number of compound statements in procedures, functions, and triggered SQL statements
<i>sCCR</i>	Number of compound statements that describe cursor declarations in procedures and triggered SQL statements
<i>sDCR</i>	Number of cursor declarations in procedures and triggered SQL statements
<i>sCHD</i>	Number of compound statements that specify handler declarations in procedures, functions, and triggered SQL statements
<i>sDHD</i>	Number of handler declarations in procedures, functions, and triggered SQL statements
<i>sHCN</i>	Number of condition values specified in handler declarations in procedures, functions, and triggered SQL statements
<i>nRFF</i>	Number of function calls in routines
<i>nRFP</i>	Total number of function arguments in routines
<i>nRFC</i>	Total number of function definition candidates among the routines in the SQL statements (to the number of function calls <i>nRFF</i> , add the number of function definitions that have subtypes as arguments for which the arguments are abstract data types)
<i>nPRFF</i>	Number of plug-in function calls used by routine SQL objects
<i>nPRFP</i>	Total number of plug-in parameters of plug-in function calls used by routine SQL objects

Variable name	Explanation
<i>nPA</i>	Number of procedure calls in routines
<i>nPP</i>	Total number of procedure parameters in routines
<i>nPPI</i>	Total number of input parameters in routine procedures (including input/output parameters)
<i>nPPO</i>	Total number of output parameters in routine procedures (including input/output parameters)
<i>nPPA</i>	Number of plug-in procedure calls in routine SQL objects
<i>nPPP</i>	Total number of plug-in parameters of plug-in procedures used by routine SQL objects
<i>nRSFF</i>	Number of system defined scalar function calls in routines
<i>nRSFP</i>	Total number of system defined scalar function call arguments in routines
<i>nPJA</i>	Number of external Java procedure calls in routine
<i>nPJP</i>	Total number of arguments of external Java procedures in routine
<i>nRJFC</i>	Number of external Java function calls in routine
<i>nRJFP</i>	Total number of arguments of external Java functions in routine
<i>nAR</i>	Number of abstract data types used by component specifications in routines (excluding supertypes and abstract data types that emerge depending on the abstract data type attributes)
<i>nARA</i>	Total number of attributes used by component specifications in routines
<i>nRPAT</i>	Total number of abstract data types used as parameters of plug-in routines used by routine SQL objects (excluding abstract data types that are supertypes or abstract data type attributes)
<i>nRPAA</i>	Number of abstract data types used as parameters of plug-in routines used by routine SQL objects (including supertypes)
<i>nRPAF</i>	Total number of attributes of abstract data types used as parameters of plug-in routines used by routine SQL objects
<i>nRCAT</i>	Number of constructor function calls in routines
<i>nRCAA</i>	Number of abstract data types of constructor functions in routines (including supertypes)
<i>nRCAF</i>	Total number of abstract data type attributes of constructor functions in routines

#: When triggers are used, all of the trigger activation conditions of the triggers activated by execution of SQL statements must be counted.

16.6 Determining the size of a user LOB RDAREA

The following formula is used to determine the size of a user LOB RDAREA.

Formula

Size of user LOB RDAREA (bytes)

$$= (\text{total number of pages in the directory page part} + \text{total number of pages in the data page part}) \times 8,192^{\#}$$

$\#$: Page length of user LOB RDAREA.

(1) Total number of pages in the directory page part

Formula

Total number of pages in the directory page part

a

$$= \sum_{i=1}^a Si \div 64,000 \uparrow \times 96 + 7 + 3 \times (a - 1)$$

$i=1$

a : Number of HiRDB files that constitute the user LOB RDAREA

Si : Number of segments for each HiRDB file specified with the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`)

(2) Total number of pages in the data page part

Formula

Total number of pages in the data page part

b

$$= \sum_{j=1}^b (C_j + 1,024) \div 8,192 \uparrow$$

$j=1$

b : Total number of rows in LOB columns

Count rows with a data length of 0, but do not count rows with the NULL value.

C_j : Length of each BLOB data (bytes)

16.7 Determining the size of the registry RDAREA

The following formula is used to determine the size of the registry RDAREA.

Formula

Registry RDAREA size (bytes)

$$= \text{registry RDAREA page length}^{\#} \times \text{total number of registry RDAREA pages} \times 1.3$$

#

This is the page length specified by the `create rdarea` statement of the registry facility initialization utility (`pdreginit`).

Finding the total number of registry RDAREA pages

Total number of registry RDAREA pages (pages)

a

$$= \sum_{i=1} \{ \uparrow Si \div d \uparrow + \uparrow Si \div e \uparrow + 6 \times (a + 1) + 2 \times \uparrow 20,480 \div b \uparrow$$

+ number of pages storing registry management tables + number of pages storing indexes of registry management table

a : Number of HiRDB files that constitute the registry RDAREA

b : Page length of the registry RDAREA (bytes)

c : Segment size specified by the `create rdarea` statement of the registry facility initialization utility (`pdreginit`)

d : $\downarrow (b - 20) \div \{ (\uparrow c \div 32 \uparrow \times 8) + 56 \} \downarrow$

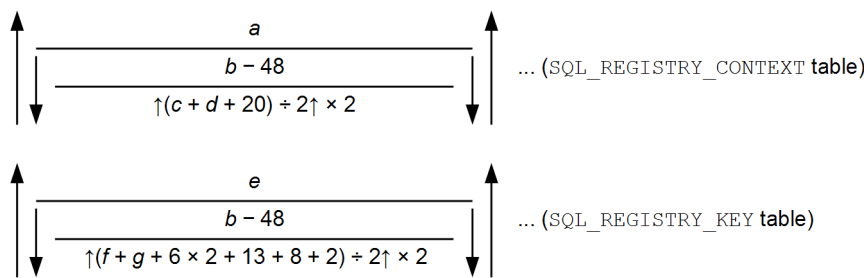
e : $\downarrow (125 \times b) \div (16 \times d) \downarrow \times d$

Si : Number of segments of each HiRDB file specified by the `create rdarea` statement of the registry facility initialization utility (`pdreginit`)

Each table or index is allocated in segments. The value obtained for each table or index is rounded up in segments.

(1) Number of pages storing registry management tables

Formula



a : Number of contexts for registry management tables

b : Page length of the registry management table

c : Length of registry context names

d : Length of access passwords

e : Number of registry management table key values (number of key names registered in the registry management table)

f : Length of registry key name

g : Length of registry key value (add when the registry key value length is 32,000 bytes or less)

(2) Number of pages storing registry management table indexes

Formula

Number of pages storing registry management table indexes (pages)
 = number of pages storing indexes of `SQL_REGISTRY_CONTEXT` tables
 + number of pages storing indexes of `SQL_REGISTRY_KEY` tables

For details about the number of pages storing indexes of `SQL_REGISTRY_CONTEXT` tables and the number of pages storing indexes of `SQL_REGISTRY_KEY` tables, see *16.1.3 Calculating the number of index storage pages*. However, calculate the formula using 30% as the percentage of unused areas specified in the `CREATE INDEX` statement.

The following table lists the values used in the formula for the number of pages storing indexes.

Values used in formula for number of pages storing registry RDAREA indexes

Name of table	Type	Key length	Key type	Average duplication level
<code>SQL_REGISTRY_CONTEXT</code>	72	$a + 1$	Number of context names (number of registry management table contexts)	1
<code>SQL_REGISTRY_KEY</code>	73	$f + 6$	Number of key values (number of registry management table key values)	1

a : Length of registry contexts

f : Length of registry key names

16.8 Determining the size of the registry LOB RDAREA

The following formula is used to determine the size of the registry LOB RDAREA.

Formula

Size of registry LOB RDAREA (bytes)

$$= (\text{total number of pages in the directory page part} + \text{total number of pages in the data page part}) \times 8,192^{\#}$$

$\#$: Page length of the registry LOB RDAREA

(1) Total number of pages in the directory page part

Formula

Total number of pages in the directory page part

a

$$= \sum_{i=1}^{\uparrow} S_i \div 64,000 \uparrow \times 96 + 7 + 3 \times (a - 1)$$

$i=1$

a : Number of HiRDB files that constitute the registry LOB RDAREA

S_i : Number of registry LOB RDAREA segments

(2) Total number of pages of data page part

Formula

Total number of pages in the data page part

b

$$= \sum_{j=1}^{\uparrow} (C_j + 1,024) \div 8,192 \uparrow$$

$j=1$

b : Number of registry key values that exceed 32,000 bytes

C_j : Length of registry key values that exceed 32,000 bytes

16.9 Determining the size of the list RDAREA

Use the following formula to determine the size of the list RDAREA:

Formula

Size of list RDAREA (bytes)

= {

a

$\sum_{i=1}^n (\lceil Si \div f \rceil + \lceil Si \div g \rceil)$

$i=1$

$+ 6 \times (a + 1) + \lceil (1,024 \times n) \div (25 \times b) \rceil + \lceil 20,480 \div b \rceil + c \times e$

$\} \times b$

a : Number of HiRDB files that constitute the list LOB RDAREA

b : Page length of the list RDAREA[#] (bytes)

c : Number of segments for the list RDAREA

e : Size of a segment for the list RDAREA[#]

f : $\lfloor \{b - 20\} \div \{\lceil e \div 32 \rceil \times 8\} + 56 \rfloor$

g : $\lceil (125 \times b) \div (16 \times f) \rceil \times f$

n : Maximum number of lists that can be created in one list RDAREA[#]

Use the following formula to find the maximum number of list RDAREAs.

$\lceil \text{Total number of lists retained in server} \div \text{number of list RDAREAs placed in server} \rceil$

Si : Number of HiRDB file segments that constitute the list RDAREA[#]

[#]: Use the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`), or the `initialize rdarea` statement of the database structure modification utility to specify the 500 smallest integer multiple values at or above the value found using the above formula.

17

Determining the Size of System Files and Audit Trail Files

This chapter describes how to determine the size of system files such as system log files, synchronization point dump files and status files, as well as the size of audit trail files.

17.1 Determining the size of system log files

This section describes the methods for determining the size of system log files. The topics covered include:

- Total size of system log files
- Amount of system log information that is output during table definition
- Amount of system log information that is output during index definition
- Amount of system log information that is output during table data updating
- Amount of system log information that is output during database creation by a utility
- Amount of system log information that is output depending on the SQL manipulation
- Amount of system log information that is output during the definition of extended system-defined scalar functions
- Amount of system log information that is output during the execution of the RDAREA automatic extension facility
- Amount of system log information that is output during execution of the PURGE TABLE statement
- Amount of system log information that is output during execution of the free page release utility
- Amount of system log information that is output during execution of the facility for predicting reorganization time
- Amount of system log information that is output during updatable backup holds

For details about estimating the size of the system log files and notes on the usage of updatable online reorganization, see the *HiRDB Staticizer Option Version 9 Description and User's Guide*.

17.1.1 Total size of system log files

(1) How to obtain the total size of system log files

The following formula is used to obtain the total size of all system log files.

Formula

Total size of all system log files (bytes) = $(\uparrow a \div c \uparrow \times 3)^{\#} \times b$

a: Amount of system log information to be output

For the procedure for obtaining this value, see (2), as follows.

b: System log file record length that is specified in the `pd_log_rec_leng` operand

c: Use one of the following values:

```
pd_log_rec_leng = 1024: 1000
pd_log_rec_leng = 2048: 2000
pd_log_rec_leng = 4096: 4000
```

Notes

- Because there may be unused space in system log files, it is recommended that the allocation size be at least 1.2 times the value obtained.
- Because extension information is added when the inner replica facility is used, allocate 1.1 times the value obtained.
- The value obtained is converted to a size per unit of time, and then an estimate is made of the size of one system log file and the number of system log files. The unload intervals for the system log files must be taken into account.

#: This is the formula for obtaining the total number of system log file records.

(2) Determining the size of system log information

The following formula is used to obtain the amount of system log information that will be output.

Formula

Amount of system log information to be output (bytes) = $\sum \{b + e + \lceil b \div (a - 256) \rceil \times 256 + d\} + c$

■ For a recovery-unnecessary front-end server only, add:

+f

\sum refers to the total for each transaction. However, if an executing transaction satisfies all the following conditions, that part will not be output to the system log:

- The transaction is performing a retrieval, and it ends with the COMMIT statement.
- The server that is executing the transaction is a single server, a dictionary server, or a back-end server.

a: Value of the `pd_log_max_data_size` operand

b: 1,336 + amount of system log information that is output depending on the database manipulation

Note that the value is 0 for a recovery-unnecessary front-end server.

This is the total amount of system log information that is obtained for each transaction, as explained in 17.1.2 *Amount of system log information output during table definition* through 17.1.5 *Amount of system log information output during database creation by a utility*.

c: Amount of system log information that is output depending on the SQL manipulation

See 17.1.6 *Amount of system log information that is output depending on the SQL manipulation*.

d: Value calculated using the following formula for each server type

- For a HiRDB single server configuration
 $2 \times \text{pd_log_rec_leng operand value}$
- For a HiRDB parallel server configuration
 - For a front-end server

Execution of UAP using the HiRDB library to implement a connection method that supports multi-threading and complies with X/Open, and use of the transaction transfer facility	Use of recovery-unnecessary front-end server	Formula
Used	Not used (cannot be used)	$(\lceil 1,336 \div \text{value of pd_log_rec_leng operand} \rceil + 3) \times \text{value of pd_log_rec_leng operand}$
Not used	Used	0
Not used	Not used	$2 \times \text{value of pd_log_rec_leng operand}$

- For a back-end server or a dictionary server

The system includes a front-end server that uses a recovery-unnecessary front-end server	pd_rpl_reflect_mode operand specification value	Formula
Yes	uap	$\text{Value of pd_log_rec_leng operand} + \lceil \{176 + 128 \times (\text{maximum number of BESs updated in one transaction} + 1)\} \div \text{value of pd_log_rec_leng operand} \rceil \times \text{value of pd_log_rec_leng operand}$
	server	
No	uap	$2 \times \text{value of pd_log_rec_leng operand}$
	server	

e:

- For a HiRDB parallel server configuration
For a front-end server: $8 + 72 \times \text{maximum number of BESs and DSs accessed or updated in a single transaction}$

Other than a front-end server: 0

- For a HiRDB single server configuration: 0

$f: 10 \times \text{value of } \text{pd_log_rec_leng}$

Notes

The system log is output when any of the following operations takes place:

- Table definition
- Index definition
- Table data updating
- Database creation by a utility

If rollback occurs while the database is being updated during any of these operations, the amount of system log information applicable to the part of the database updated up to that point is added and output; this must be taken into account in estimating the size of system log information.

(3) Notes on determining the amount of system log information that is output

Large amounts of system log information are output when the following SQLs are executed. Since system log file capacity might run short, execute these only after determining the amount of system log information that will be output. If your calculations indicate that system log file capacity will be insufficient, take the actions indicated below.

SQL statements that output large amounts of system log information	How to avoid running short of system log file capacity
CREATE INDEX, Method 1 (index definition)	Define an index that specifies <code>EMPTY</code> in its index option, and create the index using the database reorganization utility (<code>pdreorg</code>). When you create the index, specify pre-update log acquisition mode as the log acquisition mode.
CREATE INDEX, Method 2 (index definition) CREATE INDEX, Method 3 (substructure index definition)	Execute the SQL code in no-log mode. When you execute one of the SQLs at left, a particularly large amount of system log information will be output. For this reason, we recommend running on no-log mode regardless of the amount of system log file capacity. For details about operation when UAPs or utilities are executed in the no-log mode, see the <i>HiRDB Version 9 System Operation Guide</i> .
DROP SCHEMA (delete schema)	Individually delete the definitions shown below in the schema of the specified authorization identifier to reduce the amount of system log information that is output at once. <ul style="list-style-type: none"> • Base tables • View tables (including public views) • Indexes • Access privileges • Routines (including public procedures and public functions that define authorization identifiers specified by <code>DROP SCHEMA</code>) • Triggers • Abstract data types • Index types

17.1.2 Amount of system log information output during table definition

The formulas used to obtain the amount of system log information that is output during table definition are presented as follows.

(1) HiRDB single server configuration

Condition	Amount of system log information (bytes)
Table not partitioned	$((1,256 \times b + 2,500) \times 1.2) \times a + (632 \times a \times d)^{\#}$
Table partitioned	$((1,256 \times b + 1,800 \times c + 2,500) \times 1.2) \times a + (632 \times a \times d)^{\#}$

a: Total number of tables to be defined

b: Average number of columns in tables to be defined

c: Average number of partitions in tables to be defined

d: Number of RDAREAs for storing LOB columns

#: Add this value if a LOB column is defined in any of the tables.

(2) HiRDB parallel server configuration

Condition	Amount of system log information (bytes)
Amount of system log information output by dictionary server	$((1,256 \times b + 1,800 \times c + 2,500) \times 1.2) \times a$
Amount of system log information output by back-end server	$912 \times a \times d + (632 \times a \times e)^{\#}$

a: Total number of tables to be defined

b: Average number of columns in tables to be defined

c: Average number of partitions in tables to be defined

d: Number of partitions in the back-end server for tables to be defined

e: Number of RDAREAs for storing LOB columns in the back-end server

#: Add this value if a LOB column is defined in any of the tables that are defined in the back-end server.

17.1.3 Amount of system log information output during index definition

The formulas used to obtain the amount of system log information that is output during index definition (including when a primary key is added) are as follows.

(1) HiRDB single server configuration

Condition	Amount of system log information (bytes)
Indexes not partitioned	n $\sum_{i=1} \{ 5,624 + 800 \times b + 1,256 \times \lceil b \times 5 \div 24 \rceil + 1,256^{\#} + 288 \times g$ $+ (132 + X_i \times \lceil (100 - f) \div 100 \rceil) \times W_i + 272 \times W_i + 1,940 \times \lceil W_i \div V_i \rceil$ $\}$
Indexes partitioned	n $\sum_{i=1} \{ 5,124 + 800 \times b + 500 \times c + 800 \times a + 1,256 \times \lceil b \times 5 \div 24 \rceil$ $+ 1,256^{\#} + 288 \times g +$ a $\sum \{ (132 + X_{ij} \times \lceil (100 - f) \div 100 \rceil) \times W_{ij} + 272 \times W_{ij}$

Condition	Amount of system log information (bytes)
	$j=1$ $+ 1,940 \times \lceil W_{ij} \div V_{ij} \rceil$ $\}$

#: If there is no exception value specification, 0.

a: Number of partitions for table for which index is defined

b: Number of index component columns

c: Number of RDAREAs that store indexes

f: Value (%) of the PCTFREE operand specified during index definition

This is the percentage of unused space in a page.

g: Number of replica generations of RDAREAs that store indexes

If there are no replica RDAREAs, 0.

n: Total number of indexes to be defined

V: Segment size of user RDAREA used to store index (pages)

W: Number of index storage pages

For details, see 16.1.3 *Calculating the number of index storage pages*.

If EMPTY is specified by CREATE INDEX, 0.

X: Page length of user RDAREA used to store index (bytes)

(2) HiRDB parallel server configuration

Condition	Amount of system log information (bytes)
Amount of system log information output by dictionary server	N $\sum_{i=1} \{ 5,124 + 800 \times b + 500 \times c + 800 \times a + (1,256 \times \lceil b \times 5 \div 24 \rceil) + 1,256^{\#} + 288 \times g \}$
Amount of system log information output by back-end server	N $\sum_{i=1} \{ 814 + (132 + X_i \times \lceil (100 - f) \div 100 \rceil) \times W_i + 272 \times W_i + 1,940 \times \lceil W_i \div V_i \rceil \}$

#: If there is no exception value specification, 0.

a: Number of partitions for table for which index is defined

b: Number of index component columns

c: Number of RDAREAs used to store index

f: Value (%) of the PCTFREE operand specified during index definition

This is the percentage of unused space in a page.

g: Number of replica generations of RDAREAs that store indexes

If there are no replica RDAREAs, 0.

n: Total number of indexes to be defined

V: Segment size of user RDAREA used to store index (pages)

W: Number of index storage pages

For details, see *16.1.3 Calculating the number of index storage pages*.

If `EMPTY` is specified by `CREATE INDEX, 0`.

X: Page length of user RDAREA used to store index (bytes)

17.1.4 Amount of system log information output during table data updating

When you manipulate rows in tables, the system logs shown below are output.

For a HiRDB single server configuration, add the amount of system log information calculated here to the amount of log information output by a single server. For a HiRDB parallel server configuration, add the amount of system log information calculated here to, respectively, the amounts of log information output by the back-end server and dictionary server that manage the RDAREAs that store the tables and indexes being updated.

Table 17–1: Types of system log information that are output when table rows are manipulated

Type of system log information	Description
Base row log information	This log information is output when a table's row data is added, deleted, or updated.
Branch row log information	<p>This log information is output when row data is manipulated in columns with the following data types:</p> <ul style="list-style-type: none"> • <code>VARCHAR</code>^{#1} • <code>NVARCHAR</code>^{#2} • <code>MVARCHAR</code>^{#1} • Repetition columns • Abstract data type • <code>BINARY</code> type^{#3}
Index log information	This log information is output when index keys are added, deleted, or updated. Determine the amount of index log information on the basis of the type of database manipulation (<code>INSERT</code> , <code>DELETE</code> , or <code>UPDATE</code> statement), as shown in <i>Table 17-2 Amount of log information depending on type of database manipulation</i> .
Event log	This log information is output when HiRDB Datareplicator is used or when row data containing repetition columns is added, deleted, or updated.

#1: Branch row log information is output if either one of the following conditions is satisfied:

- The no-split option is not specified and the actual data length is 256 bytes or greater.
- The no-split option is specified and the actual total length of data per row exceeds the page length.

#2: Branch row log information is output if either one of the following conditions is satisfied:

- The no-split option is not specified and the actual data length is 128 bytes or greater.
- The no-split option is specified and the actual total length of data per row exceeds the page length.

#3: If the actual total length of data per row exceeds the page length, branch row log information is output.

Table 17–2: Amount of log information depending on type of database manipulation

Type of data manipulation (SQL statement)	Amount of log information
Key addition (<code>INSERT</code> statement)	Amount of addition log information
Key deletion (<code>DELETE</code> statement)	Amount of deletion log information
Key updating (<code>UPDATE</code> statement)	Amount of deletion log information for the key before updating + amount of addition log information for the key after updating

The amount of system log information that is output when a database is updated (INSERT, DELETE, or UPDATE) can be obtained from Formulas 1 and 2 as follows, depending on the type of operation (INSERT, DELETE, or UPDATE).

The amount of system log information that is output during UAP execution that does collect log is 460 bytes, which is based on the fact that segment allocations occur.

Condition	Formula (bytes)
Amount of system log information that is output to add (INSERT) or delete (DELETE) n rows in a table	$(a + b + c) \times n$
Amount of system log information that is output to update (UPDATE) n rows in a table	$(a^{\#1} + d^{\#2} + e^{\#3}) \times n$

a : Amount of base row log information (bytes)

b : Total amount of all branch row log information (bytes)

c : Total amount of all index log information (bytes)

d : Total amount of all branch row log information subject to update processing (bytes)

e : Total amount of all index log information subject to update processing (bytes)

n : Number of rows manipulated

#1: This value is added when the column value to be updated (UPDATE) is in the base row.

#2: This value is added when the column value to be updated (UPDATE) is in the branch row.

#3: This value is added when an index is defined for the column being updated (UPDATE).

(1) Determining the amount of base row log information

The following table lists the formulas for determining the amount of base row log information per data item.

Table 17–3: Formulas for determining the amount of base row log information per data item

Type of data manipulation (SQL statement)		Amount of log information output (bytes)
Data addition (INSERT statement)		$k + 152$
Data deletion (DELETE statement)		
Data updating (UPDATE statement)	FIX table with $f \leq 12$	f $\sum_{i=1} d_i +$ f $\sum_{j=1} d_j + 4 \times f + 152$
	Non-FIX table or $f > 12$	$k_1 + k_2 + 160$ ■ If LOCK is specified in the pd_nowait_scan_option operand, add: 314×2
	Row interface used	$2 \times k_1 + 160$

k : Length of row to be added or deleted

k_1 : Length of row before updating

k_2 : Length of row after updating

f : Number of columns to be updated

d_i : Data length of column before updating

d_j : Data length of column after updating

Note 1

The values of k , k_1 , and k_2 depend on the specification of `FIX`, as shown as follows.

- *Table 16-1 List of data lengths*
- *Table 16-2 Data lengths for the variable-length character string type (except abstract data type and repetition columns)*
- *Table 16-3 Data lengths for the variable-length character string type (abstract data type)*
- *Table 16-4 Data lengths for the variable-length character string type (repetition columns)*
- `FIX` specified
Total data length of all columns in table + 4.
- `FIX` not specified
Total data length of all columns in table + 6 + 2 × total number of columns in table.

Note 2

If HiRDB Datareplicator is being used, or if updatable online reorganization is executing, then the same amount will be output to the log file for updating 12 or fewer columns of the `FIX` table as for updating 13 or more columns.

Note 3

When a `BLOB` column is defined in a table, fix the row length of *Table 17-3 Formulas for determining the amount of base row log information per data item* at nine bytes for `BLOB` columns, and add the amounts of log information shown in the following table.

Table 17-4: Formulas for determining the amount of log information per `BLOB` column data item

Type of data manipulation (SQL statement)	Specification of recovery	Amount of log information output (bytes)
Data addition (<code>INSERT</code> statement)	Not specified or <code>partial</code> specified	$604 + 180 \times p5$
	All specified	$2,348 + p1 + 8,340 \times p2$ $+ (148 + lt) \times p3$
	Not specified	300
Data deletion (<code>DELETE</code> statement)	Not specified or <code>partial</code> specified	$460 + 180 \times p6$
	All specified	
	Not specified	468
Data updating (<code>UPDATE</code> statement)	Not specified or <code>partial</code> specified	$604 + 180 \times p5 + 460 + 180 \times p6$
	All specified	$2,496 + p1 + 8,334 \times p2$ $+ (148 + lt) \times p3$
	Not specified	312
Data concatenation operation (<code>UPDATE</code> statement) $Bb \leq 7,168$	Not specified or <code>partial</code> specified	2,344
	All specified	$8,340 \times a + 1,600 + d + 8,340 \times p4 + (148 + lt2) \times p3$
	Not specified	428
Data concatenation operation (<code>UPDATE</code> statement) $Bb > 7,168$	Not specified or <code>partial</code> specified	2,772
	All specified	$2,772 + Ba + 8,340 \times p4 + (148 + lt2) \times p3$
	Not specified	428
Backward deletion/updating of data (<code>UPDATE</code> statement) $Bb \leq 7,168$	Not specified or <code>partial</code> specified	2,344
	All	9,512
	No	428

Type of data manipulation (SQL statement)	Specification of recovery	Amount of log information output (bytes)
Backward deletion/updating of data (UPDATE statement) $Bb > 7,168$	Not specified or partial specified	$2,492 + 180 \times \uparrow (Bb - Bd) \div 8,192 \uparrow$
	All	$2,492 + 180 \times \uparrow (Bb - Bd) \div 8,192 \uparrow$
	No	$428 + 180 \times \uparrow (Bb - Bd) \div 8,192 \uparrow$

Bi : BLOB data length (bytes)

Ba : One of the following values:

- If $Bb > 7,168$: $8,192 - \{(Bb - 7,168) - \downarrow (Bb - 7,168) \div 8,192 \downarrow \times 8,192\}$
- If $Bb \leq 7,168$: 0

Bb : BLOB data length before update (bytes)

Bc : BLOB added data length (bytes)

Bd : Data length after update (value specified by value expression 3 of SUBSTR function) (bytes)

lt : One of the following values:

- If $Bi > 7,168$: $Bi - 7,168 - \downarrow (Bi - 7,168) \div 8,192 \downarrow \times 8,192$
- If $Bi \leq 7,168$: 0

$lt2$: One of the following values:

- If $Bc + Bb > 7,168$: $(Bc + Bb - Ba - 7,168) - \downarrow (Bc + Bb - Ba - 7,168) \div 8,192 \downarrow \times 8,192$
- If $Bc + Bb \leq 7,168$: 0

$p1$: One of the following values:

- If $Bi > 7,168$: 7,168
- If $Bi \leq 7,168$: Bi

$p2$: One of the following values:

- If $Bi > 7,168$: $\downarrow (Bi - 7,168) \div 8,192 \downarrow$
- If $Bi \leq 7,168$: 0

$p3$: One of the following values:

- If $lt = 0$ or $lt2 = 0$: 0
- If $lt > 0$ or $lt2 > 0$: 1

$p4$: One of the following values:

- If $Bc + Bb > 7,168$: $\downarrow (Bc + Bb - Ba - 7,168) \div 8,192 \downarrow$
- If $Bc + Bb \leq 7,168$: 0

$p5$: One of the following values

- If $Bi > 0$: $\uparrow (Bi + 1,024) \div 8,192 \uparrow$
- If $Bi = 0$: 0

$p6$: One of the following values

- If $Bb > 0$: $\uparrow (Bb + 1,024) \div 8,192 \uparrow$
- If $Bb = 0$: 0

a : One of the following values:

- If $Bb > 0$: 1
- If $Bb = 0$: 0

d : One of the following values:

- If $Bc + Bb \leq 7,168$: $Bc + Bb$
- If $Bc + Bb > 7,168$: 7,168

(2) Determining the amount of branch row log information

- (a) Branch row log information for VARCHAR, NVARCHAR, and MVARCHAR, when the no-split option is not specified

Calculate the amount of branch row log information that will be generated. The following table lists the formulas for determining the amount of log information per branch row.

Table 17–5: Formulas for determining the amount of log information per branch row (1)

Type of data manipulation (SQL statement)		Amount of log information output (bytes)
Data addition (INSERT statement)		$k + 152$
Data deletion (DELETE statement)		
Data updating (UPDATE statement)	Creating a new branch row by update processing	$k_2 + 160$
	Updating a branch row	$k_1 + k_2 + 160$
	Deleting a branch row by update processing	$k_1 + 160$

k : Length of one branch row to be added or deleted

k_1 : Length of one branch row before updating

k_2 : Length of one branch row after updating

Note

The following formula is used to obtain the row lengths k , k_1 , and k_2 :

$8 + \text{MIN}(\text{average length of actual data, page length of RDAREA} - 48)$

- (b) Branch row log information for abstract data columns, repetition columns, BINARY columns, and VARCHAR, NVARCHAR, and MVARCHAR when the no-split option is specified

Calculate the amount of branch row log information that will be generated. The table below lists the formulas for determining the amount of log information per branch row.

If there is more than one table storage RDAREA and the page length varies from one RDAREA to another, obtain the amount of branch row log information for each RDAREA with the same page length, then use their sum as the amount of branch row log information.

Table 17–6: Formulas for determining the amount of log information per branch row (2)

Type of data manipulation (SQL statement)	Amount of log information output (bytes)
Data addition (INSERT statement)	$SPN \times (b + 152)$
Data deletion (DELETE statement)	
Data updating (UPDATE statement)	$SPN \times (b + 160)$
Data concatenation operation (UPDATE statement) ^{#1}	$(b + 160) + (SPN - 1) \times (b + 152)$ ■ If the <code>pd_rpl_hdepath</code> operand is specified, add: 160
Backward deletion/updating of data (UPDATE statement) ^{#1, #2}	$(b \times 2 + 160) + (\downarrow(c - d) \div (b - 57) \downarrow) \times (b + 152)$ ■ If you specify the <code>pd_rpl_hdepath</code> operand, add: 160

#1

For BINARY type columns only

#2

Not applicable to compressed columns. For backward deletion/updating of data in compressed columns (UPDATE statement), use data updating (UPDATE statement).

b: Page length of an RDAREA

c: Data length before update

d: Data length after update (value specified by value expression 3 of SUBSTR function)

SPN: The following shows how to obtain this value:

If there is a branch row (for the branch condition, see #5 in *Table 16-1 List of data lengths*), obtain the value of *SPN* for all columns that constitute the table for the INSERT and DELETE statements and the value for the columns subject to updating for the UPDATE statement. However, for concatenation operations on BINARY columns, calculate *d_i* as the length of data to be added.

$$SPN = SPN1 + SPN2$$

For details about *SPN1* and *SPN2*, see (3) *Variables used in formulas* in 16.1.2 *Calculating the number of table storage pages*. Make sure, however, to use 1 for *a*.

(c) Additional log information output with updatable online reorganization

To determine the amount of log information that is output when a UAP accesses an RDAREA for the purpose of updatable online reorganization (while accessing subordinate RDAREAs that are in online reorganization hold), you must add the amount of event log information for the `pdorend` application process to the amount of log information that is ordinarily output. This section explains the additional log information.

- During updatable online reorganization, each time a row is updated, a row update start event and a row update completion event are output to the log. In addition, in the event of rollback, for each row recovered a row recovery start event and a row recovery completion event are output to the log. Therefore, when you determine system log file capacity, add the amounts of log information shown in the following table to the formulas of *Table 17-3 Formulas for determining the amount of base row log information per data item*.

Table 17-7: System log information added per item of data with updatable online reorganization

Type of data manipulation (SQL statement)	Additional log information output (bytes)
INSERT	320
DELETE	
UPDATE	$320 + c^{#1} + 160^{#2}$
INSERT rollback	320
DELETE rollback	
UPDATE rollback	$496 + \text{data length before row update} + c^{#1} + 160^{#2}$

#1

c is added when the `-e` option of the commit a database for online reorganization command (`pdorbegin`) is specified. The formula for *c* is as follows:

$$(\uparrow(\uparrow(a - b + 1) \div 8 \uparrow + 6) \div 4 \uparrow) \times 4 \times 2$$

a: ID of the largest column of the columns to be updated

b: ID of the smallest column of the columns to be updated

#2

Add this amount of log information when you perform a concatenation operation or a backward deletion/update on a BINARY type column with a defined length of 32,001 or more.

- If a BLOB column is defined for a table during updatable online reorganization, a BLOB update event log is added. Therefore, when you determine system log file capacity, add the amounts of log information shown in the following table to the formulas of *Table 17-4 Formulas for determining the amount of log information per BLOB column data item*.

Table 17–8: System log information added per item of BLOB data with updatable online reorganization

Type of data manipulation (SQL statement)	Additional log information output (bytes)
INSERT	0
DELETE	224
UPDATE	$224 + 160^{#2}$
INSERT rollback	0
DELETE rollback	$148 + p1 + 8,340 \times p2 + (148 + lt) \times p3^{#1}$
UPDATE rollback	$148 + p1 + 8,340 \times p2 + (148 + lt) \times p3^{#1} + 160^{#2}$

#1

For details about variables $p1$, $p2$, $p3$, and lt , see *Table 17-4 Formulas for determining the amount of log information per BLOB column data item*. In these variables, substitute the length of the BLOB data to be deleted.

#2

Add this amount of log information when you perform a concatenation operation or a backward deletion/update.

- During updatable online reorganization, a new repetition column update event log is output in the cases listed below. Therefore, when determining system log file capacity, add the amounts of log information shown in the table below to the formulas of *Table 17-3 Formulas for determining the amount of base row log information per data item*.
 - When a row containing repetition columns is deleted
 - When an entire repetition column is specified for update processing
 - When elements in a repetition column are specified for update processing
 - When elements in a repetition column are specified for deletion processing
 - When specified elements in a repetition column are specified for update processing

Table 17–9: System log information added per item of data that includes a repetition column with updatable online reorganization

Type of data manipulation (SQL statement)	Additional log information output (bytes)
UPDATE-SET (column specification)	164
UPDATE-ADD (element specification) (when new elements are added by updating)	N $\sum_{i=1} 164$
UPDATE-DELETE (element specification) (when new elements are deleted by updating)	M $\sum_{i=1} (p4 + 160)$
UPDATE-SET (element specification) (when only the specified elements are updated)	N $\sum_{i=1} ((p4 + 160) \times 2)$
DELETE	156
UPDATE-SET (column specification) rollback	156
Rollback of UPDATE-ADD (element specification) (when new elements are deleted by updating)	N $\sum_{i=1} (p4 + 180)$
Rollback of UPDATE-DELETE (element specification)	M

Type of data manipulation (SQL statement)	Additional log information output (bytes)
(when new elements are added by updating)	$\sum_{i=1} 184$
Rollback of UPDATE-SET (element specification) (when only the specified elements are updated)	N $\sum_{i=1} (p4 + 180)$
DELETE rollback	156

n : Number of repetition columns to be updated

m : Number of elements to be deleted

$p4$: $\uparrow \{ \uparrow$ (Average of the largest subscript number that is specified - average of the smallest subscript number that is specified + 1) $\div 8 \uparrow \} \div 4 \uparrow \times 4$

(3) Determining the amount of index log information

Use the formulas shown in the following table to determine the amount of index log information that is output for each row operated on in an index.

Table 17–10: Amount of index log information per index

Type of data manipulation (SQL statement)			Amount of log information output (bytes)
Key addition (INSERT statement)	Adding a new key		$k_I + 156$ or $(k_I + 156) \times 2^\#$
	Adding the same key as for existing row	$d \leq 200$	$k_I + 156$
		$d > 200$	$k_I + 292$
Key deletion (DELETE statement)	Deleting a key value		$k_2 + 156$
	Deleting the same key value as for existing row	$d \leq 200$	$k_2 + 156$
		$d > 200$	$k_2 + 292$
Key updating (UPDATE statement)			Amount of log information for key deletion + amount of log information for key addition

d : Number of duplicated key values

k_1 : Length of key to be added (bytes)

k_2 : Length of key to be deleted (bytes)

Note

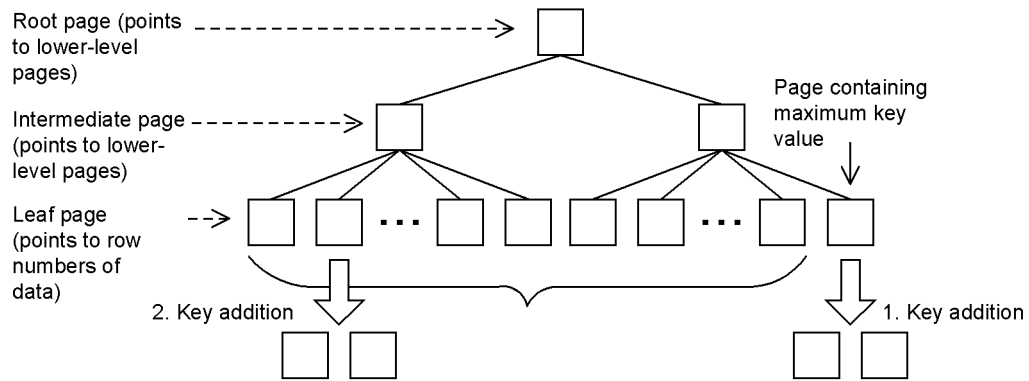
Length of key refers to the database storage key length. For details about how to determine the key length, see 16.1.3 Calculating the number of index storage pages.

$\#$: Use this formula for indexes with UNIQUE specified using the index key no-lock option.

(a) Determining the amount of index log information for index page splitting

The following figure illustrates the concept of index page splitting.

Figure 17–1: Concept of index page splitting



Explanation:

1. Splitting a page in two by adding a key to the rightmost leaf page (containing the maximum key value in the figure) is called page splitting containing the maximum key value.
2. Splitting a page in two by adding a key to any other leaf page is called page splitting without the maximum key value.

When a page that stores an index is split, HiRDB uses one of the two methods explained below to store the key value.

- **Page splitting without the maximum key value**

When a key value is added or deleted, HiRDB stores the key value by splitting the key and the unused area at a ratio of approximately 50:50. Page splitting without the maximum key value occurs in the following cases:

- The index storage page is too small to store the key value.
- a rows with the same key value are added while there are more than 200 duplicated key values (the value of a can be obtained from the formula shown below; in this case, page splitting occurs at every a rows).

Formula

$$a = \lceil \text{Page length of RDAREA for storing index (bytes)} \div 4 \rceil$$

- **Page splitting containing the maximum key value**

If a key value is added or updated in an index storage page containing the maximum key value, HiRDB stores the key value by splitting the key and the unused area at the ratio specified by the `PCTFREE` operand of `CREATE INDEX`.

For example, if `PCTFREE = 30` is specified, HiRDB stores the key value by splitting the key and unused area at a ratio of approximately 70:30.

Page splitting containing the maximum key value occurs if it is impossible to acquire as much free space as specified in the `PCTFREE` operand of `CREATE INDEX` when a key value is to be added. However, this does not apply to an upper-level page.

The following table lists the amount of index log information output per instance by split type.

Table 17–11: Amount of index log information by split type

Split type	Condition			Amount of log information output (bytes)
Page splitting containing the maximum key value	Adding a key value that is different from any key value already in the index	There is enough unused area to add the key in upper-level page	$2 \times k_l + a + 8 \times (m + 1) \leq 31,516$	$2 \times k_l + 472 + a + 8 \times (m + 1)$
			$2 \times k_l + a + 8 \times (m + 1) > 31,516$	$2 \times k_l + 632 + a + 8 \times (m + 1)$
		There is not enough unused area to add the key in upper-level page	$2 \times k_l + a + 8 \times (m + 1) \leq 31,516$	$n-1 \sum_{i=2} (288 + a)$

Split type	Condition			Amount of log information output (bytes)
				$+ 2 \times k_I + 472 + a +$ $8 \times (m + 1)$
			$2 \times k_I + a + 8 \times (m + 1) > 31,516$	$n-1$ $\sum (288 + a)$ $i=2$ $+ 2 \times k_I + 628 + a +$ $8 \times (m + 1)$
	Adding the same key value as one already in the index	$d_I \leq 200$	There is enough unused area to add the key in upper-level page	$2 \times k_I + 472 + a + 8$ $\times (m + 1)$
			There is not enough unused area to add the key in upper-level page	$n-1$ $\sum (288 + a)$ $i=2$ $+ 2 \times k_I + 472 + a +$ $8 \times (m + 1)$
		$d_I > 200$	There is enough unused area to add the key in lower-level page	$k_I + 472 + a$
			There is not enough unused area to add the key in lower-level page	$k_I + 462 + 2 \times a$
Page splitting without the maximum key value	There is not enough unused area to add the key	There is enough unused area to add the key in upper-level page	$2 \times k_I + a + 8 \times (m + 1) \leq 31,516$	$2 \times k_I + 332 + a + 8$ $\times (m + 1)$
			$2 \times k_I + a + 8 \times (m + 1) > 31,516$	$2 \times k_I + 492 + a + 8$ $\times (m + 1)$
		There is not enough unused area to add the key in upper-level page	$2 \times k_I + a + 8 \times (m + 1) \leq 31,516$	$n-1$ $\sum (288 + a)$ $i=2$ $+ 2 \times k_I + 332 + a +$ $8 \times (m + 1)$
			$2 \times k_I + a + 8 \times (m + 1) > 31,516$	$n-1$ $\sum (288 + a)$ $i=2$ $+ 2 \times k_I + 492 + a +$ $8 \times (m + 1)$

a : Page length of RDAREA storing the index (bytes)

d_I : Number of duplicated key values

k_I : Length of key value to be added (bytes)

Length of key refers to the database storage key length. For details about how to determine the key length, see 16.1.3 Calculating the number of index storage pages.

m : Number of index levels where split occurred

n : Number of upper page levels affected by splitting

If an upper-level page affected by leaf page splitting is also split, the value of n is 3 ($n \geq 3$).

Note

These formulas are used to estimate the amount of update log information for each row and index part. The derived value does not include the amount of log information related to system management that is output when a new page or segment is allocated during addition or update processing. Therefore, if you are handling large amounts of data, you must add the amounts of log information shown in the following table to determine the amount.

Table 17–12: Amount of log information for page allocation and segment allocation

Condition	Amount of log information output (bytes)
Allocation of a new page for storing rows resulting from data addition (INSERT) or updating (UPDATE)	440
Index page splitting resulting from data addition (INSERT) or updating (UPDATE)	$544 \times n + 272$
Segment allocation resulting from the above page allocation (each time as many pages are allocated as the segment size)	1,940

n

Number of index levels when page splitting occurred

(4) Determining the size of event log information

The event log information is output when row data containing repetition columns is added, deleted, or updated using HiRDB Datareplicator. The following table shows the amount of event log information that is output when a single row is manipulated.

Table 17–13: Amount of event log information that is output when a single row is manipulated

Type of data manipulation	Amount of event log information (bytes)
Data addition (INSERT statement)	$156 \times n$
Data updating (UPDATE statement)	New elements added by updating (UPDATE ADD)
	$164 \times n$
	Elements deleted by updating (UPDATE DELETE)
	$n \sum_{i=1} (p5 + 160)_i$
	Only specified elements updated (UPDATE SET)
	$n \sum_{i=1} (p5 + 160)$
	Specified repetition columns updated (UPDATE SET)
	$164 \times n$

n : Number of repetition columns being updated

$p5$: $\uparrow \{ \uparrow (\text{average of the largest subscript number that is specified} - \text{average of the smallest subscript number that is specified} + 1) \div 8 \uparrow \} \div 4 \uparrow \times 4$

17.1.5 Amount of system log information output during database creation by a utility

When you execute the following utilities, the system outputs the system log information shown in Table 17-14
Formulas for determining the amount of system log information output during database creation by a utility.

- Database load utility (pdload command)
- Database reorganization utility (pdreorg command)
- Rebalancing facility (pdrbal command)

The system log size is calculated by adding the value shown in *Table 17-15 Values added when calculating the amount of system log information, and conditions required to perform this addition* to the value obtained in *Table 17-14 Formulas for determining the amount of system log information output during database creation by a utility*. For row-partitioned tables and indexes, the amount of system log information is calculated for each RDAREA in which tables and indexes are stored.

- For a HiRDB single server configuration:
Add the total amount of system log information calculated for each RDAREA to the amount of log information output by a single server.
- For a HiRDB parallel server configuration:
Add the total amount of system log information calculated for each RDAREA to the amount of log information of the back-end server that manages the tables and indexes to be processed. If you reorganize dictionary tables with the database reorganization utility (pdreorg), however, add the amount of log information output by the dictionary server.

Table 17–14: Formulas for determining the amount of system log information output during database creation by a utility

Number	Condition	Amount of system log information output (bytes)	
		a specified in -l option	p specified in -l option
1	Batch index creation performed (c specified in -i option)	$n \sum_{i=1} \{$ $[132 + \uparrow(100 - f) \div 100 \uparrow \times Xi]$ $\times Wi$ $+ 280 \times Wi + 1,940 \times \uparrow Wi \div Vi \uparrow$ $\}$ $+ 280 \times m + 1,940 \times \uparrow m \div s \uparrow + a \times r$	$n \sum_{i=1} \{$ $280 \times Wi + 1,940 \times \uparrow Wi \div Vi \uparrow$ $\}$ $+ 280 \times m + 1,940 \times \uparrow m \div s \uparrow + c \times r$
2	Batch index creation not performed (s specified in -i option)	$n \sum_{i=1} \{$ $280 \times Wi + 1,940 \times \uparrow Wi \div Vi \uparrow$ $+ b \times r + e \times d$ $\}$ $+ 280 \times m + 1,940 \times \uparrow m \div s \uparrow + a \times r$	$n \sum_{i=1} \{$ $280 \times Wi + 1,940 \times \uparrow Wi \div Vi \uparrow$ $+ b \times r + e \times d$ $\}$ $+ 280 \times m + 1,940 \times \uparrow m \div s \uparrow + c \times r$
3	No index is created (n or x is specified in the -i option)	$280 \times m + 1,940 \times \uparrow \div s \uparrow + a \times r$	$280 \times m + 1,940 \times \uparrow m \div s \uparrow + c \times r$
4	Index creation or regeneration performed (nothing loaded with pdload, or ixrc or ixor specified in pdreorg -k option)	Add the following amount to the system log: <ul style="list-style-type: none"> • <i>Table 17-13 Amount of system log information output during index definition</i> • <i>Table 17-12 Amount of log information for page allocation and segment allocation</i> • <i>Table 17-17 Amount of system log information output during execution of the PURGE TABLE statement</i> 	Add the following amount to the system log: <ul style="list-style-type: none"> • <i>Table 17-12 Amount of log information for page allocation and segment allocation</i> • <i>Table 17-17 Amount of system log information output during execution of the PURGE TABLE statement</i>

Note:

The amount of system log information when indexes are created in batch or singularly needs to be calculated for the number of indexes.

a: Amount of log information output when one data item is added, as obtained from *Table 17-3 Formulas for determining the amount of base row log information per data item* and *Table 17-4 Formulas for determining the amount of log information per BLOB column data item*.

b: Amount of log information output when one data item is added, as obtained from *Table 17-10 Amount of index log information per index*.

c: Amount of log information output when one data item is added, as obtained from *17-4 Formulas for determining the amount of log information per BLOB column data item*.

d: Amount of system log information output during index splitting

See *Table 17-11 Amount of index log information by split type* and *Table 17-12 Amount of log information for page allocation and segment allocation*.

e: Number of times index splitting occurs

f: Value (%) of the PCTFREE operand specified during index definition (percentage of unused space in a page)

m: Number of table storage pages

See *16.1.2 Calculating the number of table storage pages*.

n: Total number of indexes defined for table

r: Number of rows to be stored in table (rows)

s: Segment size of user RDAREA used to store table (pages)

V_i: Segment size of user RDAREA used to store index (pages)

W_i: Number of index storage pages

See *16.1.3 Calculating the number of index storage pages*.

X_i: Page length of user RDAREA used to store index (bytes).

For items 1 and 2 in *Table 17-14 Formulas for determining the amount of system log information output during database creation by a utility*, the following table lists the values that are added when calculating the amount of system log information, and the conditions required to perform this addition.

Table 17-15: Values added when calculating the amount of system log information, and conditions required to perform this addition

Conditions for addition	Values added to the amount of system log information
When LOB columns are defined	<p>Add the following amount of log information for (the number of LOB columns × the number of rows):</p> <ul style="list-style-type: none"> • See <i>Data addition (INSERT statement)</i> in <i>Table 17-4 Formulas for determining the amount of log information per BLOB column data item</i> • When the recovery attribute of the LOB column is not <code>recovery no</code> $3,544 \times (\text{LOB data length} \div 31,744)$ (bytes)
When the database load utility (<code>pdload</code>) is executed with the <code>-d</code> option specified, or the database reorganization utility (<code>pdorg</code>) is executed	Add the amount of log information of all indexes and all LOB columns (or LOB attributes) defined in <i>Tables</i> or <i>Rebalancing tables</i> in <i>Table 17-17 Amount of system log information output during execution of the PURGE TABLE statement</i> and target tables.
When a table having a LOB column or LOB attribute is reorganized using the database reorganization utility (<code>pdorg</code>), if the following two conditions are met:	<p>Add the following log amount of information for each LOB RDAREA configuration file:</p> $\sum \{17,000 \times \uparrow(\text{the number of HiRDB file segments} \div 64,000) \uparrow \times 95\}$ <p>Add the following amount of log information for each item (the number of LOB columns or LOB attributes × the number of rows).</p> <p>■ LOB column is <code>recovery all</code> specification and also <code>-l a</code> specification</p>

Conditions for addition	Values added to the amount of system log information
<ul style="list-style-type: none"> The table is reorganized without the <code>-j</code> option specified The LOB column does not specify <code>recovery no</code> and <code>n</code> is not specified for the <code>-l</code> option 	$\uparrow 17,600 \div sr \uparrow \times sr$ ■ Other than the above $\uparrow 3,200 \div sr \uparrow \times sr$ <i>sr</i> : Record length of the system log file specified in the <code>pd_log_rec_leng</code> operand.
The HiRDB Datareplicator linkage facility is used (the <code>pd_rpl_hdepath</code> operand is specified) and the table contains repetition columns	Add the log information shown in <i>Table 17-13 Amount of event log information that is output when a single row is manipulated</i> for each added row.

17.1.6 Amount of system log information that is output depending on the SQL manipulation

If `-k cnc` is specified in the `pdhibegin` operand of the system common definition, system log information is output when `CONNECT`, `DISCONNECT`, or `set session authorization` is executed.

You can use the formula shown below to calculate the amount of system log information that is output depending on the SQL manipulation. For a HiRDB single server configuration, add this amount to the amount of log information output by a single server; for a HiRDB parallel server configuration, add it to the amount of log information output by the front-end server.

Formula

Amount of system log information (bytes)

$= 568 \times (\text{CONNECTs count} + \text{set session authorization executions count})$

17.1.7 Amount of system log information that is output during the definition of extended system-defined scalar functions

The amount of system log information that is output when you use the `pdextfunc` command to define an extended system-defined scalar function is shown below. For a HiRDB single server configuration, add this amount to the amount of log information output by a single server. For a HiRDB parallel server configuration, add it to the amount of log information output by the dictionary server.

Amount of system log information (bytes) = 233,529

17.1.8 Amount of system log information that is output during the execution of the RDAREA automatic extension facility

When you use the RDAREA automatic extension facility, the system outputs log information during the execution of automatic extension.

The table below shows formulas for finding the amount of system log information that is output. For a HiRDB single server configuration, add this amount of log information to the amount of log information output by a single server; for a HiRDB parallel server configuration, add it to the amount of log information output by the back-end server and dictionary server that manage the applicable RDAREA.

Table 17–16: Amount of system log information that is output during the execution of the RDAREA automatic extension facility

Type of RDAREA	Amount of system log information (bytes)
LOB RDAREA	1,956

Type of RDAREA	Amount of system log information (bytes)
Other than LOB RDAREA	$1,372 + (144 + p) \times 2$

p: Page size of RDAREA subject to automatic extension

17.1.9 Amount of system log information output when the PURGE TABLE statement is executed

The amount of log information output when the PURGE TABLE statement is executed is determined from the total log information calculated for the table from all indexes, LOB columns, and LOB attributes. For partitioning tables and partitioning indexes, determine the total amount of log information for each RDAREA.

For a HiRDB single server configuration, add the value found to the amount of log information to be output by a single server. For a HiRDB parallel server configuration, add it to the amount of log information of the back-end server that manages the RDAREA in which the applicable table is stored (including the RDAREAs in which the index, LOB column and LOB attributes defined in the table are stored).

The following table shows formulas for determining the amount of system log information that is output when a PURGE TABLE statement is executed. Find the variables *A*, *B* and *C* in the formulas for each RDAREA constituent file.

Table 17–17: Amount of system log information output during execution of the PURGE TABLE statement

Type	Amount of system log information (bytes)
Tables	$1,000 + 1,100 \times \text{number of allocated segments}$
Rebalancing tables	$1,000 + 1,100 \times \text{number of allocated segments} + 400 \times \uparrow 1,024 \div \text{number of partitioning RDAREAs} \uparrow$
Indexes [#]	$1,000 + 1,100 \times \text{number of allocated segments}$
LOB columns or LOB attributes [#]	$1,000 + 17,000 \times A + 160 \times B + 2 \times C$

A: $\uparrow (\text{number of segments used by HiRDB files} \div 64,000) \uparrow \times 95$

B: $\uparrow (\text{number of segments used by HiRDB files} \div 64,000) \uparrow$

C: $\uparrow (\text{number of segments used by HiRDB files} \div 64,000) \uparrow \times 8,150$

[#]: For plug-ins, the initialization log for each plug-in is output. For details, see the documentation for each plug-in.

17.1.10 Amount of system log information output when the free page release utility (pdreclaim) is executed

System log information is output when free pages or segments of tables and indexes are released by the free page release utility (pdreclaim).

The table below shows the amounts of log information that are output by combination of specified option and applicable resources. The item numbers under the description of the amount of log information correspond to the item numbers in Table 17-18 Amount of system log information output when the free page release utility is executed.

Option	Applicable resources	Amount of log information
None	Tables	Value of item (1)
	Indexes	Value of item (2)
-j	Tables	Value of item (3)
	Indexes	
-a	Tables	Sum of the values of items (1) and (4)

Option	Applicable resources	Amount of log information
	Indexes	Sum of the values of items (2) and (4)

The table below shows formulas for finding the amount of log information that is output. When tables and indexes are row-partitioned, you must calculate the amount of log information for each partitioned RDAREA. For a HiRDB single server configuration, find the total amount of log information calculated for each row-partitioned RDAREA. Add this to the amount of log information that a single server outputs. For a HiRDB parallel server configuration, find the amounts of log information calculated for each row-partitioned RDAREA for each server, and then total them. Add these totals respectively to the amounts of log information of the back-end servers and dictionary servers that manage the RDAREAs that store the applicable resources.

Table 17–18: Amount of system log information output when the free page release utility is executed

Item No.	Type		Amount of system log information (bytes)
(1)	Tables		$140 \times n$ ■ If the <code>-o</code> option is not specified: $+ 504 \times m$
(2)	Indexes	With no key whose key value duplication is 201 or more	$\{((k + 14) \div (j - 68)) \times 1,408 + k + 12 \times h + 908\} \times n + 304 \times q$ ■ If the <code>-x</code> option is specified, add: $+ \{m - n - (((k + 14) \times 2 \times m) \div (j - 68))\} \times (j + 76) \times 0.7$
		With a key whose key value duplication is 201 or more	$(724 + k) \times n + 285 \times \text{MAX}(n, 16 \times n \div (j - 78)) - 159 + 304 \times q$ ■ If the <code>-x</code> option is specified, add: $+ \{m - n - (((k + 14) \times 2 \times m) \div (j - 68))\} \times (j + 76) \times 0.7$
(3)	Segments	If the <code>-j</code> option is specified	$2,380 \times p$ ■ If <code>index</code> is specified for the <code>-k</code> option, add: $+ 304 \times p$
(4)		If the <code>-a</code> option is specified	$3,100 \times p$ ■ If <code>index</code> is specified for the <code>-k</code> option, add: $+ 304 \times p$

h: Index levels

j: Page size (bytes)

k: Index key length (bytes)

m: Number of used pages (excluding full pages)

n: Number of used free pages

p: Number of used free segments

Includes the number of segments in the process of being released.

q: Number of non-full segments

This value is obtained by subtracting the number of full segments from the number of used segments.

You can check the number of used segments and the number of full segments by using the database condition analysis utility (`pddbstat`).

17.1.11 Amount of system log information that is output during execution of the facility for predicting reorganization time

If you use the facility for predicting reorganization time, you must add the amount of log information obtained from the following formula to the amount of log information that is output by the dictionary server for a HiRDB parallel server configuration (or that is output by the single server for a HiRDB single server configuration):

Formula

Amount of system log information (bytes) =

$$n \times \{1,604 \times (A + B + C + D) \times (\uparrow E \uparrow + 1)\}$$

$$+ m \times \{872 \times (a + b + c + 1)\}$$

$$+ 11,680 \times \uparrow \{(A + B + C + D) \times (\uparrow E \uparrow + 1)\} \times 30 \div 540 \uparrow$$

$$+ 332 \times \{(A + B + C + D) \times (\uparrow E \uparrow + 1)\} \times 30$$

$$+ 7,760$$

A: Number of tables that have been created + 61

B: Number of indexes that have been created + 124

C: Total number of BLOB columns defined for the tables that have been created + 3

D: Total number of BLOB attributes defined for the tables that have been created

E: Average number of partitions in the table storage RDAREA

If the RDAREA is not partitioned, the value is 1. The average value is rounded up.

a: Number of RDAREAs storing tables processed by SQL statements or commands

b: Number of RDAREAs storing indexes processed by SQL statements or commands

c: Number of LOB RDAREAs storing tables processed by SQL statements or commands

n: Number of times the condition analysis result accumulation facility (pddbst -k logi -e) executed

m: Number of SQL statements or commands executed to update the database management table

For details about the SQL statements and commands for updating the database management table, see the *HiRDB Version 9 System Operation Guide*.

17.1.12 Amount of system log information output during an updatable backup hold

When the database is updated during an updatable backup hold, the additional amount of system log information shown by the following formula is output. For a HiRDB single server configuration, add the amount of system log information found here to the amount of log information output by a single server. For a HiRDB parallel server configuration, add the corresponding amounts of log information for the back-end servers and dictionary servers that manage the RDAREAs that are being updated.

However, when HiRDB terminates abnormally or is forcibly terminated during a backup hold, the backup hold is not inherited when HiRDB restarts (unless it is a reference-possible backup hold). For this reason, the system log information calculated here is not output.

Formula

$$\sum_{i=1}^a (S_i + 200) \times T_i$$

a: Number of RDAREAs updated during the updatable backup hold

S_i: Page size of RDAREAs (bytes)

T_i: Number of update pages of RDAREA updated during the updatable backup hold

Use the following procedure to find the number of RDAREA update pages.

Procedure

1. Execute the statistics analysis utility (HiRDB file statistical information pertaining to database manipulation) at the following points:
 - At the start of the updatable backup hold

- At the release of the updatable backup hold
2. Check the number of synchronization WRITES (SYNC-W) to find the number of update pages from that differential.

17.2 Determining the size of synchronization point dump files

(1) Determining the size of a synchronization point dump file

The following formula is used to determine the size of a synchronization point dump file.

Formula

Size of synchronization point dump file (bytes)

$$= \text{MAX}(12, \text{number of records in synchronization point dump file})^{\#1} \times 4,096^{\#2}$$

#1: See (2) as follows; the obtained value is specified in the `-n` option of the `pdloginit` command. If the obtained value is less than 12, specify 12 for the `-n` option of the `pdloginit` command.

#2: This is the record length of a synchronization point dump file.

(2) Determining the number of records in a synchronization point dump file

Condition		Formula for determining the number of records
HiRDB single server configuration		$\lceil (96 + 112 \times a) \div 4,096 \rceil + 3$
HiRDB parallel server configuration	Front-end server	4
	Back-end server	$\lceil (96 + 112 \times a) \div 4,096 \rceil + 3$
	Dictionary server	$\lceil (96 + 112 \times a) \div 4,096 \rceil + 3$

a: Value of the `pd_lck_until_disconnect_cnt` operand

17.3 Determining the size of status files

(1) Determining the size of a status file

The following formula is used to determine the size of a status file.

Formula

Size of status file (bytes) = $a \times b$

a : Number of records in the status file.

See (2) as follows; the obtained value is specified in the `-c` option of the `pdstsinit` command.

b : Record length of the status file. This value is specified in the `-l` option of the `pdstsinit` command.

(2) Determining the number of records in a status file

The following formula is used to determine the number of records in a status file.

Formula

Number of records

$$= \lceil \lceil S \div (\text{record length} - 40) \rceil + \lceil S \div 100 \rceil + S + 2 \rceil \times 1.2 \uparrow$$

See (3) as follows for the value of S .

(3) Determining the value of S

(a) HiRDB single server configuration

Type	Formula for S
Unit	$\begin{aligned} & \lceil (2,056 + 128 \times d) \div g \rceil + \lceil 2,512 \div g \rceil + \lceil 40 \div g \rceil \\ & + \lceil 308 \div g \rceil + \lceil 4,000 \div g \rceil + \lceil 43,536 \div g \rceil + \lceil 32 \div g \rceil \end{aligned}$
Server	$\begin{aligned} & \lceil 128 \div g \rceil + \lceil 3,280^{#1} \div g \rceil + \lceil (592^{#2} + j \times b) \div g \rceil \\ & + \lceil 8,192 \div g \rceil + \lceil (8,192 - 128) \div g \rceil \times \{ \lceil (a + m) \div k \rceil - 1 \} \\ & + \lceil \lceil \lceil (c \div 127 \uparrow + 1) \times 2,048 \rceil \div 8,192 \uparrow \times 8,192 \rceil \div g \rceil \\ & + \lceil 2,048 \div g \rceil + \lceil \{ 244 + 64 \times (2 \times h + 7) \} \div g \rceil \\ & + \lceil 96 \div g \rceil + \lceil \{ 48 + 68 \times (2 \times h + 7) \} \div g \rceil \\ & + \lceil [20,480 \times \{ 1 + \lceil ((152 \times e) - (16 \times f) - (80 \times A) + (8 \times B)) \div 20,476 \rceil + \lceil (4f + 4C) \div 20,480 \rceil \} \rceil \div g \rceil \\ & + \lceil (12 \times c + 4) \div g \rceil \\ & + w \\ & + \lceil (4 + 16 \times c) \div g \rceil \\ & \blacksquare \text{ If the status file record length} < 4,096, \text{ add:} \\ & + \lfloor \text{MAX}\{3,400 \div r + 0.7, 1\} \rfloor \times \text{MAX}(\lfloor 4,096 \div s \rfloor, 2) \times n \\ & + \lfloor 5,662,310 \div r + 0.7 \rfloor \times \text{MAX}(\lfloor 4,096 \div s \rfloor, 2) \times p \\ & \blacksquare \text{ If } 4,096 \leq \text{the status file record length} < 12,288, \text{ add:} \\ & + \lfloor \text{MAX}\{3,400 \div \lfloor (s - 348) \div 20 \rfloor + 0.7, 1\} \rfloor \times n \\ & + \lfloor (5,662,310 \div \lfloor (s - 348) \div 20 \rfloor) + 0.7 \rfloor \times p \\ & \blacksquare \text{ If } 12,288 \leq \text{the status file record length, add:} \\ & + \lfloor \text{MAX}\{3,400 \div \lfloor (s - 876) \div 20 \rfloor + 0.7, 1\} \rfloor \times n \end{aligned}$

Type	Formula for S
	$+ \downarrow (5,662,310 \div \downarrow (s - 876) \div 20 \downarrow) + 0.7 \downarrow \times p$ <p>■ If the <code>pd_log_auto_unload_path</code> is specified, add:</p> $+ \uparrow 20,848 \div g \uparrow$ <p>■ If you are using the rapid system switchover facility, add:</p> $+ \uparrow (\uparrow (256 \times i + 4,096) \div 4,096 \uparrow) \times 4,096 \div g \uparrow$ <p>■ If <code>Y</code> is specified for the <code>pd_dbbuff_modify</code> operand, add:</p> $+ v$ <p>■ If HiRDB Staticizer Option has finished setup and the <code>pd_max_reflect_process_count</code> operand is specified, add:</p> $+ \uparrow (128 + 704) \div g \uparrow$ $+ \uparrow 256 \div g \uparrow$ <p>■ If XDS is defined, add:</p> $+ \uparrow 32 \div g \uparrow$

a: Number of specified `pdlogadfg` operands

b: Number of specified `pdlogadfg -d spd` operands

c: Value of `pd_max_rdarea_no` operand

d: 108 (single server) or 74 (utility special unit)

e: Number of specified `pdbuffer` operands

f: Number of `-i` options specified in the `pdbuffer` operand

g: Record length of status file minus 40

h: Value of `pd_max_users` + value of `pd_max_reflect_process_count`

i: When the inner replica facility is used, the value of the `pd_max_file_no` operand

When the inner replica facility is not used, the value of the `pd_max_rdarea_no` operand

j: 736

k: 11

m: 1

n: If the number of RDAREAs in the server $\leq 3,400$, the value is 1.

If $3,401 \leq$ number of RDAREAs in the server $\leq 6,800$, the value is 2.

If $6,801 \leq$ number of RDAREAs in the server, the value is 3.

p: If the number of RDAREAs in the server $\leq 10,200$, the value is 0.

If $10,201 \leq$ number of RDAREAs in the server $\leq 5,672,510$, the value is 1.

If $5,672,511 \leq$ number of RDAREAs in the server $\leq 11,334,820$, the value is 2.

If $11,334,821 \leq$ number of RDAREAs in the server, the value is 3.

r: $\downarrow (s - 348) \div 20 \downarrow + \downarrow g \div 20 \downarrow \times (\text{MAX}(\downarrow 4,096 \div s \downarrow, 2) - 1)$

s: Status file record length

v: 32-bit mode: $\uparrow (24 + 28 \times (\times + 512) + 32 + 112 \times D) \div g \uparrow$

64-bit mode: $\uparrow (32 + 32 \times (\times + 512) + 32 + 144 \times D) \div g \uparrow$

w: 32-bit mode: $\uparrow ((12 + (\uparrow \uparrow (c \div 8) \uparrow \div 4 \uparrow) \times 4) \times z) \div g \uparrow$

64-bit mode: $\uparrow ((12 + (\uparrow \uparrow (c \div 8) \uparrow \div 8 \uparrow) \times 8) \times z) \div g \uparrow$

x: Value of `pd_max_add_dbbuff_shm_no` operand

y: Value of `pd_max_add_dbbuff_no` operand

z : Value obtained by adding the `pd_max_add_dbbuff_no` operand value to the total number of `-i` options specified in the `pdbuffer` operand (if a value other than Y is specified in the `pd_dbbuff_modify` operand, 0 is used)

A : With the `-o` option specified for the `pdbuffer` operand of the system common definition: 1

Without the `-o` option specified for the `pdbuffer` operand of the system common definition: 0

B : Total number of RDAREAs specified in the `-r` and `-b` options to the `pdbuffer` operand of the system common definition

C : Total number of RDAREAs containing indexes specified in the `-i` option to the `pdbuffer` operand of the system common definition

D : If Y is specified for the `pd_dbbuff_modify` operand of the system common definition:

- If the `pd_max_add_dbbuff_no` operand of the system common definition is specified: $e + y$
- If the `pd_max_add_dbbuff_no` operand of the system common definition is omitted: 1,000

If N is specified for the `pd_dbbuff_modify` operand of the system common definition: e

#1: For the 64-bit mode, use a value of 3,456.

#2: For the 64-bit mode, use a value of 688.

(b) HiRDB parallel server configuration

Type	Formula for S
Unit	$\begin{aligned} & \uparrow [2,056 + 128 \times \{14 \times (p - P + q + N) + (p - P + q + r) \\ & + (25 + 15 \times (p - P) + 7 \times q + 3 \times r) + (s \times 2 + 1) + (N \times 16) \\ & + (38 + 4 \times (p - P + q + r + N) + d + z + 3 \times (p - P + q + r + N))\}] \div j \uparrow \\ & + \uparrow 944 \div j \uparrow + \uparrow 4,816 \div j \uparrow + \uparrow 40 \div j \uparrow \\ & + \uparrow 308 \div j \uparrow + \uparrow 4,000 \div j \uparrow + \uparrow 43,536 \div j \uparrow \\ & + \uparrow (16 + 8 \times M) \div j \uparrow + \uparrow 16 \div j \uparrow \\ & \blacksquare \text{ If the unit has a system manager, add:} \\ & + \uparrow (L + 16 \times M) \div j \uparrow \\ & \blacksquare \text{ If you are not using the standby-less system switchover (effects distributed) facility, add:} \\ & + \uparrow 32 \div j \uparrow \\ & \blacksquare \text{ If you are using the standby-less system switchover (effects distributed) facility, add:} \\ & + \uparrow [20,480 \times \{1 + \uparrow ((152 \times f) - (16 \times g) - (80 \times Q) + (8 \times R)) \div 20,476 \uparrow + \uparrow (4g + 4S) \div 20,480 \uparrow\}] \div j \uparrow \end{aligned}$
Front-end server	$\begin{aligned} & \uparrow 128 \div j \uparrow + \uparrow 3,280^{\#1} \div j \uparrow + \uparrow (592^{\#2} + A \times b) \div j \uparrow \\ & + \uparrow 8,192 \div j \uparrow + \uparrow (8,192 - 128) \div j \uparrow \times \{\uparrow (a + C) \div B \uparrow - 1\} \\ & + \uparrow 8,192 \div j \uparrow + \uparrow 2,048 \div j \uparrow \\ & + \uparrow \{244 + 64 \times (2 \times e + 7)\} \div j \uparrow + \uparrow 96 \div j \uparrow + \uparrow 32 \div j \uparrow \\ & \blacksquare \text{ If the } \texttt{pd_log_auto_unload_path} \text{ operand is specified, add:} \\ & + \uparrow 20,848 \div j \uparrow \end{aligned}$
Dictionary server	$\begin{aligned} & \uparrow 128 \div j \uparrow + \uparrow 3,280^{\#1} \div j \uparrow + \uparrow (592^{\#2} + A \times b) \div j \uparrow \\ & + \uparrow 8,192 \div j \uparrow + \uparrow 8,192 \div j \uparrow \times \{\uparrow (a + C) \div B \uparrow - 1\} \\ & + \uparrow [\uparrow \{(\uparrow c \div 127 \uparrow + 1) \times 2,048\} \div 8,192 \uparrow \times 8,192] \div j \uparrow \\ & + \uparrow 2,048 \div j \uparrow + \uparrow \{244 + 64 \times (2 \times h + 7)\} \div j \uparrow + \uparrow 96 \div j \uparrow \\ & + \uparrow \{48 + 68 \times (2 \times h + 7)\} \div j \uparrow \\ & + \uparrow [20,480 \times \{1 + \uparrow ((152 \times f) - (16 \times g) - (80 \times Q) + (8 \times R)) \\ & \div 20,476 \uparrow + \uparrow (4g + 4S) \div 20,480 \uparrow\}] \div j \uparrow \\ & + \uparrow (12 \times c + 4) \div j \uparrow + \uparrow 32 \div j \uparrow \end{aligned}$

Type	Formula for S
	$+ H$ $+ \uparrow (4 + 16 \times c) \div j \uparrow$ <p>■ If the status file record length < 4,096, add:</p> $+ \downarrow \text{MAX}\{3,400 \div D + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4,096 \div n \downarrow, 2)$ <p>■ If $4,096 \leq$ the status file record length < 12,288, add:</p> $+ \downarrow \text{MAX}\{(3,400 \div \downarrow (n - 348) \div 20 \downarrow) + 0.7, 1\} \downarrow$ <p>■ If $12,288 \leq$ the status file record length, add:</p> $+ \downarrow \text{MAX}\{(3,400 \div \downarrow (n - 876) \div 20 \downarrow) + 0.7, 1\} \downarrow$ <p>■ If you are using the system switchover facility, add:</p> $+ \uparrow ((\uparrow (256 \times c + 4,096) \div 4,096 \uparrow \times 4,096) \div j \uparrow$ <p>■ If the <code>pd_log_auto_unload_path</code> operand is specified, add:</p> $+ \uparrow 20,848 \div j \uparrow$ <p>■ If you specify Y for the <code>pd_dbbuff_modify</code> operand, add:</p> $+ G$
Back-end server	$\uparrow 128 \div j \uparrow + \uparrow 3,280^{\#1} \div j \uparrow + \uparrow (592^{\#2} + A \times b) \div j \uparrow$ $+ \uparrow 8,192 \div j \uparrow + \uparrow 8,192 \div j \uparrow \times \{\uparrow (a + C) \div B \uparrow - 1\}$ $+ \uparrow [\uparrow \{(\uparrow c \div 127 \uparrow + 1) \times 2,048\} \div 8,192 \uparrow \times 8,192] \div j \uparrow$ $+ \uparrow 2,048 \div j \uparrow + \uparrow \{244 + 64 \times (2 \times i + 7)\} \div j \uparrow + \uparrow 96 \div j \uparrow$ $+ \uparrow \{48 + 68 \times (2 \times i + 7)\} \div j \uparrow + \uparrow (12 \times c + 4) \div j \uparrow$ $+ \uparrow [20,480 \times \{1 + \uparrow ((152 \times f) - (16 \times g) - (80 \times Q) + (8 \times R))$ $\div 20,476 \uparrow + \uparrow (4g + 4S) \div 20,480 \uparrow\} \div j \uparrow^{\#3}$ $+ \uparrow 32 \div j \uparrow + H + \uparrow (4 + 16 \times c) \div j \uparrow$ <p>■ If the status file record length < 4,096, add:</p> $+ \downarrow \text{MAX}\{3,400 \div D + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4,096 \div n \downarrow, 2) \times k$ $+ \downarrow 5,662,310 \div D + 0.7 \downarrow \times \text{MAX}(\downarrow 4,096 \div n \downarrow, 2) \times m$ <p>■ If $4,096 \leq$ the status file record length < 12,288, add:</p> $+ \downarrow \text{MAX}\{(3,400 \div \downarrow (n - 348) \div 20 \downarrow) + 0.7, 1\} \downarrow \times k$ $+ \downarrow (5,662,310 \div \downarrow (n - 348) \div 20 \downarrow) + 0.7 \downarrow \times m$ <p>■ If $12,288 \leq$ the status file record length, add:</p> $+ \downarrow \text{MAX}\{(3,400 \div \downarrow (n - 876) \div 20 \downarrow) + 0.7, 1\} \downarrow \times k$ $+ \downarrow (5,662,310 \div \downarrow (n - 876) \div 20 \downarrow) + 0.7 \downarrow \times m$ <p>■ When using the system switchover facility, add:</p> $+ \uparrow ((\uparrow (256 \times y + 4,096) \div 4,096 \uparrow \times 4,096) \div j \uparrow$ <p>■ If you are using the standby-less system switchover (effects distributed) facility, add:</p> $+ \uparrow 2,816 \div j \uparrow + \uparrow 80 \div j \uparrow$ $+ \uparrow 32 \div j \uparrow$ <p>■ If the <code>pd_log_auto_unload_path</code> operand is specified, add:</p> $+ \uparrow 20,848 \div j \uparrow$ <p>■ If you specify Y for the <code>pd_dbbuff_modify</code> operand, add:</p> $+ G$ <p>■ If the HiRDB Staticizer Option has been set up and the <code>pd_max_reflect_process_count</code> operand is specified, add:</p> $+ \uparrow 256 \div j \uparrow$ $+ \uparrow (128 + 704) \div j \uparrow$ <p>■ If XDS is defined, add:</p>

Type	Formula for S
	$+ \uparrow 32 \div j \uparrow$

a: Number of specified `pdlogadfg` operands

b: Number of specified `pdlogadfg -d spd` operands

c: Value of `pd_max_rdarea_no` operand

d: 3

For a HiRDB parallel server configuration on a single server machine, the value is 2.

e: (value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand) + 1

However, if a HiRDB parallel server configuration is used with a single server machine, (value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand)

f: Number of specified `pdbuffer` operands

g: Number of `-i` options specified in the `pdbuffer` operand

h: Value of `pd_max_dic_process` operand + value of `pd_max_reflect_process_count` operand

i: Value of `pd_max_bes_process` operand + value of `pd_max_reflect_process_count` operand

j: Record length of status file minus 40

k: If the number of RDAREAs in the server $\leq 3,400$, the value is 1.

If $3,401 \leq$ number of RDAREAs in the server $\leq 6,800$, the value is 2.

If $6,801 \leq$ number of RDAREAs in the server, the value is 3.

m: If the number of RDAREAs in the server $\leq 10,200$, the value is 0.

If $10,201 \leq$ number of RDAREAs in the server $\leq 5,672,510$, the value is 1.

If $5,672,511 \leq$ number of RDAREAs in the server $\leq 11,334,820$, the value is 2.

If $11,334,821 \leq$ number of RDAREAs in the server, the value is 3.

n: Status file record length

p: Number of back-end servers in unit

q: 1 if there is a dictionary server in the unit; otherwise, 0

r: 1 if there is a front-end server in the unit; otherwise, 0

s: 1 if there is a system manager in the unit; otherwise, 0

y: When the inner replica facility is used, value of the `pd_max_file_no` operand

When the inner replica facility is not used, value of the `pd_max_rdarea_no` operand

z: 1 in the following cases:

- When there is a system manager in the unit
- When there is no system manager in the unit and `local` is specified in the `pd_mlg_msg_log_unit` operand

0 when there is no system manager in the unit and `manager` is specified in the `pd_mlg_msg_log_unit` operand or specification is omitted

A: 736

B: 11

C: 1

D: $\downarrow (n - 348) \div 20 \downarrow + \downarrow j \div 20 \downarrow \times (\text{MAX}(\downarrow 4,096 \div n \downarrow, 2) - 1)$

G:

32-bit mode: $\uparrow (24 + 28 \times (I + 512) + 32 + 112 \times T) \div j \uparrow$

64-bit mode: $\uparrow (32 + 32 \times (I + 512) + 32 + 144 \times T) \div j \uparrow$

H:

32-bit mode: $\uparrow((12 + (\uparrow\uparrow(c \div 8)\uparrow \div 4\uparrow) \times 4) \times K) \div j\uparrow$

64-bit mode: $\uparrow((12 + (\uparrow\uparrow(c \div 8)\uparrow \div 8\uparrow) \times 8) \times K) \div j\uparrow$

I: Value of `pd_max_add_dbbuff_shm_no` operand

J: Value of `pd_max_add_dbbuff_no` operand

K: Value obtained by adding the `pd_max_add_dbbuff_no` operand value to the total number of specified `-i` options in the `pdbuffer` operand (if a value other than *Y* is specified in the `pd_dbbuff_modify` operand, 0 is used)

L: If the `pd_system_expand_unit` operand is specified: 16

If the `pd_system_expand_unit` operand is omitted: 0

M: If the `pd_system_expand_unit` operand is specified: number of units specified in the `pd_system_expand_unit` operand

If the `pd_system_expand_unit` operand is omitted: 0

N: If the `pd_ha_max_act_guest_servers` operand is specified: the value specified for the `pd_ha_max_act_guest_servers` operand

If the `pd_ha_max_act_guest_servers` operand is omitted: 0

P: Number of alternate BESs in the unit (alternate BESs when using the standby-less system switchover (1:1) facility)

Q: With the `-o` option specified for the `pdbuffer` operand of the system common definition: 1

Without the `-o` option specified for the `pdbuffer` operand of the system common definition: 0

R: Total number of RDAREAs specified in the `-r` and `-b` options of the `pdbuffer` operand of the system common definition

S: Total number of RDAREAs containing indexes specified in the `-i` option of the `pdbuffer` operand of the system common definition

T: If *Y* is specified for the `pd_dbbuff_modify` operand of the system common definition:

- If the `pd_max_add_dbbuff_no` operand of the system common definition is specified: $f + J$
- If the `pd_max_add_dbbuff_no` operand of the system common definition is omitted: 1,000

If *N* is specified for the `pd_dbbuff_modify` operand of the system common definition: f

#1: For the 64-bit mode, use a value of 3,456.

#2: For the 64-bit mode, use a value of 688.

#3: Do not add this value if you use the standby-less system switchover (effects distributed) facility.

17.4 Determining audit trail file capacity

Determine the capacity of the HiRDB file system for audit trail files using the following formulas.

Formula

HiRDB file system capacity for audit trail files (MB) = $a + 19$

a: Maximum amount of audit trail data (MB)

$$\uparrow (\sum \{b \times c\}) \div (1,024 \times 1,024) \uparrow$$

When an audit trail file input/output error has occurred or an audit trail file swap has been executed using the `pd_audswap` command, the swap occurs before the audit trail file reaches capacity. In this case, the free space cannot be used until registration of data to the audit trail table terminates. For this reason, we recommend doubling the value of *a* in advance.

b: Audit trail record size (bytes)

Find the audit trail record size using the following formula.

$$464 + d + e + f + g + h + i$$

c: Number of audit trail records

This is the number of records recorded by audit trail event type.

d: Length of SQL statement output to the audit trail record (bytes)

If the value of `pd_aud_sql_source_size` is 0 or is not specified: 0

If there is no SQL statement in the record item: 0

If there is a SQL statement in the record item, but the SQL statement is `NULL`: 0

In other cases:

$$\uparrow \text{MIN}(\text{value of } pd_aud_sql_source_size, \text{average SQL statement length}) \div 4 \uparrow \times 4 + 8$$

e: Length of SQL data output to audit trail record (bytes)

If the `pd_aud_sql_data_size` value is 0 or is not specified: 0

If there is no SQL data in the record item: 0

If there is SQL data in the record item, but the SQL data is `NULL`: 0

In other cases:

$$\uparrow \text{MIN}(\text{value of } pd_aud_sql_data_size, \text{average length of SQL data}) \div 4 \uparrow \times 4 + 8$$

f, g, h: Length of additional user information 1, 2 and 3 output to audit trail record (bytes)

If there is no additional user information in the record item or if that information is `NULL`: 0

In other cases:

$$\uparrow \text{average length of additional user information} \div 4 \uparrow \times 4 + 8$$

i: Length of additional related product information output to audit trail record (bytes)

If there is no additional related product information in the record item or if that additional related product information is `NULL`^{#1}: 0

In other cases:

$$\uparrow \text{average length of additional related product information}^{\#2} \div 4 \uparrow \times 4 + 8$$

#1

For details about conditions when additional related product information is `NULL`, see the relevant product documentation.

#2

For details about average length of additional related product information, see the relevant product documentation. If you find no pertinent information in that documentation, calculate the maximum length of additional related product information.

For details about the audit trail record length and information output to the audit trail file, see the *HiRDB Version 9 System Operation Guide*.

18

Determining Work Table File Size

This chapter explains how to determine the size of a work table file.

18.1 Overview of work table files

This section describes the work table files that are used to temporarily store information needed to execute SQL statements.

(1) Work table file creation timing

A work table file is a file that stores temporary information that is generated when the following operations are performed:

- Execution of SQL statements[#]
- Batch index creation
- Index re-creation
- Index reorganization
- Execution of the rebalancing utility

[#]

A work table file is used at the time that a specific SQL statement is executed, such as when you join multiple tables in a `SELECT` statement for retrieval, or execute `CREATE INDEX`. The following types of SQL processing require work table files:

1. Retrieval specifying the `UNION [ALL]` or `EXCEPT [ALL]` clause
2. `DROP SCHEMA`
3. `DROP TABLE`
4. `DROP INDEX`
5. Revocation of access privileges by using `REVOKE`
6. `CREATE INDEX`
7. Creation of a list from a base table with an `ASSIGN LIST` statement
8. Specification of the following in a `SELECT` statement:
 - Retrieval by joining multiple tables
 - Specification of an `ORDER BY` clause in a column for which no index is defined
 - Specification of an `ORDER BY` clause in a row-partitioned table
 - Specification of a value expression containing a set function in the selection expression (applies only to a HIRDB parallel server configuration)
 - Specification of a value expression that includes the window function `COUNT (*) OVER ()` in the selection expression
 - Specification of a `GROUP BY` clause
 - Specification of a `DISTINCT` clause
 - Specification of a retrieval condition based on multiple columns for which an index is defined
 - Specification of a retrieval condition for a column for which a repetition column index is defined
 - Specification of the facility for batch acquisition from functions provided by plug-ins for the SQL optimization option, and specification of and searching for functions provided by plug-ins that use a plug-index for a retrieval condition.
 - Specification of a retrieval condition based on a column for which an index is defined, and for which either a `FOR UPDATE` clause is specified or an update using this cursor exists
 - Specification of a `FOR READ ONLY` clause
 - Specification of a subquery of a quantified predicate
 - Specification of a subquery of the `IN` predicate

- Creation of an internally derived table in a retrieval from a view table or a retrieval in which a `WITH` clause is specified
9. Specification in the query body at the insertion source of the `INSERT` statement of either of the following:
- An update table for a subquery that has an external reference
 - An update table for the main query of the query expression body at the insertion source
10. Specification in an `UPDATE` statement of any of the following:
- A subquery that has an external reference in a search condition or update value, and specification of an update table in that subquery
 - A subquery with a quantified predicate in a search condition
 - A subquery with the `IN` predicate in a search condition
 - A column for which an index is defined as the update target and search condition, and use of that index
11. Specification in a `DELETE` statement of any of the following:
- An update table for a subquery in a search condition that has an external reference
 - A subquery with a quantified predicate in a search condition
 - A subquery with the `IN` predicate in a search condition
 - A column for which an index is defined as the search condition, and use of that index
12. `ALTER TABLE ADD PRIMARY KEY`
13. `ALTER TABLE DROP PRIMARY KEY`

(2) Storing work table files

HiRDB creates a work table file in a HiRDB file system area. The HiRDB administrator must do the following:

- *Use the `pdfmkfs` command to initialize HiRDB file system areas for creation of work table files.*
- *Use the `pdwork` operand of the system definition to specify the name of the HiRDB file system area that is to be used.*

This section explains how to determine the values to be specified in options of the `pdfmkfs` command options. The following table describes the relationships between the `pdfmkfs` command options and work table file-related items.

Table 18–1: Options for which values need to be specified

Option	Description
<code>-n</code>	Size of HiRDB file system area in which work table files are to be created
<code>-l</code>	Maximum number of HiRDB files (work table files) that can be created in the HiRDB file system area
<code>-e</code>	Maximum number of secondary allocations for the HiRDB file system area
<code>-a</code>	Whether the HiRDB file system area extends automatically

18.2 Determining the size of a HiRDB file system area (pdfmkfs -n command)

Use the `pdfmkfs` command's `-n` option to specify the size of a HiRDB file system area in which a work table file is created.

The following formula is used to obtain the size of a HiRDB file system area in which work table files are to be created.

Formula

Size of HiRDB file system area (bytes) = A + B

A

Size of a work table file to be used by an SQL statement. For details about how to obtain this value, see *18.2.1 Size of a work table file used by an SQL statement*.

B

Size of a work table file used by the database load utility (`pdload`), database reorganization utility (`pdrorg`), and rebalancing utility (`pdrbal`). For details about how to obtain this value, see *18.2.2 Size of a work table file used by a utility*.

If you do not execute an SQL statement that uses a work table file concurrently with a utility that also uses a work table file, specify either A or B, whichever is larger, as the size of the HiRDB file system area.

Notes

If the size of the HiRDB file system area obtained with this formula is too large for one HiRDB file system area, initialize multiple HiRDB file system areas with the `pdfmkfs` command and specify the `pdwork` operand in the HiRDB system definition. In this case, note the following:

- Set the size of each HiRDB file system area to the same value.
- Make the size of each HiRDB file system area larger than the size of a work table (for storing column information).
- If you divide a HiRDB file system area into too many segments, unused area is distributed among multiple HiRDB file system areas, and a shortage of space may occur because the space is not used efficiently.
- If the size of a single work table file exceeds 2 gigabytes, use large files. A single work table file cannot be partitioned among multiple HiRDB file system areas.

18.2.1 Size of a work table file used by an SQL statement

To determine the size of a work table file used by an SQL statement, use the following **formula**:

Size of a work table file used by an SQL statement (bytes) = MAX (a, b) × c

a: Maximum size of a work table file that is used by one SQL statement

Calculate the size of a work table file for each SQL statement and use the largest such size as the value of a. For details about how to obtain this value, see *(1) Formula for calculating the size of the work table file to be used by one SQL* as follows.

b: Maximum size of a work table file that is used by an `ASSIGN LIST` statement.

Calculate the size of a work table file for each `ASSIGN LIST` statement and use the largest such size as the value of b. For details about how to obtain this value, see *(2) Formula for calculating the size of the work table file to be used by the ASSIGN LIST statement* as follows.

c: Value of the `pd_max_users` operand + value of the `pd_max_reflect_process_count` operand

However, when multiple front-end servers are being used, the back-end servers are (value of the `pd_max_bes_process` operand + value of the `pd_max_reflect_process_count` operand).

(1) Formula for calculating the size of the work table file to be used by one SQL

The following formula is used to calculate the size of the work table file that is to be used by one SQL statement.

Formula

Size of work table file to be used by one SQL statement (bytes) = $a \times b + c \times d$

a: Size of a column information work table

b: Maximum number of column information work tables

c: Size of a location information work table

d: Maximum number of location information work tables

(a) Obtaining the size of a column information work table

To obtain the size of a column information work table, use the following formula:

Formula

Size of a column information work table (bytes)

= $\uparrow a \div \text{MIN}\{\downarrow(b - 48) \div c \downarrow, 255\} \uparrow \times b \times 2$

a: Number of rows in the column information work table (see *Table 18-2 Obtaining the number of rows in a column information work table*)

b: Page length of the work table (use Formula 1 as follows)

c: Row length of the work table (use Formula 2 as follows)

Formula 1

Page length for a work table[#] = $\text{MAX}\{\uparrow(\text{row length for work table} + 48) \div 2,048 \uparrow \times 2,048, 4,096\}$

[#]: The page length of a work table must be no greater than 32,768 bytes.

Formula 2

Row length for a work table[#] =

$$\sum_{i=1}^n A_i + 2 \times n + 6$$

A_i:

Data length for each column in work table (see *Table 18-3 Obtaining the data length for each column and the number of columns in a work table* for the calculation procedure)

n:

Number of columns in work table (see *Table 18-3 Obtaining the data length for each column and the number of columns in a work table* for the calculation procedure)

[#]: The row length of a work table must be no greater than 32,720 bytes.

If the `LIMIT` clause is specified and the value of (number of offset rows + number of rows specified in the `LIMIT` clause) is 32,768 or greater, add 12 to the row length of the work table obtained from Formula 2.

However, addition of 12 is not necessary in the following cases:

- The table to be searched is locked in the EX mode.
- `WITHOUT LOCK` was specified.
- The rapid grouping facility was specified.
- Multiple tables are to be joined.

Table 18–2: Obtaining the number of rows in a column information work table

SQL statement	Number of rows in column information work table
SELECT statement	This is the total number of rows subject to retrieval in individual tables. If multiple tables are joined, then use the resulting number of rows, if it is greater.
<ul style="list-style-type: none"> CREATE INDEX ALTER TABLE ADD PRIMARY KEY 	This is the number of rows in a table. For an index for repetition columns, use the total number of elements per repetition column among the index component columns.

Table 18–3: Obtaining the data length for each column and the number of columns in a work table

SQL statement	N	A _i
SELECT statement	<p>Number of columns specified in the select expression + number of columns specified in the GROUP BY clause[#] + number of columns specified in the ORDER BY clause[#] + number of column specified in the HAVING clause[#] + 1 if the FOR UPDATE clause is specified[#]</p> <p>If ROW is specified in the selection expression, specify the total number of rows in the table.</p>	<p>Data length for each column</p> <p>However, for large object data (BLOB), character data with a defined length of 256 or greater (including National and mixed character data), or binary data for columns that do not have the following attributes or for location information columns: 12</p> <ul style="list-style-type: none"> Column specified in a join condition (join column) Selection expression with DISTINCT clause specified Column specified in the subquery selection expression with a quantified predicate Column specified in the subquery selection expression with the IN predicate Selection expression in a query specification subject to set operation with UNION [ALL] or EXCEPT [ALL] Column specified in the ORDER BY clause
<ul style="list-style-type: none"> CREATE INDEX ALTER TABLE ADD PRIMARY KEY 	1 (index information column) + 1 (positional information column)	<ul style="list-style-type: none"> For an index information column, specify the sum of the data lengths for index component columns 12 for a positional information column

Note

For details about the data lengths of columns, see the following tables:

- Table 16-1 List of data lengths
- Table 16-2 Data lengths for the variable-length character string type (except abstract data type and repetition columns)
- Table 16-3 Data lengths for the variable-length character string type (abstract data type)
- Table 16-4 Data lengths for the variable-length character string type (repetition columns)

#

When the columns are the same as ones specified in the selection expression, there is no need to add this term.

(b) Obtaining the maximum number of column information work tables

The following table shows how to calculate the maximum number of column information work tables.

Table 18–4: Obtaining the maximum number of column information work tables

SQL statement	Maximum number of work tables for storing column information ^{#1}
SELECT statement	When none of 1-10 as follows is applicable: 0

SQL statement	Maximum number of work tables for storing column information ^{#1}
	<p>When any of 1-10 as follows is applicable: Sum of all the applicable values from 1-10</p> <ol style="list-style-type: none"> When multiple tables are joined for retrieval Number of additional work tables (HiRDB single server configuration) = number of joined tables + 1 Number of additional work tables (HiRDB parallel server configuration) = number of joined tables \times 2 If the join key column has an index and there is a limitation condition, the number of work tables is 0. When specifying the ORDER BY clause Number of additional work tables = 2 When an index containing all the columns specified in the ORDER BY clause is to be used for search processing = 0 When specifying a value expression containing a set function in the selection expression without specifying the GROUP BY clause^{#2} Number of additional work tables = 1 When specifying the GROUP BY clause Number of additional work tables = number of GROUP BY clauses specified \times 2 When specifying the DISTINCT clause Number of additional work tables = number of DISTINCT clauses specified \times 2 When specifying the UNION [ALL] or EXCEPT [ALL] clause Number of additional work tables (HiRDB single server configuration) = number of UNION [ALL] or EXCEPT [ALL] clauses specified + 2 Number of additional work tables (HiRDB parallel server configuration) = (number of UNION [ALL] or EXCEPT [ALL] clauses specified + 1) \times 2 When specifying the FOR UPDATE clause or when using this cursor for updating purposes and specifying a search condition for a column with the index defined^{#2} Number of additional work tables = 2 When specifying the FOR READ ONLY clause Number of additional work tables = 1 When specifying a subquery (quantified predicate) Number of additional work tables = number of subqueries specified + (number of =ANY quantified predicates for a column with the index defined) + (number of IN predicate subqueries specified for a column with the index defined) + (number of =SOME quantified predicates for a column with the index defined) When specifying the window function COUNT (*) OVER () in a selection expression Number of increased work tables = number of query specifications in which the window function is specified in the selection expression
<ul style="list-style-type: none"> CREATE INDEX ALTER TABLE ADD PRIMARY KEY 	2

#1: A work table might not be created depending on the access cost determined by HiRDB.

#2: Applicable only to a HiRDB parallel server configuration

(c) Obtaining the size of a location information work table

To obtain the size of a location information work table, use the following formula:

Formula

Size of a location information work table (bytes)

$$= \uparrow a \div 184^{\#} \uparrow \times 4,096 \times 2$$

#: If an index-type plug-in function is specified as the search condition, use the value 155.

a : Number of rows in the location information work table

The following shows the procedure for obtaining the number of rows in the location information work table:

SQL statement	Obtaining the number of rows in the location information work table
SELECT statement UPDATE statement DELETE statement	<p>If the search condition contains one predicate that includes a column with the index defined, use the number of rows for which the predicate is true. If there is more than one predicate, use the sum of the following values:</p> <ul style="list-style-type: none"> • If OR operation is conducted on the predicates, the total number of rows for which at least one predicate is true. • If AND operation is conducted on the predicates, sum of the larger numbers of rows for which the predicates are true.

(d) Obtaining the maximum number of location information work tables

The following table shows how to calculate the maximum number of location information work tables.

Table 18–5: Obtaining the maximum number of location information work tables

SQL statement	Maximum number of location information work tables
SELECT statement	<p>Number of indexes to be used during search + 1 in either of the following cases:</p> <ol style="list-style-type: none"> 1. Search condition is specified for multiple columns with index defined. 2. The FOR UPDATE clause is specified or this cursor is used for updating purposes and a search condition is specified for the column with index defined.[#] 3. Search condition is specified for a column for which a repetition column index is defined. 4. <i>Facility for batch acquisition from functions provided by plug-ins</i> is specified as the SQL optimization option and a function provided by a plug-in that uses a plug-in index is specified as a search condition. <p>In all these cases, the value is the number of indexes used during a search + 1.</p>
UPDATE statement DELETE statement	<p>If the search condition contains a column with an index defined, use the number of indexes used during search processing + 1.</p>

[#]: This is applicable to a HiRDB single server configuration only.

(2) Formula for calculating the size of the work table file to be used by the ASSIGN LIST statement

To obtain the size of a work table file used by the ASSIGN LIST statement, use the following formula:

Formula

Size of work table file to be used by the ASSIGN LIST statement (bytes) =

$$\sum_{i=1}^n (B_i \times 2)$$

n : Number of predicates in the selection condition of the ASSIGN LIST statement

B_i : Size of the work table used to process predicate i in the search condition. Use the following formula to obtain this value:

$$B_i = \uparrow \text{number of rows for which predicate } i \text{ is true in the base table for the list}^{\#} \div 504 \uparrow \times 4,096 \times 1.5$$

(bytes)

[#]: If the predicate is a condition for a repetition column, this value is the total number of elements that are true.

18.2.2 Size of a work table file used by a utility

If you create an index in batch mode, re-create an index, reorganize an index, or reorganize data using the rebalancing utility, you need the following size of work table file:

Formula

Size of a work table file used by a utility (bytes) = $\{A + B\} \times 2 \times D \div C$

A: Number of rows in the work table required for index creation 1

B: Number of rows in the work table required for index creation 2

C: Number of rows per work table page

D: Page length of work table

Notes

- If you create multiple indexes in batch mode or re-create multiple indexes using one utility, obtain the size for the index with the longest index key.
- If you execute batch index creation and re-creation concurrently, obtain the size of work table file for each operation and add the sizes.
- If you execute multiple utilities concurrently, obtain the total of the sizes of the work table files calculated for each utility.

(1) Obtaining the number of rows in the work table required for index creation 1

To obtain the number of rows in the work table required for index creation 1, use the following formula:

Formula

Number of rows in work table 1

= $\lceil c \div \{ \lfloor a \times (100 - b) \times 0.01 \rceil \div (d + 22) \rfloor \} \rceil$

a: Page size of a user RDAREA used to store the index

b: Percentage of unused area specified in the PCTFREE operand of the CREATE INDEX statement

c: Number of data items

For the index for repetition columns, use the sum of the elements of each row per repetition column among the index component columns.

d: Length of index key

For details about the length of the index key, see *Table 16-5 List of index key lengths*. Because the key lengths stored in the database are based on a 4-byte boundary, it becomes $\lceil \text{key length} \div 4 \rceil \times 4$.

For multiple indexes, add the key lengths of all component columns on the basis of *Table 16-5 List of index key lengths*.

(2) Obtaining the number of rows in the work table required for index creation 2

To obtain the number of rows in the work table required for index creation 2, use the following formula:

Formula

Number of rows in work table 2

= $\lceil c \div \{ \lfloor a \times (100 - b) \times 0.01 \rceil \div (d + 14) \rfloor \} \rceil$

a: Page size of a user RDAREA used to store the index

b: Percentage of unused area specified in the PCTFREE operand of the CREATE INDEX statement

c: Number of rows in the work table required for index creation 1

Use the value obtained at (1) previously.

d: Length of index key

For details about the length of the index key, see *Table 16-5 List of index key lengths*. Because the key lengths stored in the database are based on a 4-byte boundary, it becomes $\lceil \text{key length} \rceil \div 4 \times 4$.

For multiple indexes, add the key lengths of all component columns on the basis of *Table 16-5 List of index key lengths*.

(3) Obtaining the number of rows per work table page

To obtain the number of rows per work table page, use the following formula:

Formula

$$\text{Number of rows per work table page} = \text{MIN}\{\lfloor (b - 48) \div a \rfloor, 255\}$$

a: Length of row in the work table (index key length + 18)

For details about the length of the index key, see *Table 16-5 List of index key lengths*. The key length is $\lceil \text{key length} \rceil \div 4 \times 4$.

For multiple indexes, add the key lengths of all component columns on the basis of *Table 16-5 List of index key lengths*.

b: Page length of the work table

See (4) as follows.

(4) Obtaining the page length of a work table

To obtain the page length of a work table, use the following formula:

Formula

$$\text{Page length of work table}^\# = \text{MAX}\{\lceil (\text{Row length of work table} + 48) \div 2,048 \rceil \times 2,048, 4,096\}$$

[#]: The page length of a work table must be no more than 32,768 bytes.

a: Length of row in the work table (index key length + 18)

For details about the length of the index key, see *Table 16-5 List of index key lengths*. The key length is $\lceil \text{key length} \rceil \div 4 \times 4$.

For multiple indexes, add the key lengths of all component columns on the basis of *Table 16-5 List of index key lengths*.

18.3 Determining the maximum number of files (pdfmkfs -l command)

To specify the maximum number of work table files to be created in a HiRDB file system area, use the `pdfmkfs` command's `-l` option.

You can use the following formula to determine the maximum number of work table files that need to be created in a HiRDB file system area:

Formula

Maximum number of files = $\text{MAX}(a, b) \times c + 20 + 2^{\#}$

a: Number of work table files to be used by one SQL statement

Calculate the number of work table files to be used by each SQL statement and specify the largest such value for *a* in the formula; see (1) as follows.

b: Number of work table files to be used by an `ASSIGN LIST` statement

Calculate the number of work table files to be used by each `ASSIGN LIST` statement and specify the largest such value for *b* in the formula; see (2) as follows.

c: Value of the `pd_max_users` operand + value of the `pd_max_reflect_process_count` operand

However, when multiple front-end servers are being used, the back-end servers are (value of the `pd_max_bes_process` operand + value of the `pd_max_reflect_process_count` operand).

#: Add this value if you execute an SQL statement that uses a work table file concurrently with a utility that also uses a work table file (database load utility or database reorganization utility).

(1) Obtaining the number of work table files to be used by one SQL statement

To obtain the number of work table files to be used by one SQL statement, use the following formula:

Formula

Number of work table files to be used by one SQL statement =
maximum number of column information work tables + maximum number
of location information work tables

For details about the maximum numbers of column information work tables and location information work tables, see *18.2.1 Size of a work table file used by an SQL statement*.

(2) Obtaining the number of work table files to be used by an `ASSIGN LIST` statement

To obtain the number of work table files to be used by an `ASSIGN LIST` statement, use the following formula:

Formula

Number of work table files to be used by an `ASSIGN LIST` statement =
number of predicates in the search condition of `ASSIGN LIST`
statement $\times 2$

(3) Note

When specifying multiple HiRDB file system areas to create work table files, note the following:

- If the value obtained is greater than 4,096, specify a value of 4,096 in the `-l` option.

18.4 Determining the maximum number of extensions (pdfmkfs -e command)

To specify the maximum number of extensions for a work table file in a HiRDB file system area, use the `-e` option in the `pdfmkfs` command.

You can use the following formula to determine the maximum number of extensions for the HiRDB file system area:

Formula

Maximum number of extensions

= MIN(maximum number of files × 23, 60,000)

Note 1

When the maximum number of extensions is smaller than the estimated value, it might become impossible to secure area even when there is free space in the HiRDB file system area.

Note 2

For details about how to obtain the maximum number of files, see *18.3 Determining the maximum number of files (pdfmkfs -l command)*.

19

Storage Requirements for Utility Execution

This chapter explains how to determine the file sizes and storage requirements for execution of utilities.

19.1 Determining the file sizes required for utility execution

19.1.1 File sizes required for the execution of the database load utility (pdload)

The following table shows the formulas for determining the file sizes required for the execution of the database load utility (pdload):

File type	Formula (bytes)
Input data file	$h \times b$
Index information file	B-tree index: $(d + y) \times (b + e) + 512$ Plug-in index: $(12 + q) \times p + 1,024$ These formulas are for the size of one index. If there are multiple indexes, determine the size of each index.
Error information file	$k \times f + s \times 200$
Temporary file for creating error information file	In the following conditions, the work file output directory will need <i>number of key duplication errors</i> $\times 8$ + <i>number of errors detected by plug in function</i> $\times 200$ of space for each server that has a table storage RDAREA. For details about work file output destination directories, see 2.3.2 <i>Creating a work file output directory</i> . <ul style="list-style-type: none"> For a HiRDB single server configuration, a utility special unit contains the input files. For a HiRDB parallel server configuration, the server that contains the input files is different from the server containing the table storage RDAREAs.
LOB input file	EasyMT used to create the LOB input file: a $\sum_{i=1} (LOB \text{ data length} + 400)$ LOB input file by column: b $\sum_{i=1} (LOB \text{ data length} + 4)i$
LOB middle file	B $\sum_{i=1} \{$ c $\sum_{j=1} (LOB \text{ file name length} - ij + 36) + 24\} + 1,024 + c \times 84$
Error data file	$\text{MIN}(f, g) \times h$
Process results file	$1,500 + \text{number of servers storing table} \times 500$
Work file [#]	$[4 + 2 \times R + 2 \times r + 4 \times I \times R + \{b \div (\text{value of past message output interval specified by the } -m \text{ option})\}] \times 200$
Work file for sorting	Condition 1: Size of index information file + $4 \times (b + e)$

File type	Formula (bytes)
	<p>Condition 2:</p> <p>{Size of index information file + $4 \times (b + e)$} $\times 2$</p> <ul style="list-style-type: none"> • Condition 1 When $1,024 \text{ (KB)} \geq E$ • Condition 2 When $1,024 \text{ (KB)} < E$ <p><i>E</i>: Buffer size The buffer size obtained according to 19.1.12 Buffer size used to determine the size of the work file for sorting.</p>

a: Number of input rows \times number of LOB columns

b: Number of input rows (for a repetition column, number of input rows \times number of elements)

c: Number of LOB columns

d: Index key length

See Table 16-5 List of index key lengths. For variable-length data, treat a single column as multicolumn and use the largest defined length.

e: Number of existing rows (for a repetition column, number of existing rows \times number of elements)

f: Number of error data items

g: Number of output rows specified in the `errdata` operand of the `source` statement

h: Average source record length

k: If there is a column with an abstract data type, the value is 300; otherwise, it is 120.

m: For a DAT-format file or a binary format file output by `pdrorg`, the value is 0.

For any other file, the value is (record length of one row in the input file $\times 4$).

p: If index storage RDAREAs are initialized, the value is $(b + e)$; otherwise, the value is *b*.

q: Value as follows

- 27 for the abstract data type stored in the LOB RDAREA
- Key length + 2 for the abstract data type of a maximum of 255 bytes of definition length
- 2 for the abstract data type of 256 bytes or more of definition length

Typical abstract data type values are as follows.

- 27 for the `SGMLTEXT` type
- 2 for the `FREEWORD`, `GEOMETRY`, and `XML` types

r: Number of RDAREAs for LOB storage

s: Number of servers

y: If all key component columns are fixed length, the value is 10; if they include a variable length, the value is 12.

I: Number of indexes

R: Number of partitioned index or table RDAREAs

Note

When calculating the size of index information files and sort work files, if the index configuration columns are repetition columns, *b* and *e* do not refer to the number of rows but to (number of rows \times number of elements).

#

Output if `lv12` is specified as the information message output suppression level in the `-m` option.

19.1.2 File sizes required for the execution of the database reorganization utility (pdrorg)

The following table shows the formulas for determining the file sizes required for the execution of the database reorganization utility (pdrorg):

File type	Formula (bytes)
Unload data file ^{#1} (no options specified)	n $\sum_{i=1} (P_i + M) + L_i^{#7} + 1,200 + A + B + c \times 96 + D + I + F$
Unload data file ^{#1} (-w option specified)	<p>DAT or extended DAT format:</p> $\{$ c $\sum_{i=1} (\text{maximum length of converted character string for column } i^{#2} + 1) \times C + M$ $\} \times n$ <p>FIX table in binary format:</p> $\{$ c $\sum_{i=1} (\text{column data length } i^{#3}) + M$ $\} \times n$ <p>Non-FIX table in binary format:</p> $\{$ c $\sum_{i=1} (\text{column data length } i^{#3} + G) + M + 4 \times (c + 1)$ $\} \times n$ <p>Fixed-length character format:</p> $\{$ c $\sum_{i=1} (\text{maximum length of converted character string for column } i^{#4} + \text{crlf}) \times C + M$ $\} \times n$
Unload data file ^{#1} (-j option specified or during reorganization in units of schemas) ^{#5}	n $\sum_{i=1} (P_i + M) + L_i^{#7} +$ n $\sum_{i=1} \{$ m $\sum_{j=1} (O_{ij} + 44)$ $\} + 1,200 + A + B + c \times 96 + D + I + F$
LOB data unload file ^{#1}	n $\sum_{i=1} \{$

File type	Formula (bytes)
	m $\sum_{j=1} (O_{ij} + 44)$ $\} + 1,200 + A + B + c \times 96 + D + I + F$
Index information file	B-tree index: $(K + p) \times n + 512$ Plug-in index: $(12 + X) \times n + 1,024$ These formulas are for the size of one index. If there are multiple indexes, determine the size of each index.
Process results file	$1,700 + \text{number of servers storing table} \times 500 + \text{number of tables in schema} \times 1,000 + \text{total number of storage RDAREAs in schema} \times 100$
Work file ^{#6}	$[8 + 2 \times S + 2 \times \{n \div (\text{value of past message output interval specified by the } -m \text{ option})\} + 3 \times R + 4 \times J \times R] \times 200$
Work file for sorting	Condition 1: Size of index information file + $4 \times n$ Condition 2: $\{\text{Size of index information file} + 4 \times n\} \times 2$ <ul style="list-style-type: none"> Condition 1 When $1,024 \text{ (KB)} \geq E$ Condition 2 When $1,024 \text{ (KB)} < E$ <i>E</i> : Buffer size The buffer size obtained according to 19.1.12 Buffer size used to determine the size of the work file for sorting.

A: For key range partitioning: $48 + \text{number of partitioning conditions} \times 284$

For hash partitioning: $40 + a \times 60$

For matrix partitioning (combination of key range partitioning of the boundary value specification and hash partitioning): $48 + (\text{number of partitioning conditions} \times 284) + (40 + a \times 60)$

B: $n \times 36$ (for FIX table) or $(44 + c \times 4) \times n$ (for non-FIX table)

C: If the output character encoding of the unload data file is not a default HiRDB character encoding: 2
 Otherwise: 1

D: $16 + (\text{number of LOB columns} \times a \times 80)$

Add the value of *D* only if there are LOB columns.

F: Use the following value:

D

$\sum_{i=1} \{(\text{number of abstract-data-type attributes provided by plug-in in column } i \times 84) +$

$(\text{number of abstract-data-type LOB attributes provided by plug-in in column } i \times a \times 72)\}$

$+ 64 +$

d

$\sum_{i=1} (84 + \text{number of reverse generation functions } i \times 60)$

$i = 1$

G: Number of attributes for which the return value of the reverse generation function on column *I* is BLOB $\times 4$

I : $136 + \text{number of index partitions} \times 60$

Add this value when including the index.

J : Number of indexes

K : Index key length

See *Table 16-5 List of index key lengths*. For variable-length data, keep in mind when defining the maximum length that single columns are also handled as multicolumns.

L_i : Actual length of row

Obtain the actual row length (approximate or accurate value). If the row is `BINARY` type and has the compression specification, we recommend that you obtain an approximate value because obtaining an accurate value requires a complicated calculation.

Obtaining an approximate value:

Use the following formula to obtain from the data stored in the database an approximate value (bytes) for the sum of the actual lengths of all rows:

$$\text{Number of pages used in the table storage RDAREA} \times \text{page length of the table storage RDAREA}$$

You can obtain the number of pages used in the table storage RDAREA and the table storage RDAREA's page length from the results of a condition analysis by RDAREA (logical analysis) or table that are provided by `pddbst`.

Obtaining an accurate value:

Use the following values to obtain the actual row length for the data stored in each column:

Column's data type	Actual length (bytes)
BLOB	16
Abstract data type provided by plug-ins	2
BINARY [#]	<ul style="list-style-type: none"> If <code>pdorg -k rorg</code> is specified with the compression specification and no UOC is used $\text{Definition length} + 8 \times \text{MAX}(\text{number of times a concatenation operation was used in the UPDATE SQL statement},$ $\uparrow \text{definition length} \div \text{split compression size} \uparrow)$ Otherwise Definition length of the <code>BINARY</code> type column
Other	Actual length of each column

[#]

For obtaining the maximum actual row length, this assumes `BINARY` type data whose length is the definition length and that the compression rate is 0%.

M : Total data length of the character string type column for which the character set is specified

The value is as follows:

k

($\sum \text{column data length } i$)

$i = 1$

O_{ij} : LOB data length

P_i : Data length of the abstract data type provided by a plug-in

R : Number of partitioned table or index RDAREAs

S : Number of table-storing servers

X : Value is as follows

- 27 for the abstract data type stored in the LOB RDAREA
- Key length + 2 for the abstract data type of a maximum of 255 bytes of definition length

- 2 for the abstract data type of 256 bytes or more of definition length

Typical abstract data type values are as follows.

- 27 for the `SGMLTEXT` type
- 2 for the `FREEWORD`, `GEOMETRY`, and `XML` types

a: Number of partitioned `RDAREAs`

c: Number of column definitions

d: Number of columns for which the abstract data type provided by a plug-in is defined

k: Number of character string type columns for which the character set is specified

m: Number of LOB columns

n: Number of rows (for a repetition column, number of rows \times number of elements)

p: If all key component columns are fixed length, the value is 10; if they include a variable length, the value is 12.

crlf: Length of linefeed characters added when `cr` or `crlf` is specified in the `-W` option

Determine the length of linefeed characters from the following table:

-W option value		Value to be added
-W dat or -W extdat	, cr	1
	, crlf	2
	Not specified	1
-W fixtext	, cr	1
	, crlf	2
	Not specified	0

Note

When calculating the size of index information files and sort work files, if the index configuration columns are repetition columns, the number of rows to reload and *n* do not refer to the number of rows but to (number of rows \times number of elements).

#1: If the file is larger than 2 GB, take one of the following actions:

- Create multiple files, each of which is no larger than 2 GB.
- Use large files. For details about how to create large files, see *4.3 Creating HiRDB file system areas*.

#2: The following table lists maximum lengths of converted character strings for columns in DAT format (`-W dat`) or extended DAT format (`-W extdat`).

Table 19–1: Maximum lengths of converted character strings for columns (in DAT or extended DAT format)

Data type		Maximum length of converted character string (bytes)
Numeric data	INTEGER	11
	SMALLINT	11
	DECIMAL	40
	FLOAT	23
	SMALLFLT	23
Character string data ^{#1}	CHARACTER	Defined length + 2 ^{#2}
	VARCHAR	Actual length + 2 ^{#2}
Mixed character string data ^{#1}	MCHAR	Defined length + 2 ^{#2}

Data type		Maximum length of converted character string (bytes)
	MVARCHAR	Actual length + 2# ²
National character data# ¹	NCHAR	Defined length + 2# ²
	NVARCHAR	Actual length + 2# ²
Date data	DATE	10
Time data	TIME	8
Date interval data	INTERVAL YEAR TO DAY	9
Time interval data	INTERVAL HOUR TO SECOND	7
Time stamp data	TIMESTAMP	19 If the number of digits for fractions of a second is not 0, add the number of digits for fractions of a second + 1.
Binary data# ¹	BINARY	Actual length + 2# ²

#1: If data in extended DAT format contains a double quotation mark ("), the length of the converted character string becomes longer by the number of double quotation marks.

#2: Two bytes are added for the enclosing brackets.

If `-W dat` or `-W extdat` is specified and `sup` is specified in the operand, the maximum lengths of converted character strings take effect on the columns as shown below. Note that the *actual length* indicates the length without the trailing consecutive spaces. For details about the space-compressed output format, see the `-W` option of the database reorganization utility (`pdroorg`) in the manual *HiRDB Version 9 Command Reference*.

Data type		Maximum lengths of converted character string (bytes)
Character string data	CHARACTER	Actual length + 2
Mixed character string data	MCHAR	Actual length + 2
National character data	NCHAR	Actual length + 2

#3: For details about the data length, see the following tables:

- *Table 16-1 List of data lengths*
- *Table 16-2 Data lengths for the variable-length character string type (except abstract data type and repetition columns)*
- *Table 16-3 Data lengths for the variable-length character string type (abstract data type)*
- *Table 16-4 Data lengths for the variable-length character string type (repetition columns)*

#4: The following table lists maximum lengths of converted character strings for columns in fixed-length character format (`-W fixtext`).

Table 19–2: Maximum lengths of converted character strings for columns (fixed-length character format)

Data type		Maximum lengths of converted character string (bytes)
Numeric data	INTEGER	11
	SMALLINT	6
	DECIMAL	Number of digits + 2
	FLOAT	23
	SMALLFLT	23

Data type		Maximum lengths of converted character string (bytes)	
Character string data	CHARACTER VARCHAR	Defined length	If <code>fixtext_option</code> is specified in the <code>enclose</code> operand, add 2 to the output length.
Mixed character string data	MCHAR MVARCHAR	Defined length	
National character data	NCHAR NVARCHAR	Defined length × 2	
Date data	DATE	10	
Time data	TIME	8	
Date interval data	INTERVAL YEAR TO DAY	10	
Time interval data	INTERVAL HOUR TO SECONUD	8	
Time stamp data	TIMESTAMP	Decimal part 0:19 2:22 4:24 6:26	
Large object data	BLOB	0	
Binary data	BINARY	0	
Abstract data type	ADT	0	

#5: If you are reorganizing files in units of schemas (including unload files), use the sum of the values obtained for individual tables.

#6: Output if `lv12` is specified as the information message output suppression level in the `-m` option.

#7: To obtain an accurate value of Li , replace $(Pi + M) + Li$ with $(Li + Pi + M)$.

19.1.3 File sizes required for the execution of the statistics analysis utility (pdstedit)

The following table shows the formulas for determining the file sizes required for the execution of the statistics analysis utility (pdstedit):

File type		Formula (bytes)
Temporary work file	Statistical information about system activity	$4,096 \times \text{collection count}^{\#} \times 2$
	Statistical information about system activity per server	$4,096 \times \text{collection count}^{\#} \times \text{number of servers} \times 2$
	Statistical information about UAPs	$872 \times \text{number of UAPs to be executed or number of transactions to be executed} \times 2$
	Statistical information about SQL	$512 \times \text{number of SQLs to be executed} \times 2$
	Statistical information about SQL static optimization	$512 \times \text{SQL object cache mishit count}$
	Statistical information about SQL dynamic optimization	$49,624 \times \text{number of SELECT statements issued by OPEN or EXECUTE (including INSERT SELECT)}$
	Statistical information about SQL object execution	$512 \times \text{number of SQLs to be executed} \times \text{number of servers}$
	Statistical information about SQL object transfer	$256 \times \text{number of SQLs to be executed} \times \text{number of servers}$

File type		Formula (bytes)
	Statistical information about the history of SQL statements	$(1,024 + \text{average SQL length}) \times \text{number of SQLs to be executed}$
	Statistical information about CONNECT/DISCONNECT	$256 \times \text{number of CONNECTs and DISCONNECTs}$
	Statistical information about global buffer	$512 \times \text{number of synchronization points} \times 2$
	Statistical information about database manipulation for HiRDB files	$512 \times \text{number of synchronization points} \times 2$
	Statistical information about deferred write processing	$512 \times \text{number of deferred write operations} \times 2$
	Statistical information about indexes (input: STJ)	$128 \times \text{number of synchronization points} \times 2$
	Statistical information about indexes (input: FJ)	$128 \times \text{number of page splits} \times 2$
	Statistical information about database I/O	$256 \times \text{total number of HiRDB files that make up the RDAREAs accessed at each acquisition time interval} \times 2$
Work file for sorting	Work area for sorting the above types of temporary work files for analysis	Maximum size of the temporary work files that will be needed for analysis
DAT-format file	Statistical information about system activity	$3,404 \times \text{collection count}^\#$
	Statistical information about system activity per server	$3,262 \times \text{collection count}^\# \times \text{number of servers}$
	Statistical information about UAPs	$1,991 \times \text{number of UAPs to be executed or number of transactions to be executed}$
	Statistical information about SQL	$447 \times \text{number of SQLs to be executed}$
	Statistical information about SQL static optimization	$646 \times \text{SQL object cache mishit count}$
	Statistical information about SQL dynamic optimization	$380 \times \text{number of SELECT statements issued by OPEN or EXECUTE (including INSERT SELECT)}$
	Statistical information about SQL object execution	$520 \times \text{number of SQLs to be executed} \times \text{number of servers}$
	Statistical information about SQL object transfer	$389 \times \text{number of SQLs to be executed} \times \text{number of servers}$
	Statistical information about the history of SQL statements	$(240 + \text{average SQL length}) \times \text{number of SQLs to be executed}$
	Statistical information about CONNECT/DISCONNECT	$278 \times \text{number of CONNECTs and DISCONNECTs}$
	Statistical information about global buffer	$600 \times \text{number of synchronization points}$
	Statistical information about database manipulation for HiRDB files	$356 \times \text{number of synchronization points}$
	Statistical information about deferred write processing	$300 \times \text{number of deferred write operations}$

File type	Formula (bytes)
Statistical information about database I/O (DAT output)	$196 \times \text{total number of HiRDB files that make up the RDAREAs accessed at each acquisition time interval}$

#: Collection count = $\downarrow (\text{pdstend command input time} - \text{pdstbegin command input time}) \div \text{interval specified with the } -m \text{ option} \downarrow$

(1) Approximate size of a temporary work file

You can use the formula shown below to obtain an approximation of the size of a temporary work file that is created during execution of the statistics analysis utility. This formula assumes that all information in the statistics log file that is input to the utility is analyzed.

Size of temporary work file = size of statistics log file $\times a$ (bytes)

a

Use one of the values listed below, depending on the type of statistical information contained in the statistics log file. If the statistics log file contains more than one type of statistical information, use the largest value for *a*.

Type of statistical information	Value for <i>a</i>
System activity statistical information	3.4
SQL dynamic optimization information	135.6
Other	2

About errors

If the statistics log file contains multiple types of statistical information and one of them is SQL dynamic optimization information, the difference between the approximated value and the actual value might be large depending on the ratio of the SQL dynamic optimization information contained in the file. If you want to reduce this difference, make a separate calculation for SQL dynamic optimization information.

You can use the following formula to obtain the temporary work file size for SQL dynamic optimization information:

Size of temporary work file for SQL dynamic optimization information = number of SQL dynamic optimization information items \times 49,624 (bytes)

You can obtain the number of SQL dynamic optimization information items from the input log file summary information that is output when you execute the statistics analysis utility. For details about the input log file summary information, see *Statistics analysis utility (pdstedit)* in the manual *HiRDB Version 9 Command Reference*.

(2) Approximate size of a work file for sorting

You can use the formula shown below to obtain an approximation of the size of a work file for sorting that is created during execution of the statistics analysis utility. This formula assumes that all information in the statistics log file that is input to the utility is analyzed.

Size of work file for sorting = size of statistics log file $\times b$ (bytes)

b

Use one of the values listed below, depending on the type of statistical information contained in the statistics log file. If the statistics log file contains more than one type of statistical information, use the largest value for *b*.

Type of statistical information	Value for <i>b</i>
System activity statistical information	1.7
Other	1

(3) Approximate size of a DAT format file

You can use the formula shown below to obtain an approximation of the size of a DAT format file that is created during execution of the statistics analysis utility. This formula assumes that all information in the statistics log file that is input to the utility is analyzed.

$$\text{Size of DAT format file} = \text{size of statistics log file} \times 2 \text{ (bytes)}$$

19.1.4 File sizes required for the execution of the database condition analysis utility (pddbst)

The following table shows the formulas for determining the file sizes required for the execution of the database condition analysis utility (pddbst):

File type		Formula (bytes)
Work file	Physical analysis by RDAREA	$1 + 4.1 \times a$
	Logical analysis by RDAREA	a $1 + 4.1 \times \sum_{i=1}$ number of tables in RDAREA _{<i>i</i>} + number of indexes _{<i>i</i>} + number of LOB RDAREAs _{<i>i</i>})
	Accumulating condition analysis result or reorganization time prediction [#]	a $1 + 4.1 \times \sum_{i=1}$ number of tables in RDAREA _{<i>i</i>} + number of indexes _{<i>i</i>} + number of LOB RDAREAs _{<i>i</i>})
	Status analysis in units of tables	$1 + 4.1 \times (\text{number of storage RDAREAs})$
	Status analysis in units of indexes	$1 + 4.1 \times (\text{number of storage RDAREAs})$
	Cluster key status analysis	$1 + 4.1 \times (\text{number of storage RDAREAs})$
	Work area for sorting the above work files	Value obtained from the above formula $\times 2$

[#]: When pddbst -r ALL is specified, the number of resources in the dictionary RDAREAs as well as in the user RDAREAs must be added. For partitioned tables and indexes, add the number for each RDAREA.

a: Number of RDAREAs subject to analysis

19.1.5 File sizes required for the execution of the database copy utility (pdcopy)

The following table shows the formulas for determining the file sizes required for the execution of the database copy utility (pdcopy):

File type	Formula (bytes)
Backup file [#]	a
Full backup file [#]	$\sum \{28 \times c_i + (d_i + 28) \times (e_i + q_i)\}$

File type	Formula (bytes)
	$i = 1$ $+ 88 \times a + 220 \times b$
Differential backup file [#]	$w \times \text{size of full backup file}$
Differential backup management file	$(1 + j + m) \times 32,768$
Log point information file	1,024

[#]: If the backup file is larger than 2 GB, take one of the following actions:

- Use large files for backup files. For details about how to create large files, see 4.3 *Creating HiRDB file system areas*.
- Create multiple partitions, each of which is no larger than 2 GB, and specify multiple backup files.

a : Number of RDAREAs being backed up

b : Total number of HiRDB files in RDAREA being backed up

c_i : Number of unused pages in RDAREA being backed up

Assume 0 if you build the system before estimating.

- User RDAREAs
Determine after executing RDAREA unit status release (physical release) with the database release utility (pddbst command). The value is the resulting *total number of pages - number of used pages* of the *RDAREA page information*.
- User LOB RDAREAs
Determine after executing RDAREA unit status release (physical release) with the database release utility (pddbst command). The value is the resulting *Total number of segments - number of used segments* of the *RDAREA segment information*.

d_i : Page length of RDAREA being backed up

e_i : Number of pages used in RDAREA being backed up

Assume (number of segments in the RDAREAs being backed up \times segment size) if you build the system before estimating.

- User RDAREAs
Determine after executing RDAREA unit status release (physical release) with the database release utility (pddbst command). The value is the resulting *total number of pages - number of used pages* of the *RDAREA page information*.
- User LOB RDAREAs
Determine after executing RDAREA unit status release (physical release) with the database release utility (pddbst command). The value is the resulting *total number of segments - number of used segments* of the *RDAREA segment information*.

g : Length of a backup file name that is specified in the $-b$ option (bytes)

If multiple backup files are specified, this is the total length of the specified file names.

h : Number of backup files specified in the $-b$ option

j : $\uparrow (512 + 128 \times a) \div 32,700 \uparrow$ k : Number of consecutive differential backup operations

m : $\uparrow \{ \uparrow (256 + 128 \times a + g + 8 \times h) \div 256 \uparrow \times k \} \div 100 \uparrow$

qi : Number of directory pages in the RDAREAs being backed up

- User RDAREAs
 $6 \times (ti + 1) + 2 \times \uparrow (20,480 \div di) \uparrow + \{ \uparrow (si \div ui) \uparrow + \uparrow (si \div vi) \uparrow + 2 \times ti \}$
- User LOB RDAREA
 $7 + 3 \times (ti - 1) + \{ \uparrow (si \div 64,000) \uparrow + ti \} \times 96$

ri : Segment size of RDAREAs being backed up

si : Total number of segments in RDAREAs being backed up

This is the total number of segments for HiRDB files specified by the database initialization utility (`pdinit` command) or the `create rdarea` statement of the database configuration utility (`pdmod` command). If automatic extension is specified for the RDAREAs, add the number of extended segments.

ti : Number of HiRDB files of RDAREAs being backed up

ui : $\downarrow \{di - 20\} \div \{(\uparrow ri \div 32 \uparrow \times 8) + 56\} \downarrow$

vi : $\uparrow (125 \times di) \div (16 \times ui) \uparrow \times ui$

w : Percentage of all pages that are being updated on the RDAREAs being backed up

Determine w from the following formula (user RDAREA):

$$w = \left\{ \begin{array}{l} a \\ \sum_{i=1}^a X_i \div \\ \sum_{i=1}^a e_i \end{array} \right\} \times 1.2$$

X_i : Number of updated pages in RDAREAs being backed up

The number of updated pages refers to the number of pages that have been updated since the last time a differential backup was made. Calculate the number of updated pages for the tables and indexes stored in the RDAREAs being backed up based on the type of update SQL statements and the number of updated items, subject to the following conditions:

- INSERT

Based on 16.1 *Determining the size of a user RDAREA*, calculate the number of storage pages from the number of inserts, and add it to X_i . Calculate `PCTFREE` as 0.

- DELETE

Increase the value of X_i based on the following conditions:

Condition			Value added to X_i
Tables	Row length < di	Deleted rows are distributed over the entire table	MIN(number of deleted rows, number of table pages used)
		Deleted rows are concentrated on a few pages	MIN(number of deleted rows \div number of rows that can be stored on 1 page, number of table pages used)
	Row length > di		MIN(number of deleted rows \times number of pages required to store 1 row, number of table pages used)
Indexes	Updated keys are distributed over the entire index		MIN(number of update keys + α , number of index pages used)
	Updated keys are concentrated on a few pages		MIN(number of update keys \div number of keys that can be stored on 1 page + α , number of index pages used)

α : Number of duplicate keys in excess of 200

- UPDATE

Calculate the value to add to X_i based on the following conditions:

Condition			Value added to X_i
Tables	Update column length < di	Updated rows are distributed over the entire table	MIN(number of updated rows, number of table pages used)
		Updated rows are concentrated on a few pages	MIN(number of updated rows \div number of rows that can be stored on 1 page, number of table pages used)

Condition		Value added to X_i
	Update column length $> di$	MIN(number of updated rows \times number of pages required to store updated columns, number of table pages used)
Indexes	Updated keys are distributed over the entire index	MIN(number of update keys $\times 2 + \alpha \times 2$, number of index pages used)
	Updated keys are concentrated on a few pages	MIN(number of update keys $\times 2 \div$ number of keys that can be stored on 1 page $+ \alpha \times 2$, number of index pages used)

α : Number of duplicate keys in excess of 200

- After regenerating

Calculate β as the regenerated tables or indexes that are stored on RDAREAs being backed up.

β = number of used pages of regenerated tables or indexes + number of used segments of regenerated tables or indexes $\div ui$ + number of used segments of regenerated tables or indexes $\div vi$

Calculate β only for regenerated tables and indexes, and add the value to X_i .

- PURGE

Calculate γ as the tables or indexes on which PURGE was performed that are stored in RDAREAs being backed up.

γ = number of used segments of purged tables or indexes $\div ui$ + number of used segments of purged tables or indexes $\div vi$

Calculate γ only for regenerated tables and indexes, and add the value to X_i .

19.1.6 File sizes required for the execution of the dictionary import/export utility (pdexp)

The following table shows the formulas for determining the file sizes required for the execution of the dictionary import/export utility (pdexp):

File type		Formula (bytes)
Export file	Base table	$0.7 + 0.5 \times CMN + (0.1 \times \lceil CMN \div 10 \rceil)$ DEF $+ \sum_{n=1} (0.1 + DS_n) + 0.1 \times \lceil LRD \div 10 \rceil$ $+ 0.6 \times \lceil DIV \div 10 \rceil + 2.8 \times REF$ CHK $+ \sum_{n=1} (0.1 + 1.0 \times \lceil CS_n \div 10 \rceil)$ IDX $+ \sum_{n=1} (2.7 + 0.2 \times IR_n)$
	View table	$0.6 + 0.4 \times CMN + (0.1 \times \lceil CMN \div 10 \rceil) + e$
	Procedure	$0.6 + 0.1 \times g + h$

CHK : Number of check constraints ($0 \leq CHK \leq 254$)

CMN : Number of table columns ($1 \leq CMN \leq 30,000$)

CS_n : Size of search conditions of n th check constraint ($0 \leq CS_n \leq 2,000,000$)

DEF : Number of default value definition columns ($0 \leq DEF \leq 30,000$)

DIV : Number of partition conditions ($0 \leq DIV \leq 4,096$)

DSn: Default size of *n*th default column ($1 \leq DS_n \leq 64,003$)

IDX: Number of indexes ($0 \leq IDX \leq 254$)

IRn: Number of RDAREAs for storing *n*th index ($0 \leq IR_n \leq 4,096$)

LRD: Number of LOB RDAREAs ($0 \leq LRD \leq 4,096$)

REF: Number of reference constraints ($0 \leq REF \leq 255$)

e: Length of the source during view table definition (KB)

g: Number of resources used by a stored procedure that is exported[#]

This is the value of the `N_RESOURCE` column in the `SQL_ROUTINES` table.

h: Length of the source of a stored procedure (KB)

This is the value of the `SOURCE_SIZE` column in the `SQL_ROUTINES` table.

[#]: If exporting multiple tables, determine the previously described sizes for each table. The sum of the sizes obtained is the size of the export file.

19.1.7 File sizes required for the execution of the optimizing information collection utility (pdgetcst)

The following table shows the formulas for determining the file sizes required for the execution of the optimizing information collection utility (pdgetcst):

File type	Formula (bytes)
Parameter file that contains optimized information	$162 + 405 \times a + 567 \times b + 162 + 324 \times b \times g$
Output results file	Collecting optimizing information by retrieval (-c 1v11 specified): $202 + 131 \times e$ Collecting optimizing information by retrieval (-c 1v12 specified): $370 + 561 \times d + 196 \times d$ Registering optimizing information using the parameter file that contains optimized information: $370 + 235 \times c + 387 \times f + 196 \times g$

a: Number of specified indexes

b: Number of specified columns

c: Number of indexes defined in table

d: Number of intervals (total of number of sectors for all indexes)

e: Number of tables

f: Number of columns in table

g: Number of sectors (total of number of sectors specified in all column definitions)

19.1.8 File sizes required for the execution of the access path display utility (pdvwopt)

The following table shows the formulas for determining the file sizes required for the execution of the access path display utility (pdvwopt):

File type	Formula (bytes)
Access path information file	<p>● HiRDB single server configuration</p> $240 + 2 \times \left\{ \begin{array}{l} 60 + \text{length of SQL} \\ + \sum_{i=1}^a \left\{ \begin{array}{l} 100 + \sum_{k=1}^{c_{ij}} (160 + e_{ijk} \times 20 + f_{ijk} \times 70) \\ + d_{ij} \times 180 \end{array} \right\} \end{array} \right\}$ <p>● HiRDB parallel server configuration</p> $140 + \sum_{i=1}^a \left\{ \begin{array}{l} 60 + \text{length of SQL} \\ + \sum_{j=1}^{b_i} \left\{ \begin{array}{l} 110 + \sum_{k=1}^{c_{ij}} (270 + e_{ijk} \times 20 + f_{ijk} \times 70) \\ + d_{ij} \times 290 \end{array} \right\} \end{array} \right\}$

a : Number of retrieval SQLs

b_i : Number of queries in SQL

c_{ij} : Number of tables in query

d_{ij} : Number of join processes in query

e_{ijk} : Number of table storage RDAREAs

f_{ijk} : Number of table index definitions

19.1.9 File sizes required for execution of the rebalancing utility (pdrbal)

The following table shows the formulas for determining the file sizes required for the execution of the rebalancing utility (pdrbal):

File type	Formula (bytes)
Index information file	<p>B-tree index: $(K + d) \times N + 512$</p> <p>Plug-in index: $(12 + Y) \times N + 1,024$</p> <p>These formulas are for the size of one index. If there are multiple indexes, determine the size of each index.</p>
Work file ^{#1}	$(3 + 4 \times I \times R) \times 200$
Work file for sorting ^{#2}	<p>Condition 1: Size of index information file + $4 \times N$</p>

File type	Formula (bytes)
	Condition 2: $\{\text{Size of index information file} + 4 \times N\} \times 2$ <ul style="list-style-type: none"> Condition 1 When $1,024 \text{ (KB)} \geq E$ Condition 2 When $1,024 \text{ (KB)} < E$ <i>E</i> : Buffer size The buffer size obtained according to 19.1.12 Buffer size used to determine the size of the work file for sorting.
Execution results output file	$1,000 + \text{number of table storage RDAREAs} \times 200$

d: If all key component columns are fixed-length, the value is 10; if they include a variable-length column, the value is 12.

I: Number of indexes

K: Index key length

See Table 16-5 List of index key lengths. For variable-length data, keep in mind when defining the maximum length that single columns are also handled as multicolumns.

N: Number of rows to be moved by rebalancing (for a repetition column, number of rows \times number of elements)

R: Number of partitioned RDAREAs for table or index

Y: Value as follows

- 27 for the abstract data type stored in the LOB RDAREA
- Key length + 2 for the abstract data type of a maximum of 255 bytes of definition length
- 2 for the abstract data type of 256 bytes or more of definition length

Typical abstract data type values are shown below.

- 27 for the SGMLTEXT type
- 2 for FREEWORD, GEOMETRY, and XML type

#1

Output when `lv12` is specified in the `-m` option.

#2

This file is not needed for a plug-in index.

19.1.10 File sizes required for execution of the integrity check utility (pdconstck)

The following table shows the formulas for determining the file sizes required for execution of the integrity check utility (pdconstck).

File type	Formula (bytes)
Process result file	<code>-k set/release</code> $560 + (REF + 1) \times 70 + (CHK + 1) \times 70$ <code>-k check</code> $700 + (REF + 1) \times 70 + (CHK + 1) \times 70$ REF $+ \sum_{n=1} (490 + (RCn \times 70))$

File type	Formula (bytes)
	$GEN \times ROW$ $+ \sum_{m=1} \left(\sum_{l=1} (RC_{nml} \times 70) \right)$ CHK $+ \sum_{n=1} (490 + (CC_n \times 70))$ $GEN \times ROW$ $+ \sum_{m=1} \left(\sum_{l=1} (CC_{nml} \times 70) \right)$

REF: Number of referential constraints defined for the table

CHK: Number of check constraints defined for the table

RC: Number of referential constraint columns containing foreign keys

CC: Number of column in the search condition of check constraint

GEN: 1 if the inner replica facility is not used

If the inner replica facility is used, the number of generations (1 to 10) + 1 in which the replica RDAREA of the table exists

ROW: Upper limit of the number of outputs of the key value that caused a constraint error (value specified by the *-w* option)

19.1.11 File sizes required for execution of parallel loading (pdparaload)

The following table shows the formulas for calculating the sizes of files used in parallel loading (pdparaload).

File type	Formula (bytes)
pdload control statement file	$(\text{pdparaload control statement file size} + 200) \times R$
Files created by pdload [#]	$RDAREAs \text{ that make up table}$ $\sum_{i=1} (\text{file sizes required to execute data loading by RDAREA})$

R: Number of RDAREAs that constitute the table

[#]

The *pdparaload* command executes as many data loads in RDAREA units (*pdload*) internally as there are RDAREAs that constitute the table. For this reason, *pdparaload* uses as many files required for execution of data loads in RDAREA units as there are RDAREAs that constitute the table. For details about the sizes of files required for execution of data loads by RDAREA, see *19.1.1 File sizes required for the execution of the database load utility (pdload)*.

19.1.12 Buffer size used to determine the size of the work file for sorting

This subsection lists and describes the formulas used to calculate the buffer size for sorting.

- 32-bit mode HiRDB:

$$\text{Buffer size (bytes): } \frac{R+7}{2} + \sqrt{(B+8) \times n \times A + \frac{(R+7)^2}{4}} + C$$

- 64-bit mode HiRDB:

$$\text{Buffer size (bytes): } \frac{R+15}{2} + \sqrt{(B+8) \times n \times A + \frac{(R+15)^2}{4}} + C$$

n : Number of data items to be processed

- For `pdload`: Sum of existing data in the table and additional data to be loaded
- For `pdrorg`: Number of data items to be unloaded
- For `pdrbal`: Number of data items to be rebalanced

For a repetition column, the number of data items is the number of elements, not the number of rows.

k : Key length (maximum value). For details about the formula for obtaining the key length, see 16.1.3(3) *Examples of calculating the number of index storage pages*.

x : If all key component columns are fixed length, the value is 10. If they include a variable length column, the value is 12.

c : Number of index component columns

y : Linux edition: 2, otherwise: 1

z : For a variable-length multicolumn index: $c \times 4$, otherwise: 0

K : For a variable-length multicolumn index: $k + c + 8$, otherwise: $k + 12$

N : For a variable-length multicolumn index: $(c \times 2) + y$, otherwise: $3 + y$

R : $k + x + z$

A : 32-bit mode HiRDB: $R + (K + 8) + 28$, 64-bit mode HiRDB: $R + (K + 8) + 56$

B : 32-bit mode HiRDB: $R + (K + 8) + 56$, 64-bit mode HiRDB: $R + (K + 8) + 104$

C : 32-bit mode HiRDB: $2,092 + (N \times 32) + (K + 8)$, 64-bit mode HiRDB: $2,112 + (N \times 32) + (K + 8)$

19.2 Determining the memory size required for utility execution

19.2.1 Memory size required for the execution of the database initialization utility (pdinit)

The following tables show the formulas for determining the memory sizes required for the execution of the database initialization utility (pdinit).

(1) HiRDB single server configuration

Condition	Formula for determining memory size (KB)
32-bit mode	$\uparrow \{$ $61,440 \times (140 \times a + 20 \times b + c) \div 61,432 + 6,004 \times d \div 500$ $+ 36,008 \times a \div 1,000 + 468 \times e + 403,888$ $\} \uparrow \div 1,024 + 267$
64-bit mode	$\uparrow \{$ $61,448 \times (144 \times a + 24 \times b + c) \div 61,436 + 8,004 \times d \div 500$ $+ 36,016 \times a \div 1,000 + 468 \times e + 405,888$ $\} \uparrow \div 1,024 + 267$

a: Total number of RDAREAs

b: Number of HiRDB files in all RDAREAs

c: Sum of the lengths of all HiRDB file names

d: Total number of authorization identifiers

e: Number of RDAREAs for dictionary server

(2) HiRDB parallel server configuration

Condition	Formula for determining memory size (KB)
32-bit mode	DS
	$\uparrow \{$ $\{61,440 \times (140 \times a + 20 \times b + c)\} \div 61,432 + (6,004 \times d) \div 500$ $+ (36,008 \times a) \div 1,000 + 468 \times e + 403,888 + 348 \times f + 344 \times g$ $\} \uparrow \div 1,024 + 268$
	BES
64-bit mode	$\uparrow (4 \times b + 237,220) \div 1,024 \uparrow + 268$
	MGR
	10
64-bit mode	DS
	$\uparrow \{$ $\{61,448 \times (144 \times a + 24 \times b + c)\} \div 61,436 + (8,004 \times d) \div 500$ $+ (36,016 \times a) \div 1,000 + 468 \times e + 405,888 + 348 \times f + 344 \times g$ $\} \uparrow \div 1,024 + 268$
	BES
	$\uparrow (4 \times b + 245,744) \div 1,024 \uparrow + 268$
	MGR
	10

a: Total number of RDAREAs

- b : Number of HiRDB files in all RDAREAs
 c : Sum of the lengths of all HiRDB file names
 d : Total number of authorization identifiers
 e : Number of RDAREAs for dictionary server
 f : Total number of back-end servers
 g : Sum of the values of $(144 \times a + 24 \times b + c) \div 7,780$ for all back-end servers

19.2.2 Memory size required for the execution of the database definition utility (pddef)

The following table shows the formulas for determining the memory size required for the execution of the database definition utility (pddef):

Condition	Formula for determining memory size (KB)
HiRDB single server configuration	1,956 (or 1,957 in the 64-bit mode)
HiRDB parallel server configuration	1,956 (or 1,957 in the 64-bit mode)

19.2.3 Memory size required for the execution of the database load utility (pload)

The following tables show the formulas for determining the memory size required for the execution of the database load utility (pload). For details about the variables, see (3) *Variables used in the formulas*.

(1) HiRDB single server configuration

Condition		Formula for determining memory size (KB)
32-bit mode	Single server	$6,352 + \lceil (\alpha + \gamma) \div 1,024 \rceil$
	Utility special unit [#]	$1,968 + \lceil (\beta + \delta) \div 1,024 \rceil$
64-bit mode	Single server	$12,172 + \lceil (\alpha + \gamma) \div 1,024 \rceil$
	Utility special unit [#]	$2,423 + \lceil (\beta + \delta) \div 1,024 \rceil$

[#]: If no utility special unit is being used, use the value for a single server.

(2) HiRDB parallel server configuration

Condition		Formula for determining memory size (KB)
32-bit mode	MGR	$2,255 + \lceil \alpha \div 1,024 \rceil$
	Server machine containing input files	$2,045 + \lceil (\beta + \delta) \div 1,024 \rceil$
	BES [#]	$3,762 + \lceil \gamma \div 1,024 \rceil$
64-bit mode	MGR	$2,575 + \lceil \alpha \div 1,024 \rceil$
	Server machine containing input files	$2,406 + \lceil (\beta + \delta) \div 1,024 \rceil$
	BES [#]	$8,247 + \lceil \gamma \div 1,024 \rceil$

#: If a single server machine has multiple back-end servers, add into the calculation only for the number of back-end servers.

(3) Variables used in the formulas

α (bytes):

$$\{3,056 + A + B + (516 \times a) + (572 \times b) + (312 \times c) + (144 \times d) + (8 \times e) + (1,032 \times f) + (44 \times g) + (272 \times h) + (224 \times i) + (44 \times j) + (60 \times k) + (260 \times m) + (56 \times n) + (196 \times p) + (236 \times q) + (744 \times r) + (620 \times s)\} \times 2$$

β (bytes):

$$\{6,908 + \alpha + (C \times t) + K + (48 \times a) + (22 \times b) + (8 \times e) + (240 \times i) + (48 \times j) + (4 \times k) + (224 \times m) + (47,416 \times t) + (1,032 \times u) + (4 \times v)\}$$

γ (bytes):

$$\{37,700 + (\alpha \div 2) + C + D + F + H + P + Q + T + (80 \times a) + (1,871 \times b) + (120 \times c) + (26 \times g) + (1,532 \times i) + (36 \times j) + (44 \times k) + (1,212 \times m) + (40 \times n) + (344 \times p) + (30 \times q) + (16 \times u) + (88 \times v) + (20 \times w)\}$$

δ (bytes):

$$\{69,436 + \alpha + D + K + E + L + M + N + S + (U \times 2) + 8 + (48 \times a) + (32 \times a) + (88 \times c) + (4 \times g) + (2,156 \times k) + (24 \times t) + (1,024 \times u) + (4 \times v) + (50 \times y) + (50 \times z)\}$$

a : Number of columns

b : Number of columns with abstract data type

c : Number of parameters in constructor or reverse constructor function

d : Number of file path names specified in command line or control information file

e : Number of LOB middle files

f : Number of LOB files by the column that are specified

g : Number of table storage RDAREAs

h : Number of table row partitioning conditions

i : Number of indexes

j : Number of index storage RDAREAs

k : Number of BLOB-type columns

m : Number of plug-in indexes

n : Number of user LOB RDAREAs storing LOB-attribute abstract data type

p : Number of functions provided by plug-in

q : Number of function parameters provided by plug-in

r : Number of data-type plug-ins

s : Number of index-type plug-ins

t : Number of servers storing tables

u : Number of BLOB-type parameters among the constructor function parameters used

v : Number of user LOB RDAREAs storing LOB columns

w : Number of user LOB RDAREAs storing plug-in indexes

y : Number of BINARY columns

This is the BINARY-data columns in input data that have been excluded from processing by the `skipdata` control statement plus the number of columns actually defined in the tables.

z : Number of BINARY-attribute parameters for plug-in-provided functions

This is the number of BINARY data columns of the plug-in-provided functions in input data that have been excluded from processing by the `skipdata` control statement, plus the number of columns actually defined in the tables.

A: Total file size specified in the command line

B: Total length of file path names specified in the command line and control information file

C: If the following condition is satisfied, the value is $(pd_utl_buff_size \times 1,024 + 4,096) \times 2$; otherwise, the value is 0:

- HiRDB single server configuration
A utility special unit is used.
- HiRDB parallel server configuration
The `pdload` command is executed with a back-end server name containing the table storage RDAREA that is different from the server name specified in the `source` statement; or, the `pdrorg` command specifying the `-g` option is executed.

D: Row length

This is the sum of the defined lengths of all columns that constitute the table. The length of a BINARY-type row is the defined length for `pdload` plus MIN(defined length, 32,500) for `pdrorg` (in bytes). For a non-FIX table, add $(a + 1) \times 4$.

E: Memory required for EasyMT

Add this value if you have specified `easymt` in the `-f` option.

F: $550 \times 1,024 + 1,024 \times 1,024 + G$

Add this value if you have specified `c` in the `-i` option.

G: 256 (32-bit mode) or 512 (64-bit mode)

H: Value specified for the batch input/output local buffers \times RDAREA page length $\times J$ + value specified for the random access local buffers \times RDAREA page length

Add this value if you have specified the `-n` option. If an RDAREA's page length varies from one partitioned RDAREA to another, use the longest page length for this calculation.

J: Determine from the following table:

Value specified with -n option	Table partition type				
	Non-partitioning table	Key range partitioning table	Hash partitioning table		
			Rebalancing hash (HASHA-HASHF)		Non-rebalancing hash (HASH0-HASH6, HASHZ)
			FIX hash	Flexible hash	
<code>div</code> specified	1	Number of row partitions in the server for the table	HiRDB single server configuration: 1,024 HiRDB parallel server configuration: $(\uparrow 1,024 \div g \uparrow) \times$ (number of table storage RDAREAs on the server)	Number of row partitions in the server for the table	Number of row partitions in the server for the table
<code>div</code> not specified		1			1

K: Parameter length

Sum of the lengths of arguments in the constructor function that is used to generate values for the abstract data type. For a BLOB-type parameter, the parameter length is 8 bytes.

L: If all the following conditions are satisfied, add 1 or 2; otherwise, the value is 0:

- The `errdata` operand is specified in the `source` statement.
- A utility special unit is used (HiRDB single server configuration), or the server name specified in the `source` statement is not the name of the back-end server that contains the table storage RDAREA (HiRDB parallel server configuration).
- The table contains an abstract data type column, or a unique index is defined.

1. `errwork` operand specified: Value of `errwork` operand $\times 1,024$

2. `errwork` operand omitted: Value of `pd_utl_buff_size` \times 1,024 \times 3 \times t

M: Memory required for UOC

Add this value if you use UOC.

N: When the `maxreclen` operand is specified, calculate the following value:

If the input data file is in extended DAT format:

Value specified in `maxreclen` operand \times 1,024

If the input data file is in DAT format:

Value specified in `maxreclen` operand \times 1,024 \times 3

When the table has `BINARY` type columns, or when the input data files are in binary format:

Add the smaller of the following to the calculation:

- Value specified in `maxreclen` operand \times 1,024
- Variable D (row length)

When other than the above:

0

P: Memory required for plug-ins

Add this value if there is an abstract data type column provided by a plug-in. For details about the size of memory required by plug-ins, see the applicable plug-in documentation.

If the constructor functions' arguments are `BLOB` or `BINARY` type, add (actual parameter length stored in all abstract data types \times 2) that is defined per row.

Q: Memory requirement for output buffer

If the specified index creation method is the batch index creation mode or the index information output mode and the following condition is satisfied, add 2 megabytes:

- Number of table partitions \times number of index definitions $>$ maximum number of processes that can be open - 576

S: $(4 \times \text{Buffer length} + 1.1) \times 1,024$

The buffer length is as follows (round up the value in units of 32 kilobytes).

- The value specified in the `option` statement `file_buff_size` of the database load utility (`pdload`).
- If the above value is not specified: 1,024.

T: If there are compressed columns, split compression size \times 2 + RDAREA's page length; if there are no compressed columns, 0

For the split compression size, use the largest value among all the compressed columns. If the page length varies from one RDAREA to another for a row-partitioned table, use the longest page length for this calculation.

U: Row length of input file (bytes)

DAT format: Value specified for the `maxreclen` operand \times 1,024 (or 32,768 if the `maxreclen` operand is not specified)

Fixed-length format: Row length of input file

Binary format: 0

19.2.4 Memory size required for the execution of the database reorganization utility (pdorg)

The following tables show the formulas for determining the memory size required for the execution of the database reorganization utility (`pdorg`). For details about the variables, see (3) *Variables used in the formulas* as follows.

(1) HiRDB single server configuration

Condition		Formula for determining memory size (KB)
32-bit mode	Single server	$7,536 + \lceil (\alpha + \gamma) \div 1,024 \rceil$

Condition		Formula for determining memory size (KB)
64-bit mode	Utility special unit [#]	$2,119 + \lceil (\beta + \delta) \div 1,024 \rceil$
	Single server	$13,196 + \lceil (\alpha + \gamma) \div 1,024 \rceil$
	Utility special unit [#]	$2,593 + \lceil (\beta + \delta) \div 1,024 \rceil$

[#]: If no utility special unit is being used, use the value for a single server.

(2) HiRDB parallel server configuration

Condition			Formula for determining memory size (KB)
32-bit mode	MGR		$2,200 + \lceil \alpha \div 1,024 \rceil$
	-g option omitted	DS	$1,940 + \lceil \beta \div 1,024 \rceil$
		BES ^{#1, #2}	$4,976 + \lceil (\gamma + \delta) \div 1,024 \rceil$
	-g option specified	Server machine containing unload data file	$1,940 + \lceil (\beta + \delta) \div 1,024 \rceil$
		BES ^{#2}	$4,976 + \lceil \gamma \div 1,024 \rceil$
64-bit mode	MGR		$2,569 + \lceil \alpha \div 1,024 \rceil$
	-g option omitted	DS	$2,294 + \lceil \beta \div 1,024 \rceil$
		BES ^{#1, #2}	$9,441 + \lceil (\gamma + \delta) \div 1,024 \rceil$
	-g option specified	Server machine containing unload data file	$2,294 + \lceil (\beta + \delta) \div 1,024 \rceil$
		BES ^{#1}	$9,441 + \lceil \gamma \div 1,024 \rceil$

^{#1}: If dictionary tables are regenerated, add to the calculation the server machines that have dictionary servers.

^{#2}: If a single server machine has multiple back-end servers, add into the calculation only for the number of back-end servers.

(3) Variables used in the formulas

α (bytes):

$$\{2,592 + A + B + (116 \times a) + (260 \times b) + (6 \times c) + (272 \times d) + (44 \times g) + (272 \times h) + (224 \times i) + (44 \times j) + (60 \times k) + (260 \times m) + (56 \times n) + (196 \times p) + (236 \times q) + (744 \times r) + (620 \times s) + (24 \times t)\} \times 2$$

β (bytes):

$$\{40,940 + \alpha + (C \times t) + (D \times t) + (136 \times a) + (56 \times g) + (2,200 \times j) + (4 \times k) + (548 \times t)\}$$

γ (bytes):

$$\{101,140 + (\alpha \div 2) + C + (D \times 2) + F + H + P + Q + T + 64,010 + (48 \times a) + (128 \times a) + (1,949 \times b) + (120 \times c) + (154 \times g) + (336 \times i) + (216 \times j) + (32,056 \times k) + (1,212 \times m) + (131,224 \times n) + (344 \times p) + (30 \times q) + (20 \times w)\}$$

δ (bytes):

$$\{33,104 + \alpha + (K \times 2) + E + S + 72 + (48 \times a) + (204 \times a) + (688 \times b) + (306 \times c) + (44 \times g) + (272 \times h) + (224 \times i) + (44 \times j) + (56 \times n) + (716 \times r) + (152 \times v)\}$$

Notes

- Variables are described in (3) *Variables used in the formulas* of 19.2.3 *Memory size required for the execution of the database load utility (pdload)*.
- If one command is used to process multiple tables for the purpose of reorganizing dictionaries or reorganizing tables in units schemas, use the total size of all such tables for variables *a* through *z*.

19.2.5 Memory size required for the execution of the database structure modification utility (pdmod)

The following tables show the formulas for determining the memory size required for the execution of the database structure modification utility (pdmod).

(1) HiRDB single server configuration

Condition	Formula for determining memory size (KB)
32-bit mode	$\uparrow \{$ $4 \times a + 56,016 \times b + 53,016 \times c + 2,440 \times d + 1,724 \times e$ $+ (94,008 \times f) \div 500 + (4,008 \times g) \div 1,000 + 440,720 + h + i + j + k$ $\} \div 1,024 \uparrow + 9.8$
64-bit mode	$\uparrow \{$ $4 \times a + 56,024 \times b + 53,024 \times c + 3,040 \times d + 1,736 \times e$ $+ (100,016 \times f) \div 500 + (4,012 \times g) \div 1,000 + 450,720 + h + i + j + k$ $\} \div 1,024 \uparrow + 9.8$

a: Value of `pd_max_rdarea_no`

b: Number of indexes in local RDAREAs during the execution of `initialize rdarea` statement + number of indexes in remote RDAREAs

c: Total number of LOB columns during the execution of `initialize rdarea` statement

d: Total number of LOB-attribute abstract data types during the execution of `initialize rdarea` statement

e: Total number of plug-in columns and plug-in indexes during the execution of `initialize rdarea` statement

f: Total number of abstract data types during the execution of `initialize rdarea` statement

g: Total number of `ASSIGN LISTs` for the tables stored in the local RDAREAs during the execution of `initialize rdarea` statement

h: $8 \times a + 30,720$

Add this value if the `alter HiRDB mode to parallel` statement is used to migrate from a HiRDB single server configuration to a HiRDB parallel server configuration.

i: 46,744

Add this value if the `create rdarea` statement is used to add a data dictionary LOB RDAREA.

j: 88,064

Add this value if the `alter system` statement is used to change the dictionary table's reference privilege.

k: 54,732

Add this value if the `alter system` statement is used to change the dictionary table's column attribute to `MCHAR`.

(2) HiRDB parallel server configuration

Condition	Formula for determining memory size (KB)
32-bit mode	$\uparrow \{$ $4 \times a + 56,016 \times b + 53,016 \times c + 2,440 \times d + 1,724 \times e$ $+ (94,008 \times f) \div 500 + (4,008 \times g) \div 1,000 + 440,720 + h + i + j$ $+ 108,428 \times m$ $\} \div 1,024 \uparrow$
	$\uparrow (4 \times a + 252,755 + k) \div 1,024 \uparrow$

Condition		Formula for determining memory size (KB)
64-bit mode	FES	0.52
	MGR	9.8
	DS	$\uparrow \{$ $4 \times a + 56,024 \times b + 53,024 \times c + 3,040 \times d + 1,736 \times e$ $+ (100,016 \times f) \div 500 + (4,012 \times g) \div 1,000 + 450,720 + h + i + j$ $+ 108,432 \times m$ $\} \div 1,024 \uparrow$
	BES	$\uparrow (4 \times a + 261,112 + k) \div 1,024 \uparrow$
	FES	0.53
	MGR	9.8

a: Value of `pd_max_rdarea_no`

b: Number of indexes in local RDAREAs during the execution of `initialize rdarea` statement + number of indexes in remote RDAREAs

c: Total number of LOB columns during the execution of `initialize rdarea` statement

d: Total number of LOB-attribute abstract data types during the execution of `initialize rdarea` statement

e: Total number of plug-in columns and plug-in indexes during the execution of `initialize rdarea` statement

f: Total number of abstract data types during the execution of `initialize rdarea` statement

g: Total number of `ASSIGN LISTS` for the tables stored in the local RDAREAs during the execution of `initialize rdarea` statement

h: 46,744

Add this value if the `create rdarea` statement is used to add a data dictionary LOB RDAREA.

i: 88,064

Add this value if the `alter system` statement is used to change the dictionary table's reference privilege.

j: 54,732

Add this value if the `alter system` statement is used to change the dictionary table's column attribute to `MCHAR`.

k: 2,200

Add this value if the `initialize rdarea` statement is executed.

m: If the `move rdarea` statement is executed, add the following to the calculation (if `move rdarea` is not executed, use 0):

$$\uparrow (192 \times \text{number of moved RDAREAs} + 160 \times \text{total number of HiRDB files on moved RDAREAs} + 136 \times \text{total number of HiRDB files on moved RDAREAs that are replica RDAREAs} + 8 \times \text{total number of tables stored on moved RDAREAs} + 8 \times \text{total number of indexes stored on moved RDAREAs} + 8 \times \text{total number of LOB columns stored on moved RDAREAs}) \div 102,400 \uparrow$$

19.2.6 Memory size required for the execution of the statistics analysis utility (pdstedit)

The following table shows the formulas for determining the memory size required for the execution of the statistics analysis utility (pdstedit).

Condition	Formula for determining memory size (KB)
HiRDB single server configuration	$16,384^{\#} + 1.5 \times \text{number of hosts} \times \text{number of HiRDB servers} \times \text{number of UAPs}$

Condition	Formula for determining memory size (KB)
HiRDB parallel server configuration	

#: For the 64-bit mode, the value is 18,432.

19.2.7 Memory size required for the execution of the database condition analysis utility (pddbst)

The following subsections show the formulas for determining the memory size required for the execution of the database condition analysis utility (pddbst). For details about the variables used in the formulas, see (3) *Variables used in the formulas*.

(1) For a HiRDB single server configuration

Condition	Formula for determining memory size (KB)
32-bit mode	$9,425 + \lceil \{(\alpha + \beta) \div 1,024\} \rceil$
64-bit mode	$15,311 + \lceil \{(\alpha + \beta) \div 1,024\} \rceil$

(2) For a HiRDB parallel server configuration

Condition		Formula for determining memory size (KB)
32-bit mode	MGR	$5,117 + \lceil \{\alpha \div 1,024\} \rceil$
	DS	$4,177 + \lceil \{\beta \div 1,024\} \rceil$
	BES#	
64-bit mode	MGR	$5,645 + \lceil \{\alpha \div 1,024\} \rceil$
	DS	$8,329 + \lceil \{\beta \div 1,024\} \rceil$
	BES#	

#: The value must be obtained for each back-end server that contains the tables and indexes subject to analysis.

(3) Variables used in the formulas

α (bytes):

$$\{28,000 + A + (10,592 \times a) + (10,592 \times b) + (10,592 \times c) + (2,264 \times d) + (848 \times e) + (272 \times f) + (432 \times g) + (304 \times h)\}$$

β (bytes):

$$\{100,000 + (1,024 \times d) + (2,784 \times i) + (2,784 \times j) + (2,784 \times k)\}$$

Note

- For details about other variables, see (3) *Variables used in the formulas* in 19.2.3 *Memory size required for the execution of the database load utility (pdload)*.

19.2.8 Memory size required for the execution of optimizing the information collection utility (pdgetcst)

The following table shows the formulas for determining the memory size required for the execution of the optimizing information collection utility (pdgetcst).

Condition		Formula for determining memory size (KB)
HiRDB single server configuration		$6,060 + 0.1 \times \text{number of table storage RDAREAs} + 0.07 \times \text{number of index storage RDAREAs} + 1.0 \times \text{number of indexes} + 0.04 \times \text{number of tables in schema} + 1.0 \times \text{number of servers} + 11 \times \text{number of columns}$
HiRDB parallel server configuration	BES	$1,813 + 0.06 \times \text{number of table storage RDAREAs} + 0.05 \times \text{number of index storage RDAREAs} + 0.03 \times \text{number of indexes}$
	MGR	$3,336 + 0.1 \times \text{number of table storage RDAREAs} + 0.07 \times \text{number of index storage RDAREAs} + 1.0 \times \text{number of indexes} + 0.04 \times \text{number of tables in schema} + 1.0 \times \text{number of servers} + 11 \times \text{number of columns}$
	DS	16,402

Note

Other than these sizes, add the memory size used by the following SQL:

```
SELECT internal-information#, index-first-configuration-column-name FROM
authorization-identifier, table-identifier ORDER BY primary-index-component-
column-name WITHOUT LOCK NOWAIT;
```

```
SELECT FLOAT (COUNT (*)) FROM authorization-identifier.table-identifier WITHOUT LOCK NOWAIT;
```

```
SELECT FLOAT (COUNT (primary-index-component-column-name) ) FROM authorization-identifier.table-
identifier WITHOUT LOCK NOWAIT;
```

```
ALTER TABLE authorization-identifier.table-identifier CHANGE LOCK ROW;
```

#: The internal information is 12 bytes. Therefore, estimate as though an SQL statement were issued to retrieve a 12-byte character column (CHAR (12)), in addition to the index first configuration column name.

For details about the size of memory required by SQL, see the following sections:

- For a HiRDB single server configuration
 - 15.1.6 Formulas for size of memory required during SQL execution
 - 15.1.7 Formula for size of memory required during SQL preprocessing
- For a HiRDB parallel server configuration
 - 15.2.6 Formulas for size of memory required during SQL execution
 - 15.2.7 Formula for size of memory required during SQL preprocessing

19.2.9 Memory size required for the execution of the database copy utility (pdcopy)

The following tables show the formulas for determining the memory size required for the execution of the database copy utility (pdcopy).

(1) HiRDB single server configuration

Condition	Formula for determining memory size (KB)
Single server	$88 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$ $+ 49 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) + 64$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$ Add the following value if this server machine contains backup files: $+ 63 + \text{number of backup files} \times (2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) \times 2$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100 + 1,024$

Condition	Formula for determining memory size (KB)
	Add the following value if differential backup files are to be collected: $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32,768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32,768 \uparrow \times 32 \times 2$
Utility special unit	$63 + \text{number of backup files} \times (2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) \times 2$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100 + 1,024$

a: Length of the backup file name specified in the `-b` option (bytes). If multiple backup files are specified, this value is the total length of the file names.

b: Number of backup files specified in the `-b` option

(2) HiRDB parallel server configuration

Condition	Formula for determining memory size (KB)
MGR	$88 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$ Add the following value if differential backup files are to be collected: $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32,768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32,768 \uparrow \times 32 \times 2$
DS	$49 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) + 64$
BES	$+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$
Server machine containing backup files	$63 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $\times (\text{number of servers subject to backup} + 1)$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100 + 1,024$

a: Length of the backup file name specified in the `-b` option (bytes). If multiple backup files are specified, this value is the total length of the file names.

b: Number of backup files specified in the `-b` option

19.2.10 Memory size required for the execution of the database recovery utility (pdrstr)

The following tables show the formulas for determining the memory size required for the execution of the database recovery utility (pdrstr).

(1) HiRDB single server configuration

Condition	Formula for determining memory size (KB)
Single server	$65 + \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 50$ $+ 98 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + c$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$

Condition	Formula for determining memory size (KB)
	$+ \{(\text{number of RDAREAs subject to recovery} + 99) \div 100\} \times 5$ Add the following value if this server contains backup files: $+ 100 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100 + 1,024$ If you are inputting the unload log file or system log file, add: $+ 57 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + 64$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100 + d$ $+ 0.6 \times \text{number of RDAREAs subject to recovery} + \text{size of work buffer for sorting (value of -y option)}$ Add the following value if differential backup files are used for recovery: $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32,768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32,768 \uparrow \times 32$ $+ \uparrow (32 \times \text{differential backup count}) \div 1,024 \uparrow$
Utility special unit	$100 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100 + 1,024$

a: Length of the backup file name specified in the *-b* option (bytes). If multiple backup files are specified, this value is the total length of the file names.

b: Number of backup files specified in the *-b* option

c: If the write buffer size is specified, this value is MAX(64, write buffer size). If not, this value is 60.
The write buffer size is the value specified by the *-Y* option.

d:

- 32-bit mode

$$640 + 8 \times \uparrow \text{maximum number of concurrently executed transactions} \div 100 \uparrow$$

$$+ 5 \times \uparrow \text{number of RDAREAs subject to recovery} \div 100 \uparrow$$

$$+ \text{maximum page size of RDAREAs subject to recovery} \times 54$$

$$+ 9 \times \text{number of transactions subject to rollback}$$

$$+ 0.02 \times \text{number of RDAREA component files subject to recovery}$$

$$+ \uparrow (304 + 36 + 4 \times (\text{number of RDAREAs subject to recovery} - 1))$$

$$+ 352 + 304 \times (\text{number of RDAREAs subject to recovery} - 1)$$

$$+ 96 + 4 \times (\text{number of RDAREAs subject to recovery} - 1)$$

$$+ 384 + 320 \times (\text{number of RDAREAs subject to recovery} - 1) + 16) \div 1,024 \uparrow$$
- 64-bit mode

$$640 + 11 \times \uparrow \text{maximum number of concurrently executed transactions} \div 100 \uparrow$$

$$+ 6 \times \uparrow \text{number of RDAREAs subject to recovery} \div 100 \uparrow$$

$$+ \text{maximum page size of RDAREAs subject to recovery} \times 54$$

$$+ 9 \times \text{number of transactions subject to rollback}$$

$$+ 0.03 \times \text{number of RDAREA component files subject to recovery}$$

$$+ \uparrow (304 + 40 + 8 \times (\text{number of RDAREAs subject to recovery} - 1))$$

$$+ 400 + 336 \times (\text{number of RDAREAs subject to recovery} - 1)$$

$$+ 168 + 8 \times (\text{number of RDAREAs subject to recovery} - 1)$$

$$+ 408 + 336 \times (\text{number of RDAREAs subject to recovery} - 1) + 16) \div 1,024 \uparrow$$

(2) HiRDB parallel server configuration

Condition	Formula for determining memory size (KB)
MGR	$65 + \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 50$ Add the following value if differential backup files are used for recovery: $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32,768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32,768 \uparrow \times 32$ $+ \uparrow (32 \times \text{differential backup count}) \div 1,024 \uparrow$
DS	$35 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) + 100$ $+ 98 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + c$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{number of RDAREAs subject to recovery} + 99) \div 100\} \times 5$ If you are inputting the unload log file or system log file, add: $+ 57 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + 64$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100 + d$ $+ 0.6 \times \text{number of RDAREAs subject to recovery} + \text{size of work buffer for sorting (value of -y option)}$
BES	$98 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + c$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{number of RDAREAs subject to recovery} + 99) \div 100\} \times 5$ If you are inputting the unload log file or system log file, add: $+ 57 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + 64$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100 + d$ $+ 0.6 \times \text{number of RDAREAs subject to recovery} + \text{size of work buffer for sorting (value of -y option)}$
Server machine containing backup files	$100 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) \times \text{number of servers subject to recovery}$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100 + 1,024$

a: Length of the backup file name specified in the -b option (bytes). If multiple backup files are specified, this value is the total length of the file names.

b: Number of backup files specified in the -b option

c: If the write buffer size is specified, this value is MAX(64, write buffer size). If not, this value is 60.

The write buffer size is the value specified by the -Y option.

d:

- 32-bit mode

$$\begin{aligned}
 &640 + 8 \times \uparrow \text{maximum number of concurrently executed transactions} \div 100 \uparrow \\
 &+ 5 \times \uparrow \text{number of RDAREAs subject to recovery} \div 100 \uparrow \\
 &+ \text{maximum page size of RDAREAs subject to recovery} \times 54 \\
 &+ 9 \times \text{number of transactions subject to rollback} \\
 &+ 0.02 \times \text{number of RDAREA component files subject to recovery} \\
 &+ \uparrow (304 + 36 + 4 \times (\text{number of RDAREAs subject to recovery} - 1)) \\
 &+ 352 + 304 \times (\text{number of RDAREAs subject to recovery} - 1) \\
 &+ 96 + 4 \times (\text{number of RDAREAs subject to recovery} - 1)
 \end{aligned}$$

- + 384 + 320 × (number of RDAREAs subject to recovery - 1)
- + 16 + 32 × (number of RDAREAs subject to recovery) ÷ 1,024 ↑
- 64-bit mode
 - 640 + 11 × ↑ maximum number of concurrently executed transactions ÷ 100 ↑
 - + 6 × ↑ number of RDAREAs subject to recovery ÷ 100 ↑
 - + maximum page size of RDAREAs subject to recovery × 54
 - + 9 × number of transactions subject to rollback
 - + 0.03 × number of RDAREA component files subject to recovery
 - + ↑ (304 + 40 + 8 × (number of RDAREAs subject to recovery - 1))
 - + 400 + 336 × (number of RDAREAs subject to recovery - 1)
 - + 168 + 8 × (number of RDAREAs subject to recovery - 1)
 - + 408 + 336 × (number of RDAREAs subject to recovery - 1)
 - + 16 + 48 × (number of RDAREAs subject to recovery) ÷ 1,024 ↑

19.2.11 Memory size required for the execution of the dictionary import/export utility (pdexp)

The following tables show the formulas for determining the memory size required for the execution of the dictionary import/export utility (pdexp).

(1) HiRDB single server configuration

Conditions		Memory required by formula (KB)
32-bit mode	SDS	$1,307 + 6.7 \times \uparrow CTL \div 100 \uparrow + 0.07 \times CTL$
	Host with export file	$4,499 + 0.07 \times CTL + 0.11 \times CHK + 1.5 \times FKY + CSZ + 0.5 \times CMN + 0.01 \times (CHK + FKY + CMN + DIV + 10) + 0.6 \times DIV + DEF + 0.06 \times LOB + 0.2 \times TBL + SQL + 8,791$
64-bit mode	SDS	$1,494 + 6.7 \times \uparrow CTL \div 100 \uparrow + 0.07 \times CTL$
	Host with export file	$4,582 + 0.07 \times CTL + 0.11 \times CHK + 1.5 \times FKY + CSZ + 0.5 \times CMN + 0.01 \times (CHK + FKY + CMN + DIV + 10) + 0.6 \times DIV + DEF + 0.06 \times LOB + 0.2 \times TBL + SQL + 8,791$

(2) HiRDB parallel server configuration

Conditions		Memory required by formula (KB)
32-bit mode	MGR	$1,714 + 6.7 \times \uparrow CTL \div 100 \uparrow + 0.07 \times CTL$
	Host with export file	$4,907 + 0.07 \times CTL + 0.11 \times CHK + 1.5 \times FKY + CSZ + 0.5 \times CMN + 0.01 \times (CHK + FKY + CMN + DIV + 10) + 0.6 \times DIV + DEF + 0.06 \times LOB + 0.2 \times TBL + SQL + 8,791$
64-bit mode	MGR	$2,016 + 6.7 \times \uparrow CTL \div 100 \uparrow + 0.07 \times CTL$
	Host with export file	$5,293 + 0.07 \times CTL + 0.11 \times CHK + 1.5 \times FKY + CSZ + 0.5 \times CMN + 0.01 \times (CHK + FKY + CMN + DIV + 10) + 0.6 \times DIV + DEF + 0.06 \times LOB + 0.2 \times TBL + SQL + 8,791$

(3) Variables used in the formula

CMN: Number of columns

CHK: Number of check constraints

CSZ: Total size of search conditions of check constraint (CHK_SOURCE_LEN value in SQL_TABLES table) (bytes)

CTL: Number of controls specified in control statement file

DEF: Maximum definition length of DEFAULT clause (bytes)

DIV: Number of partition conditions

FKY: Number of foreign keys

LOB: Number of LOB storage RDAREAs

SQL: Memory required to use the following SQL statements:

- For table export/import: CREATE TABLE statement
- For procedure export/import: CREATE PROCEDURE statement
- For trigger import/export: CREATE TRIGGER statement

For details about these memory requirements, see *15.1.6 Formulas for size of memory required during SQL execution* and *15.1.7 Formula for size of memory required during SQL preprocessing*.

TBL: Number of tables, procedures and triggers actually imported or exported (for procedures and triggers, the same as CTL)

19.2.12 Memory size required for the execution of the access path display utility (pdvwopt)

The following table shows the formulas for determining the memory size required for the execution of the access path display utility (pdvwopt).

Condition		Formula for determining memory size (KB)
HiRDB single server configuration		a
HiRDB parallel server configuration	FES	$\sum_{i=1} b_i \times 0.7 + 200$

a : Number of queries in SQL

b_i : Number of tables in query

19.2.13 Memory size required for the execution of the rebalancing utility (pdrbal)

The following table shows the formulas for determining the memory size required for the execution of the rebalancing utility (pdrbal).

(1) For a HiRDB single server configuration

Formula for determining memory size (KB)
$8,756^{#3} + 536 + 0.02 \times \text{number of columns} + 0.2 \times \text{number of target RDAREAs}$ $+ 1.7 \times \text{number of source RDAREAs} + 0.26 \times \text{number of target RDAREAs} \times \text{number of indexes}$ $+ (0.09 + \uparrow \text{average index statement file length} \div \uparrow 1,024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div \uparrow 1,024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ \uparrow (\text{length of control information file} + \text{length of execution results file}) \div 1,024 \uparrow$ $+ 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ $+ 550 \times 1,024 + \text{size of work file for sorting}^{#1}$

Formula for determining memory size (KB)

-n option specified:

+ page length of RDAREA^{#2} × number of batch input/output buffer sectors × y

Applicable table containing LOB columns:

+ 64 + 0.01 × number of LOB columns + 0.18 × number of target RDAREAs

+ 0.09 × number of source RDAREAs + 0.08 × number of LOB storage RDAREAs

When the target table contains BINARY columns:

+ 33 × (number of BINARY columns × number of target RDAREAs × number of source RDAREAs)

+ buffer length used for BINARY columns^{#5}

Applicable table containing an abstract data type provided by plug-in:

+ 40 + (0.27 + 2 × length of abstract data type) × number of abstract data-type columns

+ 0.3 × number of unld_func statements

+ (128 + 0.11 × number of LOB attributes + 0.1 × number of functions specifying unld_func

+ 0.07 × number of abstract data type attributes) × number of abstract data-type columns

+ (33 × number of BINARY attributes × number of target RDAREAs × number of source RDAREAs) × 2

+ 0.01 × number of plug-in indexes + 0.19 × number of unld_func statements

+ (↑ average length of unld_func statements ÷ 1,024 ↑ × number of unld_func statements)

+ (↑ average length of reld_func statements ÷ 1,024 ↑ × number of reld_func statements)

+ number of abstract data-type columns × 1 + (number of LOB attributes × 0.05) × number of RDAREAs

+ number of data type plug-ins × 10 + number of plug-in indexes × 10

+ memory required for plug-ins

When the specified index creation method is the batch index creation mode or the index information output mode and the following condition is satisfied:

Number of table partitions × number of index definitions > maximum number of processes that can be open - 576 + 2,048

When there are compressed columns:

+ split compression size^{#4} × z + RDAREA's page length^{#2}

y: Use one of the following values:

- When the rebalancing facility is used with a FIX hash-partitioned table
(↑ 1,024 ÷ number of storage RDAREAs for the entire table ↑) × number of table storage RDAREAs in the corresponding server
- Other than the above

1

z: Use one of the following values, as applicable:

- Exclusive mode (-k exclusive): 2
- Shared mode (-k share): 1

#1: Add this value during batch index creation (-ic specified or omitted).

#2: If the page length varies from one RDAREA to another for a row-partitioned table, use the longest page length for this calculation.

#3: For the 64-bit mode, the value is 9,764.

#4: Use the largest value among all compressed columns for the split compression size.

#5: Use one of the following values, as applicable, for the buffer length used for BINARY columns:

- Exclusive mode
0
- Shared mode

n

Σ (definition length i)

$i = 1$
 n : Number of BINARY columns

(2) For a HiRDB parallel server configuration

Condition	Formula for determining memory size (KB)
MGR	$1,498^{#3} + 2 + 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ $+ (0.09 + \uparrow \text{average index statement file length} \div 1,024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div 1,024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ \uparrow (\text{length of control information file} + \text{length of execution results file}) \div 1,024 \uparrow$ Applicable table containing LOB columns: $+ 0.08 \times \text{number of LOB storage RDAREAs}$ Applicable table containing an abstract data type provided by plug-in: $+ 0.19 \times \text{number of unld_func statements}$ $+ (\uparrow \text{average length of unld_func statement} \div 1,024 \uparrow \times \text{number of unld_func statements})$ $+ (\uparrow \text{average length of reld_func statement} \div 1,024 \uparrow \times \text{number of reld_func statements})$
DS	$1,455^{#4} + 32 + 0.33 \times \text{number of target BESs} + 0.3 \times \text{number of source BESs}$ $+ 0.2 \times \text{number of target RDAREAs} + 0.22 \times \text{number of source RDAREAs} + 0.34 \times \text{number of FESs}$ $+ (0.09 + \uparrow \text{average index statement file length} \div 1,024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div 1,024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ Applicable table containing LOB columns: $+ 0.08 \times \text{number of LOB storage RDAREAs}$ When the target table contain BINARY columns: $+ 33 \times (\text{number of BINARY columns} \times \text{number of target RDAREAs} \times \text{number of source RDAREAs})$ Applicable table containing an abstract data type provided by plug-in: $+ 0.01 \times \text{number of target BESs} + 0.19 \times \text{number of unld_func statements}$ $+ (\uparrow \text{average length of unld_func statements} \div \uparrow 1,024 \uparrow \times \text{number of unld_func statements})$ $+ (\uparrow \text{average length of reld_func statements} \div \uparrow 1,024 \uparrow \times \text{number of reld_func statements})$ $+ \text{number of abstract data type columns} \times 1 + (\text{number of LOB attributes} \times 0.05) \times \text{number of RDAREAs}$ $+ \text{number of data type plug-ins} \times 10 + \text{number of plug-in indexes} \times 10$
BES	$6,601^{#5} + 50 + (517 + 0.01 \times \text{number of columns}) \times \text{number of target BESs}$ $+ (33 + 0.01 \times \text{number of columns}) \times \text{number of source BESs} + 0.2 \times \text{number of target RDAREAs}$ $+ 1.7 \times \text{number of source RDAREAs} + 0.01 \times \text{number of columns}$ $+ 0.26 \times \text{number of target RDAREAs} \times \text{number of indexes}$ $+ (0.09 + \uparrow \text{average index statement file length} \div 1,024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div 1,024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ $+ 550 \times 1,024 + \text{size of work file for sorting}^{#1}$ -n option specified: $+ \text{page length of RDAREA}^{#2} \times \text{number of batch input/output buffer sectors} \times y$ Applicable table containing LOB columns:

Condition	Formula for determining memory size (KB)
	$+ 32 + 0.01 \times \text{number of LOB columns} + (32 + 0.01 \times \text{number of LOB columns}) \times \text{number of target BESs}$ $+ 0.18 \times \text{number of target RDAREAs} + 0.1 \times \text{number of source RDAREAs}$ $+ 0.08 \times \text{number of LOB storage RDAREAs}$ When the target table contain BINARY columns: $+ 33 \times (\text{number of BINARY columns} \times \text{number of target RDAREAs} \times \text{number of source RDAREAs})$ $+ \text{buffer length used for BINARY columns}^{\#7}$ When the target table contains abstract data types provided by plug-ins: $+ 40 + (0.27 + 2 \times \text{length of abstract data type}) \times \text{number of abstract data-type columns}$ $+ 0.3 \times \text{number of unld_func statements}$ $+ \{(64 + 0.05 \times \text{number of LOB attributes}) \times \text{number of abstract data-type columns}\} \times \text{number of target BESs}$ $+ \{(64 + 0.01 \times \text{number of functions specifying unld_func} + 0.07 \times \text{number of abstract data type attributes}$ $+ 0.05 \times \text{number of LOB attributes}) \times \text{number of abstract data-type columns}\} \times \text{number of source BESs}$ $+ (33 \times \text{number of BINARY attributes} \times \text{number of target RDAREAs} \times \text{number of source RDAREAs}) \times 2$ $+ 0.01 \times \text{number of plug-in indexes} + 0.19 \times \text{number of unld_func statements}$ $+ (\uparrow \text{average length of unld_func statements} \div 1,024 \uparrow \times \text{number of unld_func statements})$ $+ (\uparrow \text{average length of reld_func statements} \div 1,024 \uparrow \times \text{number of reld_func statements})$ $+ \text{number of abstract data-type columns} \times 1 + (\text{number of LOB attributes} \times 0.05) \times \text{number of RDAREAs}$ $+ \text{number of data type plug-ins} \times 10 + \text{number of plug-in indexes} \times 10$ $+ \text{memory required for plug-ins}$ When the specified index creation method is the batch index creation mode or the index information output mode and the following condition is satisfied: $\text{Number of table partitions} \times \text{number of index definitions} > \text{maximum number of processes that can be open} - 576 + 2,048$ When there are compressed columns: $+ \text{split compression size}^{\#6} \times z + \text{RDAREA's page length}^{\#2}$

y: Use one of the following values:

- When the rebalancing facility is used for FIX hash partitioning tables
 $(\uparrow 1,024 \div \text{number of storage RDAREAs for the entire table} \uparrow) \times \text{number of table storage RDAREAs in the corresponding server}$
- Other than the above

z: Use one of the following values, as applicable:

- Exclusive mode (`-k exclusive`): 2
- Shared mode (`-k share`): 1

#1: Add this value during batch index creation (`-ic` specified or omitted).

#2: If the page length varies from one RDAREA to another for a row-partitioned table, use the longest page length for this calculation.

#3: For the 64-bit mode, the value is 1,790.

#4: For the 64-bit mode, the value is 1,671.

#5: For the 64-bit mode, the value is 6,908.

#6: Use the largest value among all compressed columns for the split compression size.

#7: Use one of the following values, as applicable, for the buffer length used for BINARY columns:

- Exclusive mode

m

$(\sum (\text{definition length } i))$

$i = 1$

$$+ \sum_{i=1}^n (\text{definition length } i \times 9) \times \text{number of target BESs}$$

m : Number of `BINARY` columns for which the compression specification is not specified

n : Number of `BINARY` columns for which the compression specification is specified

- Shared mode

$$\sum_{i=1}^n (\text{definition length } i) \times \text{number of target BESs}$$

n : Number of `BINARY` columns

19.2.14 Memory size required for execution of the free page release utility (pdreclaim) and global buffer residence utility (pdpgbfon)

Use the following formulas to determine the memory size required for execution of the free page release utility (pdreclaim) and global buffer residence utility (pdpgbfon):

Condition		Formula for the memory requirement (KB)
HiRDB single server configuration (32-bit mode)		$800 + W$
HiRDB single server configuration (64-bit mode)		$800 + W$
HiRDB parallel server configuration (32-bit mode)	MGR	$800 + X$
	Servers specified with the <code>-s</code> option ^{#1}	Y
	BES ^{#2}	Z
HiRDB parallel server configuration (64-bit mode)	MGR	$850 + X$
	Servers specified with the <code>-s</code> option ^{#1}	Y
	BES ^{#2}	Z

#1: If the `-s` option is omitted, the table storage RDAREA used for processing is at the first defined server.

#2: If there are multiple back-end servers, add this memory size for each back-end server.

W : Memory required for a single server when the database reorganization utility (pdorg) is executing.

X : Memory required for a MGR when the database reorganization utility (pdorg) is executing.

Y : Memory required for a DS when the database reorganization utility (pdorg) is executing.

Z : Memory required for a BES when the database reorganization utility (pdorg) is executing.

For details about the size of the memory required when the database reorganization utility (pdorg) is executing, see *19.2.4 Memory size required for the execution of the database reorganization utility (pdorg)*.

19.2.15 Memory size required for execution of the integrity check utility (pdconstck)

Use the following formulas to determine the size of the memory required to execute the integrity check utility (pdconstck).

Condition			Formula for the memory requirement (KB)
HiRDB single server configuration	32-bit mode		$13,995 + \alpha$
	64-bit mode		$14,766 + \alpha$
HiRDB parallel server configuration	32-bit mode	MGR	$6,675 + \alpha$
		DS	$5,910 + \beta$
	64-bit mode	MGR	$7,252 + \alpha$
		DS	$8,740 + \beta$

α : The value obtained by the following formula:

$$\begin{aligned}
& 2,175 \\
& + 0.14 \times \text{number of columns} \\
& + 0.09 \times \text{number of table storage RDAREAs} \\
& + 0.23 \times \text{number of indexes} \\
& + 0.09 \times \text{number of index storage RDAREAs} \\
& + 0.09 \times \text{number of LOB storage RDAREAs} \\
& \text{Number of foreign keys} \\
& + \sum_{r=1} (42 + 0.47 \times \text{number of foreign key component columns } r) \\
& \text{Number of check constraints} \\
& + \sum_{c=1} (5 + 0.29 \times \text{number of columns in the search condition } c + 0.85 \\
& \times \text{number of ANDs and ORs in the search condition } c + \text{length of the search condition } c)
\end{aligned}$$

β : The value obtained from the following formula

$$\begin{aligned}
& 0.2 \\
& + 0.02 \times \text{number of table storage RDAREAs} \\
& + 0.02 \times \text{number of index storage RDAREAs} \\
& + 0.02 \times \text{number of LOB column storage RDAREAs}
\end{aligned}$$

19.2.16 Memory size required for the execution of parallel loading (pdparaload)

The following table shows the formula for finding the amount of memory required to execute parallel loading (pdparaload).

Amount of memory required to execute parallel loading (KB)
$= 1,000 + \text{Size of pdparaload control statement file} + 10 + 10 + R$
R
$+ \sum_{i=1} (\text{Amount of memory required to execute data loads by RDAREA})$

R : Number of RDAREAs that constitute the table

Note

The `pdparaload` command executes as many data loads in RDAREA units (`pdload`) internally as there are RDAREAs that constitute the table. For this reason, `pdparaload` uses as many files required for execution of data loads in RDAREA units as there are RDAREAs that constitute the table. For details about the size of memory

required for execution of data loads by RDAREA, see *19.2.3 Memory size required for the execution of the database load utility (pload)*.

20

Specifying OS Parameters

This chapter describes the procedures for estimating the OS parameter values (or kernel parameter values).

20.1 Estimating HP-UX OS parameter values

This section describes the procedures for estimating the HP-UX OS parameter values (or kernel parameter values). If OS parameter values are too small, HiRDB may not function correctly. The table below shows guidelines for HP-UX OS parameter values.

If you are using HP-UX 11i, do not change the OS parameter values while HiRDB is running. Even an OS parameter that can be adjusted automatically, if changed may affect HiRDB operations.

Table 20–1: Guidelines for the HP-UX OS parameter values

OS parameter	Guideline for value
maxdsiz (32-bit mode) maxdsiz_64bit (64-bit mode)	<p>■ HiRDB single server configuration</p> <p>The target value should satisfy the following two conditions; however, if the value is less than (process private area used by the single server process^{#1}) $\div g$, specify a value greater than this:</p> <p>32-bit mode</p> <ul style="list-style-type: none"> • <i>Memory size for restarting HiRDB^{#2} + 52,428,800 (bytes) or greater</i> • <i>Value of <code>pd_work_buff_size</code> \times 1,024 + 134,217,728 (bytes) or greater</i> <p>64-bit mode</p> <ul style="list-style-type: none"> • <i>Memory size for restarting HiRDB^{#2} + 52,428,800 (bytes) or greater</i> • <i>Value of <code>pd_work_buff_size</code> \times 1,024 + 134,217,728 (bytes) or greater</i> <p>■ HiRDB parallel server configuration</p> <p>Use the following value as a guideline; however, if the value is less than (process private area used by each server process^{#3}) $\div g$, specify a value greater than this:</p> <ul style="list-style-type: none"> • <i>Size of memory required to restart HiRDB on each server^{#2} + 134,217,728 (bytes) or greater</i>
maxssiz maxssiz_64bit (64-bit mode)	Specify 80 MB or greater. Note that the unit for this value is MB. If this value is less than the value required by another program that is run on the server machine, specify the latter, which is greater.
maxfiles	HiRDB calculates and sets this value, so you do not need to specify it.
maxfiles_lim	Specify 8192.
nfile ^{#6}	<p>Specify a value at or above: $\text{MAX}\{1,600, 320 \times (h - g - i) + [a + (b \times c) + 320] \times g + 848 \times i + h \times 2 + 227 + k \times m + C\}$</p> <p>Specify at least the value obtained with the above formula by adding one of the following:</p> <p>■ HiRDB single server configuration</p> <p><i>value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code></i></p> <p>■ HiRDB parallel server configuration</p> <p>$(D + 3 \times \text{number of servers in unit}) \times E$</p> <p><i>D: Total the values found by executing the following formulas for each server in the unit.</i></p> <p>Back-end server</p> <p><i>value of <code>pd_max_bes_process</code> + value of <code>pd_max_reflect_process_count</code></i></p> <p>Dictionary server</p> <p><i>value of <code>pd_max_dic_process</code> + value of <code>pd_max_reflect_process_count</code></i></p> <p>If the <code>pd_max_bes_process</code> or <code>pd_max_dic_process</code> operand is omitted, perform the calculation using the <code>pd_max_users</code> value.</p> <p>If the unit has multiple back-end servers, perform the calculation for each back-end server.</p> <p><i>E: 16 (number of lock-release wait threads)</i></p> <p>If the value determined from this formula exceeds the system maximum, use the system maximum value.</p>
nflocks	Specify at least $a + (b \times c) + 3 + (320 \times g)$.

OS parameter	Guideline for value
maxuprc	Specify at least $\text{MAX}(\text{value of } \text{pd_max_server_process} + e, 512)$. However, if this value is less than the value required by another program that runs on the server machine, specify the higher value.
nproc	Specify at least the value of $\text{MAX}(\text{pd_max_server_process} + 20, 576)$.
msgmni	Specify the number of message queue identifiers required by all programs that are run on the server machine. For details about the number of message queue identifiers required by HiRDB, see <i>20.5 Estimating the sizes of message queues and semaphores</i> . Add the values obtained.
msgtql	See <i>msgtql formula</i> below. Specify at least the value obtained.
msgmnb	See <i>msgmnb formula</i> below. Specify at least the value obtained. However, for a multi-HiRDB configuration, specify the highest of the values obtained for the HiRDB systems.
semnmi	Specify the number of semaphore identifiers required by all programs that are run on the server machine. For details about the number of semaphore identifiers required by HiRDB, see <i>20.5 Estimating the sizes of message queues and semaphores</i> . Add the values obtained. The recommended value is 1,024 or greater.
semnms	Specify the number of semaphores required by all programs that are run on the server machine. For details about the number of semaphores required by HiRDB, see <i>20.5 Estimating the sizes of message queues and semaphores</i> . Add the values obtained. The recommended value is 7,200 or greater.
semnmu	Specify a value of 512 or greater.
semume	Specify a value of 512 or greater.
shmmax	Specify at least $\text{MAX}(p + q, r, s, t)$, and at least 200,000,000. If the global buffer dynamic update facility is used, consider the size of the global buffers to be added; if there is a possibility that the size of the added part will become greater than the specified value, specify the anticipated size of the added part. However, if the inter-process memory communication facility is used (PDIPC=MEMORY is specified in the client environment definition), specify a size of at least $\text{MAX}(p + q, r, s, \text{value of } \text{PSENDMEMSIZE}, \text{value of } \text{PDRECVMEMSIZE})$. Specify in the HiRDB system definition's SHMMAX operand a value no greater than the value for shmmax determined here.
shmmni	Specify a value of 1,000 or greater. <ul style="list-style-type: none"> If you are using the security audit facility, add 1. If you are using the facility for dynamically changing global buffers, add: <p>HiRDB single server configuration value of pd_max_add_dbbuff_shm_no operand</p> <p>HiRDB parallel server configuration $n \sum_{i=1} \text{the values of } \text{pd_max_add_dbbuff_shm_no} \text{ operands specified in each server definition}$ $n: \text{Number of back-end servers} + \text{dictionary servers on the server machine}$ </p> If you are using the inter-process memory communication facility (if PDIPC=MEMORY has been specified in the client environment definitions), add the value obtained with the following formula: <p>$A \times 2 \times 1.2$ A is the maximum number of concurrent executions of clients that use the inter-process memory communication facility. If A is unknown, substitute the total number of clients that use the inter-process memory communication facility or k.</p> If you are using in-memory data processing, add: <p>HiRDB single server configuration value of pd_max_resident_rdarea_shm_no operand</p> <p>HiRDB parallel server configuration value of pd_max_resident_rdarea_shm_no operand \times number of back-end servers</p>
shmseg	Specify a value of 120 or greater.

OS parameter	Guideline for value
	<ul style="list-style-type: none"> • If you are using the security audit facility, add 1. • If you are dynamically changing global buffers, add the maximum value of the <code>pd_max_add_dbbuff_shm_no</code> operand specified by the individual server definitions. • For in-memory data processing, add the value of the <code>pd_max_resident_rdarea_shm_no</code> operand.

a: Number of input data files and divided-input data files used by the database load utility, or the number of unload data files used by the database reorganization utility

b: Maximum number of index row partitions (indexes subject to processing by the database load utility, database reorganization utility, or rebalancing utility)

c: Number of indexes (indexes subject to processing by the database load utility, database reorganization utility, or rebalancing utility)

d: One of the following values:

- HiRDB single server configuration
Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand
- HiRDB parallel server configuration
Value of `pd_max_bes_process` operand + value of `pd_max_reflect_process_count` operand

e: Maximum number of concurrently executable commands (including utilities)

f: One of the following:

- HiRDB single server configuration: 1
- HiRDB parallel server configuration: 16

g: One of the following values:

- HiRDB single server configuration: Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand
- HiRDB parallel server configuration: Total of the following values for all back-end and dictionary servers in the unit:
Value of `pd_max_bes_process` operand + value of `pd_max_reflect_process_count` operand
Value of `pd_max_dic_process` operand + value of `pd_max_reflect_process_count` operand

h: Value of `pd_max_server_process` operand

i: Number of servers in the unit

j: One of the following:

- HiRDB single server configuration: 4
- HiRDB parallel server configuration: 35

k: Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand

m: One of the following values:

- If a front-end server exists in the unit in a HiRDB parallel server configuration: 2
- Other than the above: 0

n: If you specify the batch index creation mode or index information output mode as the index creation mode with the database load utility, database reorganization utility, or rebalancing utility, use the value obtained from the following formula:

$\text{MIN}(\text{MAX}(576, \text{number of HiRDB servers in the system} + 448) + b \times c, \text{maximum value of the } \text{pd_max_open_fds} \text{ operand})$

For details about the maximum value of the `pd_max_open_fds` operand, see the manual *HiRDB Version 9 System Definition*.

p: Size of the shared memory used by the unit controller

q : Size of the shared memory used by the single server or each server

r : Value specified in the HiRDB system definition's SHMMAX operand

s : Estimated value for shared memory used by the global buffers^{#4}

t : Estimated value for shared memory used by the security audit information buffer^{#5}

C :

For a HiRDB single server configuration: $\text{MAX}(256, (\text{number of HiRDB servers in the system} + 32)) \times (g + k) + (h - k) \times \text{MAX}(\text{number of HiRDB servers in the system}, \text{number of units})$

For a HiRDB parallel server configuration: $\text{MAX}(256, (\text{number of HiRDB servers in the system} + 32)) \times (g + k) + (h - g - k) \times \text{MAX}(\text{number of HiRDB servers in the system}, \text{number of units})$

#1: For details about the process private area used by a single server process, see *15.1.2 Calculation of required memory*.

#2: For details about estimating the memory size for restarting HiRDB, see *15.1.2 Calculation of required memory* for a HiRDB single server configuration, and *15.2.2 Calculation of required memory* for a HiRDB parallel server configuration.

#3: For details about the process private area used by each server process, see *15.2.2 Calculation of required memory*.

#4: For details about estimating the shared memory used by the global buffers, see *15.1.5 Formula for size of shared memory used by global buffers* for a HiRDB single server configuration, and see *15.2.5 Formula for size of shared memory used by global buffers* for a HiRDB parallel server configuration.

#5: For details about estimating the size of shared memory used by the security audit information, for a HiRDB single server configuration, see *15.1.2 Calculation of required memory* for a HiRDB parallel server configuration and *15.2.2 Calculation of required memory* for a HiRDB parallel server configuration.

#6: This value does not need to be specified in HP-UX 11i V3 or later.

• msgtql formula

The following table shows the formula for msgtql.

$$\begin{aligned} \text{msgtql} = & \sum_{i=1}^n (A_i + B_i) + \text{value of } \text{pd_max_users} \text{ operand} \times p \\ & + 1 \\ & + \downarrow \text{value of } \text{pd_trn_rcvmsg_store_buflen} \text{ operand} \div 72 \downarrow \end{aligned}$$

$$A_i = \sum_{j=1}^m (C_j + D_j \times 2 + E_j + 1)$$

$$B_i = \sum_{j=1}^m F_j$$

Add B_i if you are using the asynchronous READ facility (if 1 or more is specified for the `pd_max_ard_process` operand). If you are not using the asynchronous READ facility, count this as 0.

C_i : The number of global buffers allocated to each server

Total the global buffer names displayed by the `pdbufls` command in server units to confirm them.

D_i : Skips of effective synchronization point dumps generated by each server

- If a nonzero value is specified in the `pd_spd_syncpoint_skip_limit` operand: Use the value specified by the `pd_spd_syncpoint_skip_limit` operand in the formula.
- If the `pd_spd_syncpoint_skip_limit` operand is not specified or is 0: Calculate as directed in *Method based on the byte count of all system logs in Monitoring UAP status (skipped effective synchronization point dump monitoring facility)* in the *HiRDB Version 9 System Operation Guide*.

E_i : Maximum number of concurrent executions of `pdload`, `pdrorg`, `pdrbal`, and no-log mode UAPs executed in each server

F_i : Total of values specified in the `-m` options of `pdbuffer` operands among the global buffers allocated to individual servers

m: For a HiRDB single server configuration, 1. For a HiRDB parallel server configuration, the following value.

- Number of back-end servers in unit + number of dictionary servers in unit + number of guest BESs in unit

Add the number of guest BESs in the unit if the unit is using the standby-less system switchover (effects distributed) facility.

n: Number of units defined in the server machine

p: For a HiRDB single server configuration, 1. For a HiRDB parallel server configuration, the following value:

- *number of FESs in the unit + number of BESs in the unit + number of dictionary servers in the unit + value of pd_ha_max_act_guest_servers operand*

Add the value of the `pd_ha_max_act_guest_servers` operand for a unit that uses the standby-less system switchover (effects distributed) facility.

- **msgmnb formula**

The formula for `msgmnb` is as follows:

$$\text{Value of msgmnb} = \text{MAX}((A + B \times 2 + C + 1)^{\#2} \times 40, D^{\#2} \times 16, E \times F \times 4, G \times 4)^{\#1}$$

#1

Specify the largest of the values obtained for all servers in the unit.

If the unit is subject to standby-less system switchover (1:1), specify the largest value among the values obtained for all servers and alternate BESs in the unit.

If the unit is subject to standby-less system switchover (effects distributed), specify the largest value among the values obtained for all servers in the unit and all BESs in the HA group.

#2

Front-end server: 0

A: Number of global buffers allocated to the server

To confirm this number, total the global buffer names displayed by the `pdbufls` command per server.

B: Number of skips of effective synchronization point dumps generated by the server

- If a nonzero value is specified in the `pd_spd_syncpoint_skip_limit` operand:
Use the value specified by the `pd_spd_syncpoint_skip_limit` operand in the formula.
- If the `pd_spd_syncpoint_skip_limit` operand is not specified or is 0:
Calculate as directed in *Method based on the byte count of all system logs in Monitoring UAP status (skipped effective synchronization point dump monitoring facility)* in the *HiRDB Version 9 System Operation Guide*.

C: Maximum number of concurrent executions of `pdload`, `pdrorg`, `pdrbal`, and no-log mode UAPs executed on the server

D: If you are using the asynchronous READ facility (if 1 or more is specified for the `pd_max_ard_process` operand), the value is as follows:

- Total of values specified in the `-m` options of `pdbuffer` operands among the global buffers allocated to the server

If you are not using the asynchronous READ facility, use 0.

E: Value of the `pd_max_users` operand

F: For a HiRDB single server configuration, 1. For a HiRDB parallel server configuration, the following value:

- *number of FESs in the unit + number of BESs in the unit + number of dictionary servers in the unit + value of pd_ha_max_act_guest_servers operand*

Add the value of the `pd_ha_max_act_guest_servers` operand for a unit that applies the standby-less system switchover (effects distributed) facility.

G: $\downarrow \text{value of } \text{pd_trn_rcvmsg_store_buflen} \text{ operand} \div 72 \downarrow$

20.2 Estimating Solaris OS parameter values

This section describes the procedures for estimating the Solaris OS parameter values (or kernel parameter values). If OS parameter values are too small, HiRDB may not function correctly. If you are controlling the IPC function using resource control on Solaris 10 or later, set the parameters for the `system` project.

The following table provides guidelines for Solaris OS parameter values.

Table 20–2: Guidelines for the Solaris OS parameter values

OS parameter	Guideline for value
<code>rlim_fd_cur</code>	HiRDB calculates and sets this value, so you do not need to specify it.
<code>rlim_fd_max</code>	For 32-bit mode Specify 2,048. For 64-bit mode Specify 8,192.
<code>maxuprc</code>	Specify at least $\text{MAX}(d + e, 1,024)$. However, if the default values calculated from <code>maxusers</code> and <code>max_nproc</code> are greater, do not change those values.
<code>maxusers</code>	Specify at least 128.
<code>max_nprocs</code> ^{#1}	Specify at least $\text{MAX}(d + 20, 1,000)$. However, if the default value calculated from <code>maxusers</code> is greater, do not change that value.
<code>msgsys:msginfo_msgmni</code> (<code>project.max-msg-ids</code>) ^{#1, #4}	Specify the number of message queue identifiers required by all programs that are run on the server machine. For details about the number of message queue identifiers required by HiRDB, see <i>20.5 Estimating the sizes of message queues and semaphores</i> . Add the values obtained.
<code>msgsys:msginfo_msgtql</code> (<code>process.max-msg-messages</code>) ^{#1, #4}	See <i>msgtql</i> formula in <i>20.1 Estimating HP-UX OS parameter values</i> . Specify at least the value obtained.
<code>semsys:seminfo_semmni</code> (<code>project.max-sem-ids</code>) ^{#4}	Specify the number of semaphore identifiers required by all programs that are run on the server machine. For details about the number of semaphore identifiers required by HiRDB, see <i>20.5 Estimating the sizes of message queues and semaphores</i> . Add the values obtained. The recommended value is 1,024 or greater.
<code>semsys:seminfo_semmns</code> ^{#1, #3}	Specify the number of semaphores required by all programs that are run on the server machine. For details about the number of semaphores required by HiRDB, see <i>20.5 Estimating the sizes of message queues and semaphores</i> . Add the values obtained. The recommended value is 7,200 or greater.
<code>semsys:seminfo_semmnu</code> ^{#1, #3}	Specify a value of 1,024 or greater.
<code>semsys:seminfo_semume</code> ^{#1, #3}	Specify a value of 512 or greater.
<code>semsys:seminfo_semmsl</code> (<code>process.max-sem-nsems</code>) ^{#4}	Specify a value of 128 or greater.
<code>semsys:seminfo_semopm</code> (<code>process.max-sem-ops</code>) ^{#4}	Specify a value of 128 or greater.
<code>semsys:seminfo_semmap</code> ^{#1, #2, #3}	Specify a value of 1,024 or greater.
<code>shmsys:shminfo_shmmax</code> ^{#1, #5}	Specify at least $\text{MAX}(p + q, r, s, t)$, and at least 200,000,000.

OS parameter	Guideline for value
	<p>If the global buffer dynamic update facility is used, consider the size of the global buffers to be added; if there is a possibility that the size of the added part will become greater than the specified value, specify the anticipated size of the added part.</p> <p>However, if the inter-process memory communication facility is used (PDIPC=MEMORY is specified in the client environment definition), specify a size of at least $\text{MAX}(p + q, r, s, \text{value of PDSENDMEMSIZE, value of PDRECVMEMSIZE})$.</p> <p>Specify in the HiRDB system definition's SHMMAX operand a value no greater than the value for shmmax determined here.</p>
project.max-shm-memory ^{#6}	<p>Specify a value greater than the value obtained from adding the HiRDB shared memory size to the size of shared memory used by other applications and OS processes.</p> <p>You can find the size of HiRDB shared memory in the following locations:</p> <ul style="list-style-type: none"> For a HiRDB single server configuration The shared memory of <i>Table 15-2 Size of memory required for a HiRDB single server configuration</i> For a HiRDB parallel server configuration The shared memory of <i>Table 15-7 Size of memory required for each unit of a HiRDB parallel server configuration</i> <p>If you are using the facility for dynamically changing global buffers, consider the size of the global buffer to be added and, if the additional amount might possibly become larger than the set value, specify the predicted size of the additional amount.</p>
shminfo_shmmni (project.max-shm-ids) ^{#1, #4}	<p>Specify a value of 2,000 or greater.</p> <ul style="list-style-type: none"> If you are using the security audit facility, add 1. Also, if you are using the facility for dynamically changing global buffers, add: HiRDB single server configuration value of the pd_max_add_dbbuff_shm_no operand HiRDB parallel server configuration $n \sum_{i=1} \text{values of the pd_max_add_dbbuff_shm_no operands set in each server definition}$ $n: \text{number of back-end servers + dictionary servers on the server machine}$ If you are using the inter-process memory communication facility (if PDIPC=MEMORY was specified in the client environment definitions), add the value obtained with the following formula: $A \times 2 \times 1.2$ <p><i>A</i> is the maximum number of concurrent executions of clients that use the inter-process memory communication facility. If <i>A</i> is unknown, substitute the total number of clients that use the inter-process memory communication facility or <i>k</i>.</p> If you are using in-memory data processing, add: HiRDB single server configuration value of the pd_max_resident_rdarea_shm_no operand HiRDB parallel server configuration value of the pd_max_resident_rdarea_shm_no operand \times number of back-end servers
shminfo_shmseg ^{#2, #3}	<p>Specify a value of 240 or greater.</p> <ul style="list-style-type: none"> If you are using the security audit facility, add 1. If you are dynamically changing global buffers, add the maximum value of the pd_max_add_dbbuff_shm_no operands specified in the individual server definitions. For in-memory data processing, add the value of the pd_max_resident_rdarea_shm_no operand.

d: Value of pd_max_server_process

e: Maximum number of utility processes that are executed concurrently by the HiRDB administrator

k: Value of pd_max_users operand + value of pd_max_reflect_process_count operand

n: If you specify the batch index creation mode or index information output mode as the index creation mode with the database load utility, database reorganization utility, or rebalancing utility, use the value obtained from the following formula:

$\text{MIN}(\text{MAX}(576, \text{number of HiRDB servers in system} + 448) + b \times c, \text{maximum value of the } \text{pd_max_open_fds} \text{ operand})$

For details about the maximum value of the `pd_max_open_fds` operand, see the manual *HiRDB Version 9 System Definition*.

In other cases, use 0.

b: Maximum number of index row partitions (indexes subject to processing by the database load utility, database reorganization utility, or rebalancing utility)

c: Number of indexes (indexes subject to processing by the database load utility, database reorganization utility, or rebalancing utility)

p: Size of the shared memory used by the unit controller

q: Size of the shared memory used by the single server or each server

r: Value specified in the HiRDB system definition's `SHMMAX` operand

s: Estimated value for shared memory used by the global buffers

For details about estimating the shared memory used by the global buffers, see *15.1.5 Formula for size of shared memory used by global buffers* for a HiRDB single server configuration, and see *15.2.5 Formula for size of shared memory used by global buffers* for a HiRDB parallel server configuration.

t: Estimated value for shared memory used by the security audit information buffer

For details about estimating the shared memory used by the security audit information buffer, see *15.1.2 Calculation of required memory* for a HiRDB single server configuration and *15.2.2 Calculation of required memory* for a HiRDB parallel server configuration.

#1: This parameter is not required for Solaris 8.

#2: This parameter is not required for Solaris 9.

#3: This parameter is not required for Solaris 10.

#4: For Solaris 10, use Solaris' resource control to specify the parameter inside the parenthesis () for adjustment.

#5

For Solaris 10, this parameter is not required if you are making adjustments using Solaris' resource control.

#6

For Solaris 10, specify this parameter if you are making adjustments using Solaris' resource control.

20.3 Estimating AIX OS parameter values

This section describes the procedures for estimating the HP-UX OS parameter values (or kernel parameter values). If OS parameter values are too small, HiRDB may not function correctly. The following table provides guidelines for AIX OS parameter values.

Table 20–3: Guidelines for the AIX OS parameter values

OS parameter	Guideline for value
<code>data_hard</code>	The parameter default value is <code>-1 (unlimit)</code> . Do not specify this if there is no particular reason to do so.
<code>stack_hard</code>	Specify <code>-1 (unlimit)</code> .
<code>nofiles</code>	HiRDB calculates and sets this value, so you do not need to specify it.
<code>nofiles_hard</code>	The parameter default value is <code>-1 (unlimit)</code> . Do not specify this if there is no particular reason to do so.
<code>maxuproc</code>	Specify at least $\text{MAX}(\text{value of } \text{pd_max_server_process} + e, 512)$. However, if this value is less than the value required by another program that runs on the server machine, specify the higher value.
EXTSHM environment variable	For 32-bit mode HiRDB, specify <code>ON</code> (do not specify if the number of shared memory segments attached by a single process is less than 11). For 64-bit mode HiRDB, the value to specify depends on the function to be used.

e: Maximum number of concurrently executable commands (including utilities)

Note

- The maximum number of files that can be opened in the system concurrently can be controlled by *value of `maxuproc` × `nofiles`*.
- The maximum number of users who can log in to the system can be controlled by the number of fixed licenses.
- The maximum number of processes that can execute concurrently in the entire system can be controlled by *value of `maxuproc` × `number of fixed licenses`*.

! Important note

TCP ports might go into `TIME_WAIT` status, TCP ports in the system as a whole might run short and cause transactions to generate errors, or HiRDB might terminate abnormally. Should any such problems occur, set the operating system parameters to avoid port shortages. For details, see 22.4.4 *Ways to avoid a shortage of ports*.

(1) Specifying parameters unique to AIX

(a) Specifying environment variables

For AIX, you need to specify the environment variables given below in the system common definition. Note that the environment variables `PSALLOC`, `NODISCLAIM`, and `CORE_NOSHM` do not need to be specified in the system common definition.

Also, note that the environment variables below must also be set in the HiRDB command execution environment. The values of the environment variables set in the system common definition must be the same as the values set in the HiRDB command execution environment. For details about setting environment variables, see the OS documentation.

• EXTSHM

For 32-bit mode HiRDB, specify `ON`, which indicates that there are no restrictions on the number of shared memory areas in process space.

For 64-bit mode HiRDB, if you are using the page fix facility of shared memory, do not specify (omit) `putenv EXTSHM ON` in order to enable the OS's page fix facility. Also, for 64-bit mode HiRDB servers, if you are using the inter-process memory communication facility between 32-bit mode client processes, specify `ON`, indicating that there are no restrictions on the number of shared memory areas in process space.

The following table shows EXTSHM environment variable specification formats for combinations of address modes and facilities used.

Address mode of HiRDB server	Page fix facility of shared memory ^{#1}	Inter-process memory communication facility with 32-bit mode client processes	Specification format for the EXTSHM environment variable
32-bit mode	-- ^{#2}	Y	putenv EXTSHM ON ^{#3}
		N	
64-bit mode	Y	Y	-- ^{#4}
		N	Do not specify
	N	Y	putenv EXTSHM ON
		N	-- ^{#5}

Legend:

Y: Facility is used

N: Facility is not used

--: N/A

#1

The page fix facility of shared memory is used when you specify `fixed` in the `pd_shmpool_attribute` operand or `pd_dbbuff_attribute` operand. For details, see *pd_shmpool_attribute operand* or *pd_dbbuff_attribute operand* in the manual *HiRDB Version 9 System Definition*.

#2

The facility is not supported.

#3

If the size of one segment of shared memory is less than 256 MB, the OS releases shared memory in 4 KB units in shared memory detachment processing that occurs when processes stop. This could cause CPU usage rates to rise and a system switchover to occur. When the number of shared memory segments attached with a single process (the number of shared memory segments that can be checked with the `pdls -d mem` command) is fewer than 11, do not specify `EXTSHM=ON`.

#4

The page fix facility of shared memory cannot be used simultaneously with the inter-process memory communication facility between 32-bit mode client processes.

#5

This parameter is not required.

- PSALLOC

In the HiRDB command execution environment, specify `early`, which indicates that the required paging space is to be allocated immediately. You also need to specify `NODISCLAIM` at the same time. However, useless paging space might still occur.

- NODISCLAIM

In the HiRDB command execution environment, specify `true`, which prevents `nodisclaim()` from being issued as the method of processing a `free()` call.

- LDR_CNTRL

For 32-bit mode, specify this variable so that a larger data area than the standard kernel partitioning can be handled. Specify `MAXDATA=0x40000000`. This variable is not required for 64-bit mode.

- CORE_NOSHM

In the HiRDB command execution environment, specify two consecutive double quotation marks ("") to exclude the shared memory area in the `core` file, which is output when a process failure occurs. For details, see (d) *Restricting the core file output information*.

(b) Notes on the `/etc/security/limits` file specification values

Root users and HiRDB administrators must pay attention to the following specification values:

- `data`
An error occurs when the process heap area exceeds the limit value. If this limit value is not required, specify `-1` (unlimited).
- `fsize, fsize_hard`
An error occurs when the file size exceeds this limit value. If this limit value is not required, specify `-1` (unlimited).

(c) Specifying the Virtual Memory Manager (VMM) tuning parameters

Specify the following parameters when using specific facilities. VMM parameters can be specified with AIX's `vm0` command (AIX 5L V5.2 or later). For details about the `vm0` command, see the AIX documentation.

- `v_pinshm`
This parameter enables page fixing for shared memory segments. To fix shared memory pages used by HiRDB, specify `1` for this parameter. For details about how to fix shared memory pages used by HiRDB, see *pd_shmpool_attribute operand* or *pd_dbbuff_attribute operand* in the manual *HiRDB Version 9 System Definition*.
- `maxpin`
This parameter specifies the maximum percentage of page-fixed real memory. When you fix shared memory pages used by HiRDB, specify this parameter so that the real memory size within the percentage specified in this parameter is larger than the total size (including the size of paged memory fixed by the OS) of all page-fixed memory on the machine, including the size of the page-fixed HiRDB shared memory. For details about how to fix shared memory pages used by HiRDB, see *pd_shmpool_attribute operand* or *pd_dbbuff_attribute operand* in the manual *HiRDB Version 9 System Definition*.

(d) Restricting the core file output information

You must specify the settings in such a manner that in AIX 5L V5.2 or later, the core file output in the event of a process failure does not include the shared memory area.

- Setting contents
HiRDB administrator's environment variable (k shell)

```
$ export CORE_NOSHM=
```
- Assumptions
 - The `fullcore` parameter of the system attribute (`sys0`) is `true`.
- Notes
 - The AIX version determines whether environment variable `CORE_NOSHM` is enabled. See the OS documentation to determine whether you can use this environment variable.
 - HiRDB automatically sets `fullcore` to `true` when it registers with the OS (when the `pdsetup` command executes).
The `fullcore` parameter may have later been reset to `false` with an OS command; make sure that the current `fullcore` parameter is `true`.
 - Do not specify this environment variable `CORE_NOSHM` setting in `/etc/environment`.

(e) Notes on bogging down of programs due to high I/O load on the JFS/JFS2 file system

Execution of programs that send requests to the JFS/JFS2 file system for large output can cause the performance of system disk I/O to degrade. Programs such as `pdcopy`, or the `compress`, `cp`, and `dd` commands for large size files can cause programs running on the same system to stall for up to 20 to 30 seconds.

In particular, if the system operates with a cluster configuration that uses HA monitor or HACMP to monitor system response time, system switchover might occur.

To minimize this problem, you can equalize the write request frequency from application programs by setting an OS parameter for the system parameter (`sys0`). By specifying the following OS parameters, you can control I/O requests so that large numbers of I/O requests not completed for writing to the disk device do not accumulate in the file cache.

OS parameters	Guide to specification value [#]
Maxpout	33
Minpout	16

#

The optimum setting value of the maxpout/minpout parameter depends on the system configuration or I/O characteristics of applications.

Therefore, it is effective to set a value listed in the above table and increase the value until application I/O performance is acceptable.

For details about the maxpout/minpout parameter setting values, see the documentation for the OS.

(f) Notes on correcting the system time

Note the following points when you use the NTP program to correct the system time.

- For the NTP adjustment method, choose Slew mode, which gradually corrects the shift in time so that large corrections of tens of seconds do not occur at once.
- In the AIX environment assignment file (/etc/environment), set GETTOD_ADJ_MONOTONIC=1 so that time will not be turned back.

The AIX version determines whether the environment variable GETTOD_ADJ_MONOTONIC is enabled.

See the OS documentation to determine whether you can use this environment variable.

20.4 Estimating Linux kernel parameter values

This section describes the procedures for estimating the Linux kernel parameter values (or kernel parameter values). If kernel parameter values are too small, HiRDB may not function correctly. The following table provides guidelines for Linux kernel parameter values.

Table 20–4: Guidelines for the Linux kernel parameter values

Kernel parameter	Guideline for value	Example of option settings file [#]
hard nofile	Specify 8,192.	/etc/security/ limits.conf
soft nofile	There is no need to specify this value because it will be set by HiRDB.	--
fs.file- max	$\text{MAX}\{1,600, 320 \times (h - g - i) + [a + (b \times c) + 320] \times g + 848 \times i + h \times 2 + 227 + k \times m + C\}$ <p>Specify at least the value obtained in the above formula by adding one of the following:</p> <ul style="list-style-type: none"> ● HiRDB single server configuration value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code> ● HiRDB parallel server configuration $(D + 3 \times \text{number of servers in the unit}) \times E$ D: Total the values found by executing the following formulas for each server in the unit. Back-end server value of <code>pd_max_bes_process</code> + value of <code>pd_max_reflect_process_count</code> Dictionary server value of <code>pd_max_dic_process</code> + value of <code>pd_max_reflect_process_count</code> If the <code>pd_max_bes_process</code> or <code>pd_max_dic_process</code> operand is omitted, perform this calculation using the <code>pd_max_users</code> value. If the unit has multiple back-end servers, perform this calculation for each back-end server. E: 16 (number of lock-release wait threads) If the value determined from this formula exceeds the system maximum, use the system maximum value. 	/proc/sys/fs/ file-max
nproc	Specify at least (value of <code>pd_max_server_process</code> + <i>e</i> , 512). If the value required by another program running on the server machine is greater than this value, use that value.	/etc/security/ limits.conf
threads- max	Specify at least $\text{MAX}((\text{value of } \text{pd_max_server_process} + 20), 576)$.	/proc/sys/ kernel/threads- max
msgmnb	See 20.1 <i>Estimating HP-UX OS parameter values</i> , under <i>msgmnb formula</i> . Specify at least the value obtained. However, for a multi-HiRDB configuration, specify the largest of the values obtained for the HiRDB systems.	/proc/sys/ kernel/msgmnb
msgmni	Specify the number of message queue identifiers required by all programs that are run on the server machine. For details about the number of message queue identifiers required by HiRDB, see 20.5 <i>Estimating the sizes of message queues and semaphores</i> . Add the values obtained.	/proc/sys/ kernel/msgmni
SEMMNI	Specify the number of semaphore identifiers required by all programs that are run on the server machine. For details about the number of semaphore identifiers required by HiRDB, see 20.5 <i>Estimating the sizes of message queues and semaphores</i> . Add the values obtained. The recommended value is 1,024 or greater.	Parameter 4 in / proc/sys/ kernel/sem
SEMMNS	Specify the number of semaphores required by all programs that are run on the server machine. For details about the number of semaphores required by HiRDB, see 20.5 <i>Estimating the sizes of message queues and semaphores</i> . Add the values obtained. The recommended value is 7,200 or greater.	Parameter 2 in / proc/sys/ kernel/sem

Kernel parameter	Guideline for value	Example of option settings file#
shmmax	Specify at least $\text{MAX}(p + q, r, s, t)$, and at least 200,000,000. If the global buffer dynamic update facility is used, consider the size of the global buffers to be added; if there is a possibility that the size of the added part will become greater than the specified value, specify the anticipated size of the added part. Specify the value of shmmax obtained here or less in the SHMMAX operand of the HiRDB system definition.	/proc/sys/ kernel/shmmax
shmmni	Specify a value of 2,000 or greater. <ul style="list-style-type: none"> If you are using the security audit facility, add 1. If you are using the facility for dynamically changing global buffers, add the following values. If you are using the security audit facility, add 1. HiRDB single server configuration value of the <code>pd_max_add_dbbuff_shm_no</code> operand HiRDB parallel server configuration $\sum_{i=1}^n \text{values of the } pd_max_add_dbbuff_shm_no \text{ operands set in each server definition}$ n : number of back-end servers + dictionary servers on the server machine If you are using in-memory data processing, add: HiRDB single server configuration value of the <code>pd_max_resident_rdarea_shm_no</code> operand HiRDB parallel server configuration value of the <code>pd_max_resident_rdarea_shm_no</code> operand \times number of back-end servers 	/proc/sys/ kernel/shmmni
SHMALL	Specify the value obtained by adding the size of shared memory required for other programs running on the same server machine to <i>si</i> , which is the size of shared memory allocated by HiRDB on the server.	/proc/sys/ kernel/shmall

Legend:

--: Not applicable

#: The file depends on the OS and kernel versions being used. See the applicable OS documentation and set appropriate values, using the values provided in the table as guidelines. With some OS versions, parameter settings may not be necessary. If a kernel parameter is not supported by the OS being used, its setting is not needed.

a: Number of input data files and divided-input data files used by the database load utility, or the number of unload data files used by the database reorganization utility

b: Maximum number of index row partitions (indexes subject to processing by the database load utility, database reorganization utility, or rebalancing utility)

c: Number of indexes (indexes subject to processing by the database load utility, database reorganization utility, or rebalancing utility)

e: Maximum number of concurrently executable commands (including utilities)

g: One of the following values:

- HiRDB single server configuration: Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand
- HiRDB parallel server configuration: Total of the following values for all back-end and dictionary servers in the unit:
Value of `pd_max_bes_process` operand + value of `pd_max_reflect_process_count` operand
Value of `pd_max_dic_process` operand + value of `pd_max_reflect_process_count` operand

h: Value of `pd_max_server_process` operand

i: Number of servers in the unit

k: Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand

m: One of the following values:

- If a front-end server exists in the unit in a HiRDB parallel server configuration: 2
- Other than the above: 0

n: If you specify the batch index creation mode or index information output mode as the index creation mode with the database load utility, database reorganization utility, or rebalancing utility, use the value obtained from the following formula:

$\text{MIN}(\text{MAX}(576, \text{number of HiRDB servers in the system} + 448) + b \times c, \text{maximum value of the } \text{pd_max_open_fds} \text{ operand})$

For details about the maximum value of the `pd_max_open_fds` operand, see the manual *HiRDB Version 9 System Definition*.

p: Size of the shared memory used by the unit controller

q: Size of the shared memory used by the single server or each server

r: Value specified in the HiRDB system definition's `SHMMAX` operand

s: Estimated value for shared memory used by global buffers

For details about estimating the shared memory used by global buffers, see *15.1.5 Formula for size of shared memory used by global buffers* for a HiRDB single server configuration, and see *15.2.5 Formula for size of shared memory used by global buffers* for a HiRDB parallel server configuration.

si: Shared memory allocated by HiRDB at the corresponding server machine

- For a HiRDB single server configuration
Value obtained in *15.1.3 Formulas for shared memory used by a unit controller*
+ value obtained in *15.1.4 Formulas for shared memory used by a single server*
+ value obtained in *15.1.5 Formula for size of shared memory used by global buffers*
- For a HiRDB parallel server configuration
Value obtained in *15.2.3 Formulas for shared memory used by a unit controller*
+ value obtained in *15.2.4 Formulas for shared memory used by each server*
+ value obtained in *15.2.5 Formula for size of shared memory used by global buffers*

t: Estimated value for shared memory used by the security audit information buffer

For details about estimating the size of shared memory used by the security audit information buffer, see *15.1.2 Calculation of required memory* for a HiRDB single server configuration, and see *15.2.2 Calculation of required memory* for a HiRDB parallel server configuration.

C:

For a HiRDB single server configuration: $\text{MAX}(256, (\text{number of HiRDB servers in the system} + 32)) \times (g + k) + (h - k) \times \text{MAX}(\text{number of HiRDB servers in the system}, \text{number of units})$

For a HiRDB parallel server configuration: $\text{MAX}(256, (\text{number of HiRDB servers in the system} + 32)) \times (g + k) + (h - g - k) \times \text{MAX}(\text{number of HiRDB servers in the system}, \text{number of units})$

! Important note

TCP ports might go into `TIME_WAIT` status, TCP ports in the system as a whole might run short and cause transactions to generate errors, or HiRDB might terminate abnormally. Should any such problems occur, set the operating system parameters to avoid port shortages. For details, see *22.4.4 Ways to avoid a shortage of ports*.

(1) Linux-specific specifications

For the Linux version, values specified in settings files such as `xinetd.conf` for the extended Internet service daemon (hereafter, `xinetd`) must be adjusted when the daemon is started. For that reason, specify the following values in the settings file parameters. When a calculated value is smaller than the default for that parameter, you do not need to change the parameter.

For details about `xinetd` and `xinetd.conf`, see the OS documentation.

(a) The `cps` attribute

To set for `shell` service, specify the following values for the first argument (connections processed per second) of the `cps` attribute.

For a HiRDB single server configuration:

If there is no utility special unit: A

If there is a utility special unit: $\text{MAX}(A, B)$

A : number of single servers activated by the machine $\times 4$

B : maximum concurrent executions of `pdcopy` and `pdrstr` $\times 2$

For a HiRDB parallel server configuration: $\text{MAX}(C, (D + E_1 + E_2))$

C : number of dictionary servers started by the machine

\times total number of back-end servers defined in the HiRDB system $\times 6$

+ number of dictionary servers started by the machine

\times total number of back-end servers defined in the HiRDB system $\times 2$

+ number of back-end servers started by the machine $\times 7$

+ number of dictionary servers started by the machine $\times 4$

D : maximum number of concurrent executions of `pdcopy` and `pdrstr`

\times (number of back-end servers defined in the HiRDB system + 4)

E_1 : number of concurrently executable operation commands and utilities $\times 3$

E_2 : When operation involves executing the `pdtrndec` command, the following totals; when it does not, 0

When there is a FES on the machine, the value of `pd_max_users`

When there is a BES on the machine, the value of `pd_max_bes_process`

When there is a DS on the machine, the value of `pd_max_dic_process`

(b) The `instances` attribute

With the settings for `shell` service, specify *value set in (a)* $\times 2$ in the `instances` attribute (the maximum value that the service can concurrently execute).

20.5 Estimating the sizes of message queues and semaphores

This section presents the formulas for determining the sizes of message queues and semaphores required for one server machine.

(1) HiRDB single server configuration

Type	Formula
Number of message queue identifiers	$(16 + f) \times a + 30$
Number of semaphore identifiers	$\{ \uparrow \{ 2 \times (b + 3) + 12 \} \div 64 \uparrow + \uparrow c \div 64 \uparrow + g + 5 \} \times a + 2 + d$
Number of semaphores (Total number of semaphores per identifier)	$\{ 2 \times (b + 3) + c + h + 37 \} \times a + 3 + e$

a: 1 (single server) or 0 (utility special unit)

b: Value of `pd_max_users` operand + value of `pd_max_reflect_process_count` operand.

c: Number of `pdbuffer` operands specified (number of global buffers)

d: Add when the system switchover facility is being used; determine the value from the table below.

e: Add when the system switchover facility is being used; determine the value from the following table:

Condition			Value of d	Value of e
<code>pd_ha_acttype=monitor</code> (or default)			0	0
<code>pd_ha_acttype=server</code>	<code>pd_ha_agent=standbyunit</code>		1	7
	<code>pd_ha_agent</code> omitted	<code>pd_ha_server_process_standby=Y</code> (or default)	1	2
		<code>pd_ha_server_process_standby=N</code>	0	1

f: 1 (if 1 or greater is specified in the `pd_max_ard_process` operand) or 0

g: 2 (if a value is specified in the `pd_dfw_awt_process` operand) or 0

h: Value of the `pd_dfw_awt_process` operand + 1 (if a value is specified in the `pd_dfw_awt_process` operand) or 0

(2) HiRDB parallel server configuration

For details about the variables used in the formulas, see (c) *Variables used in the formulas*.

(a) When the standby-less system switchover (effects distributed) facility is not used

Type	Formula
Number of message queue identifiers	b $\sum_{i=1} V_i + 2 \times a + 3 \times b + c + d + e + 26 + m$
Number of semaphore identifiers	b $\sum_{i=1} \{ \uparrow (S_i + T_i + U_i) \div 64 \uparrow + W_i \} + 6 \times b + 2 + f$

Type	Formula
Number of semaphores (total number of semaphores per identifier)	b $\sum_{i=1} (S_i + T_i + U_i + X_i) + 26 \times b + 3 + g$

(b) When the standby-less system switchover (effects distributed) facility is used

Type	Formula
Number of message queue identifiers	b $\sum_{i=1} V_i + 2 \times a + 3 \times b + c + d + e + 26 + m$
Number of semaphore identifiers	b $\sum_{i=1} \{ \uparrow \{ Y_i \times (j + k) \} \div 64 \uparrow + W_i \} + 6 \times b + 2 + f$
Number of semaphores (total number of semaphores per identifier)	b $\sum_{i=1} \{ Y_i \times (j + k) + X_i \} + 26 \times b + 3 + g$

(c) Variables used in the formulas

a: Number of front-end servers in the server machine

b: Number of dictionary servers in the server machine + *n*

c: 4 (front-end server) or 0 (other server)

d: 8 (dictionary server) or 0 (other server)

e: $16 \times n$ (back-end server) or 0 (other server)

f and *g*: Add when the system switchover facility is being used; determine the value from the table below.

Condition		Value of f	Value of g
pd_ha_acttype=monitor (or omitted)		0	0
pd_ha_acttype=server	pd_ha_agent=standbyunit	1	<i>h</i>
	pd_ha_agent=server	1	$6 + 2 \times$ (number of BESs in the unit to which standby-less system switchover (1:1) is applied)
	pd_ha_agent=activeunits	0	0
	pd_ha_agent omitted	1	<i>i</i>
	pd_ha_server_process_standby=Y (or omitted)		
	pd_ha_server_process_standby=N	0	1

h: $6 + 2 \times$ (total number of front-end servers, dictionary servers, and back-end servers in the server machine)

i: 1 + (total number of front-end servers, dictionary servers, and back-end servers in the server machine)

j: Number of host BESs

k: Number of guest BESs

m: If there is a system manager unit, 3; otherwise, 0

n : One of the following:

- When the standby-less system switchover (effects distributed) facility is not used
Then number of back-end servers in the server machine
- When the standby-less system switchover (effects distributed) facility is used
number of host BESs in server machine + value of `pd_ha_max_act_guest_servers` operand

S_i : Number of `pdbuffer -r` operands defined for RDAREAs placed in each server

T_i : Number of `pdbuffer -i` operands defined for RDAREAs placed in each server

U_i : Number of `-o` options specified in the `pdbuffer` operand

V_i : 1 (if 1 or greater is specified in the `pd_max_ard_process` operand) or 0

W_i : 2 (if a value is specified in the `pd_dfw_awt_process` operand) or 0

X_i : Value of the `pd_dfw_awt_process` operand + 1 (if a value is specified in the `pd_dfw_awt_process` operand) or 0

Y_i : Number of `-c` options specified in the `pdbuffer` operand

20.6 Listen queue specified values

If HiRDB receives many connection requests simultaneously, the Listen queue used by the HiRDB server may become insufficient to handle them, in which case the `KFPAL1723-E` message is displayed to the client-side applications to notify them of the error. When this occurs, use an OS command to determine whether the Listen queue is insufficient. You can determine that the Listen queue is insufficient if the frequency of insufficient Listen queuing is greater than before the errors began and the errors continue to increase. For details about making this determination, refer to the OS documentation.

If you have determined that the Listen queue is insufficient, enlarge the Listen queue on the HiRDB server machine. To enlarge the Listen queue, have it steadily increase to the 200 level each. However, if the OS cannot enlarge the Listen queue, or if increasing up to the specified maximum number of concurrent connections does not stop the errors, take a measure such as increasing the HiRDB connection processing power by increasing the processing power of the server machine (such as by adding CPUs) or reduce the number of connection requests to HiRDB by reducing the maximum number of concurrent connections.

The table below shows the parameters that need to be changed (for Linux, the option settings file) when you change the value specified in the Listen queue. For details about the commands whose parameters are to be changed and how to use them, see the applicable OS documentation.

Table 20–5: Parameters for changing the value for the Listen queue

OS	Parameter to be changed [#]
HP-UX	<code>/dev/tcp tcp_conn_request_max</code>
Solaris	<code>/dev/tcp tcp_conn_req_max_q, /dev/tcp tcp_conn_req_max_q0</code>
AIX	<code>Somaxconn</code>
Linux	<code>/etc/sysctl.conf net.core.somaxconn</code>

[#]: The file depends on the OS and kernel versions being used. See the applicable OS documentation and set appropriate values, using the values provided in the table as guidelines. With some OS versions, parameter settings may not be necessary. If a kernel parameter is not supported by the OS in use, its setting is not needed.

21

Sample Files

This chapter describes the sample files provided by HiRDB (sample database, configuration, and UOC).

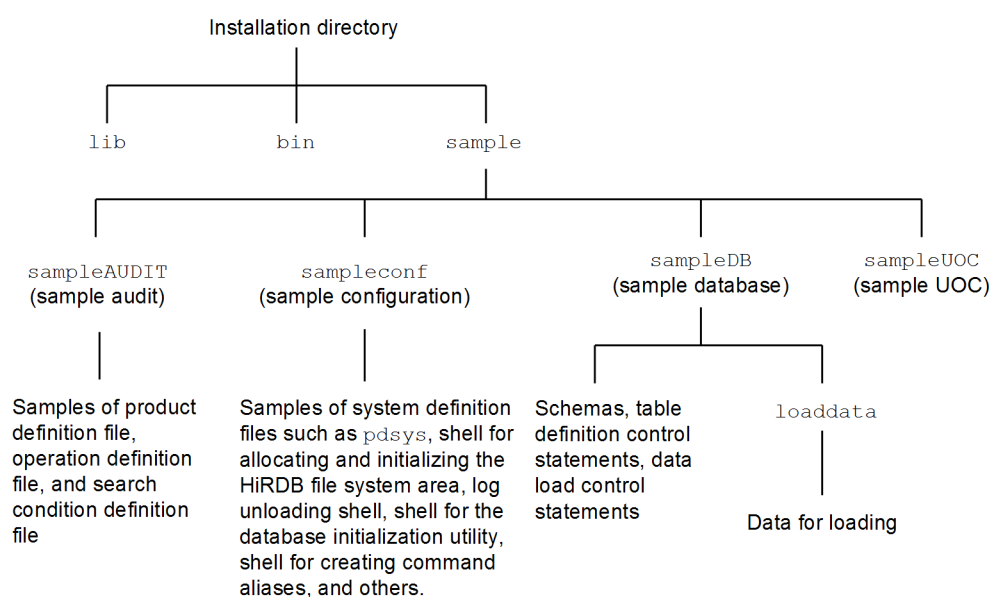
21.1 Overview of sample files

HiRDB provides the following sample files:

- Sample audit
- Sample database
- Sample configuration
- Sample UOC

The following figure shows the directory structure of the sample files. All the directories following `sample` are located under the installation directory. Executing the `pdsetup` command does not copy them to the HiRDB directory.

Figure 21–1: Directory structure of sample files



Note

The following are the installation directories:

HiRDB single server configuration	HiRDB parallel server configuration
/HiRDB_S/sample/sampleconf	/HiRDB_P/sample/sampleconf
/opt/HiRDB_S/sample/sampleconf	/opt/HiRDB_P/sample/sampleconf

21.1.1 Names of sample files

This section describes the names of the following sample files:

- Sample audit
- Sample database
- Sample configuration
- Sample UOC

(1) Sample audit file names

The sample audit files are used in connection with JP1/NETM/Audit. The following table lists their file names and descriptions.

Table 21–1: Sample audit file names and descriptions

File name	Description
HiRDB.conf	Product definition file
admjevlog_HiRDB.conf	Operation definition file
sampleaud1	Search condition definition files
sampleaud2	

For details about the environment settings that use these sample files, see *Linkage to JP1/NETM/Audit* in the *HiRDB Version 9 System Operation Guide*.

(2) Name of sample database file

The table below lists and describes the directories and files used with the sample database.

The sample files are provided in a Japanese version (containing single-byte kana characters) and an English version. Unless Shift JIS is being used as the character encoding set, the English version of the sample files should be used.

Table 21–2: Directories and files used with sample database

Name of directory or file	Contents	Remarks
tbcreate ^{#1}	Table definition statements (including schema definitions)	pddef input format
loadinf ^{#2}	Control statements for pdload	Shell
loaddata	Input data for data loading	Directory
CONTROL_FILE	Control statements for data loading	--

Legend:

--: Not applicable

#1: The file name of the English version is `tbcreate_e`.

#2: The file name of the English version is `loadinf_e`.

(3) Names of sample configuration files

The table below describes the sample configuration.

This sample configuration uses values based on the minimum configuration in order to simplify the relationships among parameters; these are not the optimum values.

Table 21–3: Contents of sample configuration

Classification	Contents	File name	
		HiRDB single server configuration	HiRDB parallel server configuration
System definitions	System common definition	Pdsys	pdsys
	Unit control information definition	Pdutsys	pdutsys1, pdutsys2
	Server common definition	--	pdsvrc
	Single server definition	sds01	--

Classification	Contents	File name	
		HiRDB single server configuration	HiRDB parallel server configuration
	Front-end server definition	--	fes1
	Back-end server definition	--	bes1, bes2
	Dictionary server definition	--	dic1
Allocation/ initialization of HiRDB file system areas	Shell for allocating RDAREAs	Fmkfile	fmkfile
	Shell for allocating system log, synchronization point dump, and status files	Sysfmkfs	sysfmkfs1, sysfmkfs2
	Shell for initializing system log, synchronization point dump, and status files	Sysfinit	sysfint
Log unloading	Shell for unloading system log files	logunld	feslogunld, bes1logunld, bes2logunld, diclogunld
Database initialization utility	Shell for creating control statements	mkinit	mkinit
	Database initialization utility control statements	rdinit01#	rdinit01#
Execution of operation commands under aliases	Sample of shell script for executing operation commands under aliases	Bourne shell	aliascmdbsh
		C shell	aliascmdcsh

Legend:

--: Not applicable

#: Can be generated by executing the control statement creation shell for the database initialization utility (mkinit).

(4) Sample UOC files

The UOC shown below is provided. The table below describes the sample UOC.

- Database load utility (pdload) file input example
- Database reconfiguration utility (pdrorg) file output example

For details about UOCs, see the manual *HiRDB Version 9 Command Reference*.

Table 21–4: Contents of sample UOC

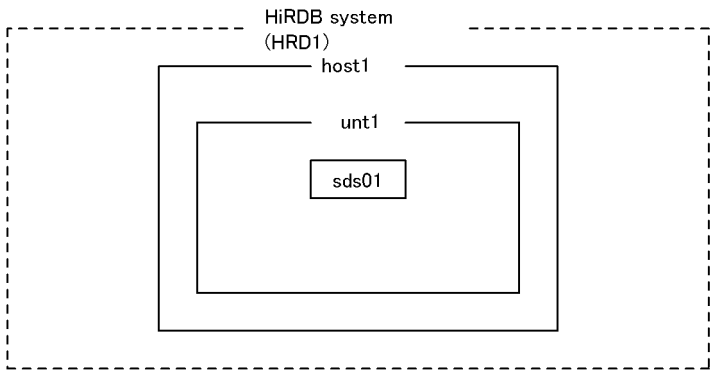
File name	Contents
sample1.c	Example of UOC for entering DAT-format input files by the database load utility
sample2.c	Example of UOC for entering binary format input files
sample4.c	Functions used by sample2.c
sampleA.c	Example of UOC that prevents output of unneeded data to an unload file
sampleB.c	Example of UOC that outputs a UOC data file

21.2 System configuration and table definition information

(1) System configuration

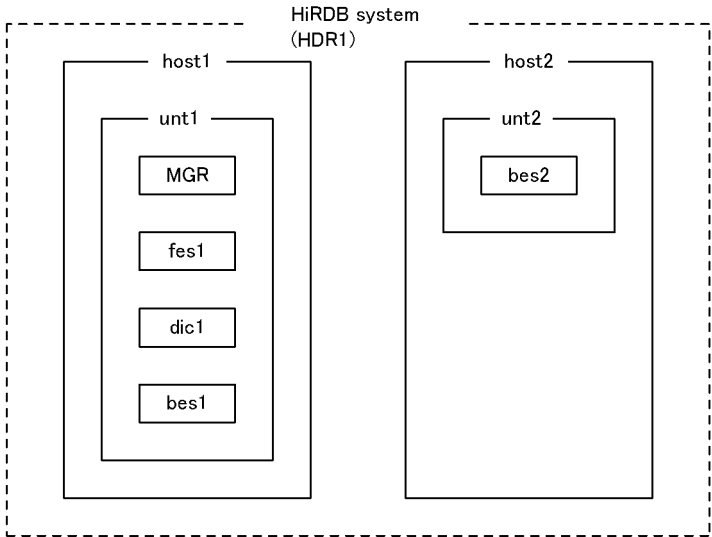
Figure 21-2 shows the relationships between the host and server for a HiRDB single server configuration, and Figure 21-3 shows the relationships among hosts and servers for a HiRDB parallel server configuration.

Figure 21–2: Configuration for a HiRDB single server configuration



HDR1: HiRDB identifier
host1: Host name
unt1: Unit identifier
sds01: Single server name

Figure 21–3: Configuration for a HiRDB parallel server configuration



HDR1: HiRDB identifier
host1, host2: Host names
unt1, unt2: Unit identifiers
MGR: System manager
fes1: Front-end server name
dic1: Dictionary server name
bes1, bes2: Back-end server names

(2) Table definition information

There is a Japanese version (containing single-byte kana characters) and an English version of the sample files for table definitions and data loading. Unless Shift JIS is being used as the character encoding set, the English version should be used.

Table 21-5 lists the tables that are to be defined, Table 21-6 shows the English version of the column attributes and indexes.

All the tables presented here have the `FIX` attribute.

Table 21–5: Contents of tables to be defined

Table name	Contents	Number of rows
CUSTOM	Customer master	100 ^{#1}
GOODS	Product master	100 ^{#1}
VENDOR	Vendor master	50 ^{#1}
TAKEODR	Orders received	-- ^{#2}
STOCK	Stock	100
WAREHUS	Warehousing	-- ^{#2}
SHIPMNT	Shipments	-- ^{#2}
SENDODR	Orders placed	-- ^{#2}
LAYIN	Purchasing	-- ^{#2}

Legend:

--: Not applicable

#1: Data loading is not applicable in the English version.

#2: Data loading is not applicable in either the Japanese or the English version.

Table 21–6: Column attributes and indexes (English version)

Table name	Column name	Column attribute	Index name
CUSTOM	CUSTOM_CD	CHAR(5)	UNIQUE CLUSTER
	CUSTOM_NAME	CHAR(30)	KEY CUSTOMX
	TELNO	CHAR(12)	
	ZIPCD	CHAR(3)	
	ADDRESS	CHAR(30)	
GOODS	PRODUCT_CD	CHAR(6)	UNIQUE CLUSTER
	PRODUCT_NAME	CHAR(30)	KEY GOODSX
	PRICE	DECIMAL(7,0)	
	VENDOR_CD	CHAR(5)	
VENDOR	VENDOR_CD	CHAR(5)	UNIQUE CLUSTER
	VENDOR_NAME	CHAR(30)	KEY VENDORX
	TELNO	CHAR(12)	
	ZIPCD	CHAR(3)	

Table name	Column name	Column attribute	Index name
TAKEODR	ADDRESS	CHAR(30)	CLUSTER KEY
	ORDER_ACCEPTED_CD	CHAR(7)	
	CUSTOM_CD	CHAR(5)	
	PRODUCT_CD	CHAR(6)	
	QUANTITY	DECIMAL(7,0)	
	RESERVED_QUANTITY	DECIMAL(7,0)	
	SURPLUS	DECIMAL(7,0)	
	ORDER_ACCEPTED_DATE	CHAR(6)	
STOCK	DELIVERY_DATE	CHAR(6)	
	PRODUCT_CD	CHAR(6)	
	STOCK	DECIMAL(7,0)	
	RESERVED_QUANTITY	DECIMAL(7,0)	
	ORDER	DECIMAL(7,0)	
WAREHUS	VENDOR_CD	CHAR(5)	
	PRODUCT_CD	CHAR(6)	
	WAREHOUSE	DECIMAL(7,0)	
	LAY_IN_NO	INTEGER	
SHIPMNT	WAREHOUSE_DATE	CHAR(6)	
	PRODUCT_CD	CHAR(6)	
	SHIPMENT	DECIMAL(7,0)	
	ORDER_ACCEPTED_CD	CHAR(7)	
SENDODR	ORDER_ACCEPTED_DATE	CHAR(6)	
	ORDER_NO	INTEGER	CLUSTER KEY SENDODRX
	VENDOR_CD	CHAR(5)	
	PRODUCT_CD	CHAR(6)	
	ORDER_QUANTITY	DECIMAL(7,0)	
	ORDER_DATE	CHAR(6)	
LAYIN	DELIVERY_DATE	CHAR(6)	
	LAY_IN_NO	INTEGER	CLUSTER KEY LAYINX
	VENDOR_CD	CHAR(5)	
	PRODUCT_CD	CHAR(6)	
	LAY_IN_QUANTITY	DECIMAL(7,0)	
	LAY_IN_DATE	CHAR(6)	

Notes

The name of the table storage RDAREA is RDDATA10.

The name of the index storage RDAREA is RDINDX10.

21.3 Use of the sample files

21.3.1 Creating the configuration files

Table 21-7 lists and explains the configuration files that are to be created for the HiRDB single server configuration; Table 21-8 lists and explains the configuration files that are to be created for the HiRDB parallel server configuration.

To use these files, they must be copied under `$PDDIR/conf`, then appropriate changes must be made to them according to the notes provided in Table 21-7 or 21-8.

Specify the HiRDB directory path in the `PDDIR` environment variable.

- HiRDB single server configuration: `/opt/HiRDB_S`
- HiRDB parallel server configuration: `/opt/HiRDB_P`

Table 21–7: Files to be created for a HiRDB single server configuration

File name	Contents	Notes
<code>pdsys</code>	System common definition	<ul style="list-style-type: none"> • Port number is set to 22200. Change it as appropriate for the execution environment. • Host name is set to <code>host1</code>. Change it as appropriate for the execution environment.
<code>pduatsys</code>	Unit control information definition	None
<code>sds01</code>	Single server definition	
<code>sysfmkfs</code>	Command for creating HiRDB file system areas for system files	
<code>sysfint</code>	Command for initializing system files	
<code>logunld</code>	Command for unloading system log files	<ul style="list-style-type: none"> • An unload log file storage directory (<code>/HiRDB_S/unloadlog</code>) must have been created beforehand. • To change the unload load file storage directory, specify <code>-o</code> in <code>pdlogunld</code>.

Table 21–8: Files to be created for a HiRDB parallel server configuration

File name	Contents	Notes
<code>pdsys</code>	System common definition	<ul style="list-style-type: none"> • Port number is set to 22200. Change it as appropriate for the execution environment. • Host names are set to <code>host1</code> and <code>host2</code>. Change them as appropriate for the execution environment.
<code>pduatsys1</code>	Unit control information definition for <code>unt1</code> of <code>host1</code>	After copying it to <code>host1</code> 's environment, change the file name to <code>pduatsys</code> .
<code>pduatsys2</code>	Unit control information definition for <code>unt2</code> of <code>host2</code>	After copying it to <code>host2</code> 's environment, change the file name to <code>pduatsys</code> .
<code>pdsvrc</code>	Server common definition	None
<code>fes1</code>	Definition for front-end server <code>fes1</code>	
<code>bes1</code>	Definition for back-end server <code>bes1</code>	
<code>bes2</code>	Definition for back-end server <code>bes2</code>	
<code>dic1</code>	Definition for dictionary server <code>dic1</code>	

File name	Contents	Notes
sysfmkfs1	Command for creating HiRDB file system areas for system files to be used by unt1 of host1	Input from host1.
sysfmkfs2	Command for creating HiRDB file system areas for system files to be used by unt2 of host2	Input from host2.
sysfint	Command for initializing system files to be used by host1 and host2	<ul style="list-style-type: none"> Input from host1. Host names are host1 and host2. Change them as appropriate for the execution environment.
Feslogunld	Command for unloading system log files for pdfes	<ul style="list-style-type: none"> Unload log file storage directory /HiRDB_P/unloadlog must have been created beforehand for each host. To change the unload load file storage directory, specify -o in pdlogunld.
bes1logunld	Command for unloading system log files for pdbes1	
bes2logunld	Command for unloading system log files for pdbes2	
dic1logunld	Command for unloading system log files for pddic	

(1) Allocating and initializing HiRDB file system areas

(a) Creating a HiRDB file system area for the database

The following instruction must be executed to create the HiRDB file system areas:

HiRDB single server configuration

Execute the following shell script:

```
$PDDIR/sample/sampleconf/fmkfile
```

Note

Be sure to set the PDDIR environment variable.

HiRDB parallel server configuration

Execute the following shell script at each host (host1 and host2):

```
$PDDIR/sample/sampleconf/fmkfile
```

Note

Be sure to set the PDDIR environment variable.

(b) Allocating files such as system log files

The following instruction must be executed to allocate files such as system log files:

HiRDB single server configuration

Execute the following shell script:

```
$PDDIR/sample/sampleconf/sysfmkfs
```

Note

Be sure to set the PDDIR environment variable.

HiRDB parallel server configuration

Execute the following shell script at host1:

```
$PDDIR/sample/sampleconf/sysfmkfs1
```

Note

Be sure to set the `PDDIR` environment variable.

Execute the following shell script at `host2`:

```
$PDDIR/sample/sampleconf/sysfmkfs2
```

Note

Be sure to set the `PDDIR` environment variable.

(c) Initializing files such as system log files

The following instruction must be executed to initialize files such as system log files:

HiRDB single server configuration

Execute the following shell script:

```
$PDDIR/sample/sampleconf/sysfint
```

Note

Be sure to set the `PDDIR` environment variable.

HiRDB parallel server configuration

Execute the following shell script at the host where the system manager is defined (`host1`):

```
$PDDIR/sample/sampleconf/sysfint
```

Note

Be sure to set the `PDDIR` environment variable.

(2) Creating the initialization control statement file

The instruction shown as follows must be executed to create the initialization control statement file.

The name of the initialization control statement file is `$PDDIR/sample/sampleconf/rdinit01`.

HiRDB single server configuration

Execute the following shell script:

```
$PDDIR/sample/sampleconf/mkinit
```

Note

Be sure to set the `PDDIR` environment variable.

HiRDB parallel server configuration

Execute the following shell script at the dictionary server:

```
$PDDIR/sample/sampleconf/mkinit
```

Note

Be sure to set the `PDDIR` environment variable.

(3) Starting HiRDB

Use the `pdstart` command to start HiRDB.

(4) Table definition

The database definition utility (`pddef`) is executed shown as follows:

```
$PDDIR/bin/pddef < $PDDIR/sample/sampleDB/tblecreate_e
```

Note

Set appropriate values in the `PDUSER`, `PDDIR`, `PDNAMEPORT`, and `PDHOST` environment variables.

(5) Data loading

Execute the following shell script to load data using the database load utility (`pdload`). For details about data loading, see the manual *HiRDB Version 9 Command Reference*.

```
$PDDIR/sample/sampleDB/loadinf_e
```

(6) Creating a shell script for executing operation commands under aliases

It may not be possible to execute a HiRDB operation command because it has the same name as an OS command or a command provided by another program. In this case, the following actions can be taken:

- Make the environment variable setting that gives HiRDB commands precedence over other commands.
- Specify the absolute path of the command to be executed.

If neither of these actions can be taken, there is a way to execute a HiRDB operation command under a user-defined name. HiRDB provides a sample shell script for this purpose.

(a) Names of sample files for shell script provided by HiRDB

HiRDB provides a sample shell script file appropriate to each platform. The table below lists and describes the sample shell script files provided by HiRDB; they are stored in the following directory:

- HiRDB single server configuration: `/opt/HiRDB_S/sample/sampleconf`
- HiRDB parallel server configuration: `/opt/HiRDB_P/sample/sampleconf`

Table 21–9: Sample shell script files for executing commands under aliases

File name	Contents	Notes
<code>aliascmdbsh</code>	Sample file for Bourne shell	Do not copy in the <code>bin</code> or <code>lib</code> directory under the HiRDB directory
<code>aliascmdcsh</code>	Sample file for C shell	

(b) Procedure for creating an alias for a command

To create an alias for an operation command:

1. Copy the shell script sample file into a desired directory. To create aliases for multiple commands, copy it once for each of the commands. Do not copy it into the `bin` or `lib` directory under the HiRDB directory.
2. Set the copy target directory for the sample file in the `PATH` environment variable, or set `path` as the search path.
3. Rename the file copied in step 1 to the alias of the HiRDB operation command. For example, command name `pdmod` might be changed to `hirmod`.
4. Open the copied sample file and change `cc...cc` to the name of the HiRDB operation command to be executed under the alias, as shown in Figures 21-4 and 21-5.

Figure 21–4: Sample file for Bourne shell

```
#!/bin/sh -f

#set HiRDB command here
HIRDB_COMMAND=cc...cc ← Set the HiRDB operation command name here (for example, to
                                change pdmod to
                                hirmod, set pdmod
                                here).

${PDDIR}/bin/${HIRDB_COMMAND} $*
exit $?
```

Figure 21–5: Sample file for C shell

```
#!/bin/csh -f

#set HiRDB command here
set HIRDB_COMMAND=cc...cc← Set the HiRDB operation command name here (for example, to
change pdmod to
hirmod, set pdmod
here).

${PDDIR}/bin/${HIRDB_COMMAND} $argv
exit $status
```

This procedure enables a HiRDB operation command to be executed under any desired name. Options can be specified in the alias command in the same manner as with the normal HiRDB operation command.

(c) Notes

1. A name other than the HiRDB operation command name must be assigned to a copy of the sample file.
2. It is possible that the `$PDDIR/bin` and `$PDDIR/lib` directories under the HiRDB directory will be deleted in their entirety when the `pdsetup -d` command is executed. For this reason, sample files must not be copied into these directories.
3. The contents of the sample files must not be changed, except for setting a HiRDB operation command name.
4. To cancel execution of a created alias command during command processing, the HiRDB command process must be terminated at an extension of the alias process. Terminating the alias process does not automatically terminate the HiRDB command process.
5. If a created alias command is executed and another process is terminated while the HiRDB command is waiting for a response to be entered, HiRDB command execution may result in an error or the response entry wait status may still be in effect. If the response entry wait status is still in effect, the HiRDB command process must be terminated.

21.3.2 HiRDB file system area names and user-created file names used with sample database

This section lists the names and sizes of the HiRDB file system areas and the names of the user-created files that are used with the sample database.

These are the names used in the provided sample database; any names could have been used.

(1) Names and sizes of HiRDB file system areas

Table 21-10 lists the names and sizes of the HiRDB file system areas that are used with the database for the HiRDB single server configuration. Table 21-11 lists the names and sizes of the HiRDB file system areas that are used with the database for the HiRDB parallel server configuration.

Table 21–10: Names and sizes of HiRDB file system areas: HiRDB single server configuration

No.	Classification	Files	Size (MB)	HiRDB file system area name	Remarks
1	System files	<ul style="list-style-type: none"> • System log files • Synchronization point dump files • Status files 	74	/HiRDB_S/rdsys011 /HiRDB_S/rdsys012 /HiRDB_S/rdsys013 /HiRDB_S/rdsys014 /HiRDB_S/rdsys015 /HiRDB_S/rdsys016	Regular files
2	System RDAREAs	<ul style="list-style-type: none"> • Master directory 	20	/HiRDB_S/rdsys02	

No.	Classification	Files	Size (MB)	HiRDB file system area name	Remarks
		<ul style="list-style-type: none"> Data directory Data dictionary 			
3	Work table files	--	20	/HiRDB_S/rdsys03	
4	User RDAREAs	--	40	/HiRDB_S/rdsys04	

Legend:

--: Not applicable

Table 21–11: Names and sizes of HiRDB file system areas: HiRDB parallel server configuration

No.	Classification	Files		Size (MB)	HiRDB file system area name	Remarks
1	System files	<ul style="list-style-type: none">• System log files• Synchronization point dump files• Status files	host1	156	/HiRDB_P/rdsys011 /HiRDB_P/rdsys012 /HiRDB_P/rdsys013 /HiRDB_P/rdsys014	Regular files
			host2	74	/HiRDB_P/rdsys015 /HiRDB_P/rdsys016	
2	System RDAREAs	<ul style="list-style-type: none">• Master directory• Data directory• Data dictionary		20	/HiRDB_P/rdsys02	
3	Work table files	--		20	/HiRDB_P/rdsys03	
4	User RDAREAs	--		40	/HiRDB_P/rdsys04	

Legend:

--: Not applicable

(2) Names of user-created files

Table 21-12 lists the names of the user-created files that are used with the sample database for the HiRDB single server configuration. Table 21-13 lists the names of the user-created files that are used with the sample database for the HiRDB parallel server configuration.

Table 21–12: Names of user-created files: HiRDB single server configuration

No.	Type		File name	Remarks
1	Definition files	System common definition	\$PDCONFPATH/pdsys	The directories have already been created during installation.
2		Unit control information definition	\$PDDIR/conf/pdutsys	
3		Single server definition	\$PDDIR/conf/sds01	
4	System log files		/HiRDB_S/rdsys011/log1 /HiRDB_S/rdsys012/log2 /HiRDB_S/rdsys013/log3 /HiRDB_S/rdsys014/log4	4 groups
5	Synchronization point dump files		/HiRDB_S/rdsys014/spd1 /HiRDB_S/rdsys015/spd2 /HiRDB_S/rdsys016/spd3	3 groups

No.	Type		File name	Remarks
6	Status files	Unit status files	/HiRDB_S/rdsys011/utsts1a /HiRDB_S/rdsys012/utsts1b /HiRDB_S/rdsys013/utsts2a /HiRDB_S/rdsys014/utsts2b /HiRDB_S/rdsys015/utsts3a /HiRDB_S/rdsys016/utsts3b	2 per unit x 3
7		Server status files	/HiRDB_S/rdsys011/sts1a /HiRDB_S/rdsys012/sts1b /HiRDB_S/rdsys013/sts2a /HiRDB_S/rdsys014/sts2b /HiRDB_S/rdsys015/sts3a /HiRDB_S/rdsys016/sts3b	2 per server x 3
8	System RDAREAs	Master directory	/HiRDB_S/rdsys02/rdmast	RDAREA name: RDMAST
9		Data directory	/HiRDB_S/rdsys02/rddirt	RDAREA name: RDDIRT
10		Data dictionary	/HiRDB_S/rdsys02/rddict	RDAREA name: RDDICT
11	Work table files		/HiRDB_S/rdsys03	--
12	RPC trace files		/HiRDB_S/spool/pdrpctr	--
13	User RDAREAs	Data section	/HiRDB_S/rdsys04/rddata10	RDAREA name: RDDATA10
14		Index section	/HiRDB_S/rdsys04/rdindx10	RDAREA name: RDINDX10

Legend:

--: Not applicable

Table 21–13: Names of user-created files: HiRDB parallel server configuration

No.	Type		File name		Remarks
1	Definition files	System common definition		\$PDCONFPATH/pdsys	The directories have already been created during installation.
2		Unit control information definition		\$PDDIR/conf/pdutsys	
3		Server common definition		\$PDCONFPATH/pdsvrc	
4		Front-end server definition		\$PDDIR/conf/fes1	
5		Dictionary server definition		\$PDDIR/conf/dic1	
6		Back-end server definition	host1	\$PDDIR/conf/bes1	
	host2		\$PDDIR/conf/bes2		
7	System log files		host1	/HiRDB_P/rdsys011/feslog1 /HiRDB_P/rdsys012/feslog2 /HiRDB_P/rdsys013/feslog3 /HiRDB_P/rdsys014/feslog4	4 groups per server

No.	Type		File name	Remarks
			/HiRDB_P/rdsys011/diclog1 /HiRDB_P/rdsys012/diclog2 /HiRDB_P/rdsys013/diclog3 /HiRDB_P/rdsys014/diclog4 /HiRDB_P/rdsys011/bes1log1 /HiRDB_P/rdsys012/bes1log2 /HiRDB_P/rdsys013/bes1log3 /HiRDB_P/rdsys014/bes1log4	
		host2	/HiRDB_P/rdsys011/bes2log1 /HiRDB_P/rdsys012/bes2log2 /HiRDB_P/rdsys013/bes2log3 /HiRDB_P/rdsys014/bes2log4	
8	Synchronization point dump files		host1 /HiRDB_P/rdsys014/fesspd1 /HiRDB_P/rdsys015/fesspd2 /HiRDB_P/rdsys016/fesspd3 /HiRDB_P/rdsys014/dicspd1 /HiRDB_P/rdsys015/dicspd2 /HiRDB_P/rdsys016/dicspd3 /HiRDB_P/rdsys014/bes1spd1 /HiRDB_P/rdsys015/bes1spd2 /HiRDB_P/rdsys016/bes1spd3 host2 /HiRDB_P/rdsys014/bes2spd1 /HiRDB_P/rdsys015/bes2spd2 /HiRDB_P/rdsys016/bes2spd3	3 groups per server
9	Status files	Unit status files	host1 /HiRDB_P/rdsys011/ut1sts1a /HiRDB_P/rdsys012/ut1sts1b /HiRDB_P/rdsys013/ut1sts2a /HiRDB_P/rdsys014/ut1sts2b /HiRDB_P/rdsys015/ut1sts3a /HiRDB_P/rdsys016/ut1sts3b host2 /HiRDB_P/rdsys011/ut2sts1a /HiRDB_P/rdsys012/t2sts1b /HiRDB_P/rdsys013/ut2sts2a /HiRDB_P/rdsys014/ut2sts2b /HiRDB_P/rdsys015/ut2sts3a /HiRDB_P/rdsys016/ut2sts3b	2 per unit x 3
10	Status files	Server status files	host1 /HiRDB_P/rdsys011/fessts1a /HiRDB_P/rdsys012/fessts1b /HiRDB_P/rdsys013/fessts2a /HiRDB_P/rdsys014/fessts2b /HiRDB_P/rdsys015/fessts3a /HiRDB_P/rdsys016/fessts3b /HiRDB_P/rdsys011/dicsts1a /HiRDB_P/rdsys012/dicsts1b /HiRDB_P/rdsys013/dicsts2a	2 per server x 3

No.	Type			File name	Remarks
				/HiRDB_P/rdsys014/dicsts2b /HiRDB_P/rdsys015/dicsts3a /HiRDB_P/rdsys016/dicsts3b /HiRDB_P/rdsys011/bes1sts1a /HiRDB_P/rdsys012/bes1sts1b /HiRDB_P/rdsys013/bes1sts2a /HiRDB_P/rdsys014/bes1sts2b /HiRDB_P/rdsys015/bes1sts3a /HiRDB_P/rdsys016/bes1sts3b	
			host2	/HiRDB_P/rdsys011/bes2sts1a /HiRDB_P/rdsys012/bes2sts1b /HiRDB_P/rdsys013/bes2sts2a /HiRDB_P/rdsys014/bes2sts2b /HiRDB_P/rdsys015/bes2sts3a /HiRDB_P/rdsys016/bes2sts3b	
11	System RDAREAs	Master directory		/HiRDB_P/rdsys02/rdmast	RDAREA name: RDMAST
12		Data directory		/HiRDB_P/rdsys02/rddirt	RDAREA name: RDDIRT
13		Data dictionary		/HiRDB_P/rdsys02/rddict	RDAREA name: RDDICT
14	Work table files			/HiRDB_P/rdsys03	--
15	RPC trace files			/HiRDB_P/spool/pdrpctr	--
16	User RDAREAs	bes1	Data section	/HiRDB_P/rdsys04/rddata10	RDAREA name: RDDATA10
17			Index section	/HiRDB_P/rdsys04/rdindx10	RDAREA name: RDINDX10
18		bes2	Data section	/HiRDB_P/rdsys04/rddata20	RDAREA name: RDDATA20
19			Index section	/HiRDB_P/rdsys04/rdindx20	RDAREA name: RDINDX20

Legend:

--: Not applicable

22

Communication Between HiRDB Servers and HiRDB Clients

This chapter explains how to connect HiRDB clients with HiRDB servers. It also describes the settings for a DNS server and for a firewall.

22.1 Connecting HiRDB clients to a HiRDB server

To connect a HiRDB client to a HiRDB server, the `Hirdb` system's host name (or IP address) must be specified in the following operands of the client environment definition:

- `PDHOST`
- `PDFESHOST`

The host name specified in these operands must be the same host name specified in the `pdunit` operand of the system common definition.

With some network configurations, a connection might not be established using the host name specified in the `pdunit` operand. In an environment using DNS, see *22.1.1 Connection to a HiRDB server with an FQDN specified*. If the network used among HiRDB servers does not match the network used between a HiRDB client and a HiRDB server, see *22.1.2 Using the multi-connection address facility to connect to a HiRDB server*.

22.1.1 Connection to a HiRDB server with an FQDN specified

The host name specified in the `pdunit` operand must be registered, together with the IP address, in the host's file at every client machine that accesses the HiRDB server. Use of DNS eliminates the need for registration in the `hosts` file, thereby eliminating the need for modifying the `hosts` file that is associated with registration and IP address changes.

Connection with a HiRDB server running on a host in a domain can be established by specifying the server machine's fully qualified domain name (FQDN) in `PDHOST` and `PDFESHOST`.

The following table lists names that can be specified in client environment definition.

Table 22–1: Names allowed in client environment definition

Name specified in client environment definition	Version 05-02 or earlier	Version 05-03 or later
Host name	S	S
FQDN	--	S [#]

S: Can be specified.

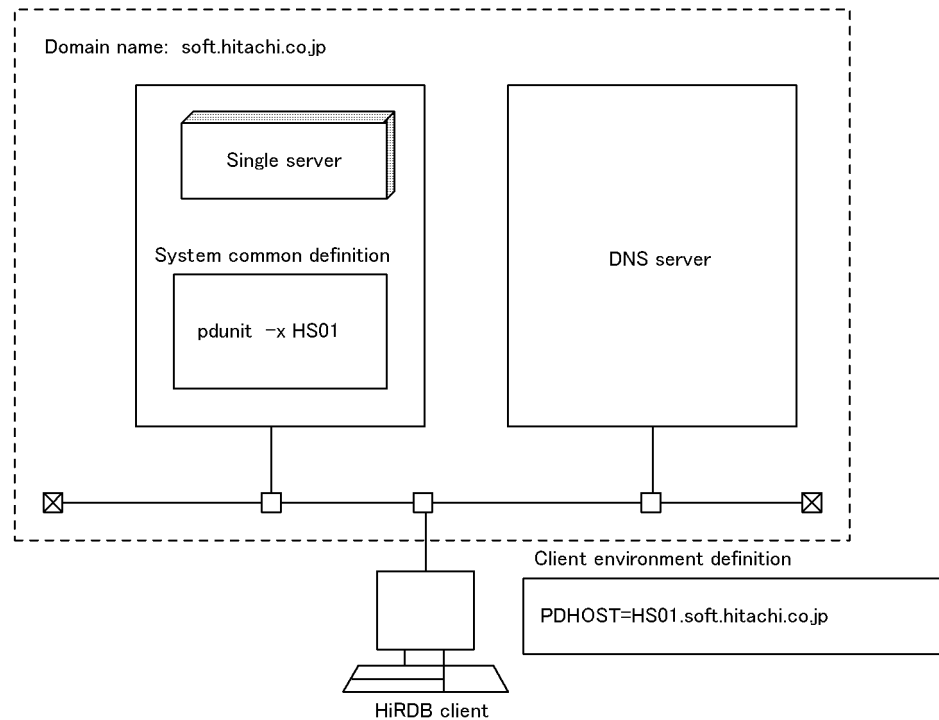
--: Cannot be specified.

[#]: This is used in a large-scale network environment to avoid having to modify the host's file when a host name or IP address is registered or when an IP address is changed.

(1) Example of network configuration and definition for connecting to a HiRDB server with an FQDN specified

The following figure shows an example of a network configuration and definition for connecting to a HiRDB system with an FQDN specified.

Figure 22-1: Example of network configuration and definition for connecting to HiRDB system with an FQDN specified



Explanation:

- The host name (HS01) in the network that is used by the HiRDB system is specified in the `-x` option of the `pdunit` operand.
- The HiRDB system's FQDN (HS01.soft.hitachi.co.jp) is specified in `PDHOST` in the client environment definition.

(2) Notes

1. An FQDN cannot be specified in `PDHOST` or `PDFESHOST` of the client environment definition in the case of connection to a HiRDB system whose version is earlier than 05-03. If specified, a server process might not be able to execute cancellation processing after the maximum client wait time (value specified in `PDCWAITTIME`) has elapsed.
2. An FQDN cannot be specified as a host name in a HiRDB system.
3. If the HiRDB server and HiRDB client use different networks, the multi-connection address facility must be used to connect to the HiRDB system; for details, see 22.1.2 *Using the multi-connection address facility to connect to a HiRDB server*.

22.1.2 Using the multi-connection address facility to connect to a HiRDB server

It might not be possible in some network configurations to connect to a HiRDB system even though the host name is specified in the `pdunit` operand. This happens when the network between the HiRDB client and HiRDB system is different from the network connecting the HiRDB system's server machines.

In such a case, the multi-connection address facility can be used. This facility enables connection to the HiRDB system without having to specify the same host name in the `PDHOST/PDFESHOST` operand and the `pdunit` operand.

(1) Using the multi-connection address facility

To use the multi-connection address facility, the `-m` option of the `pdstart` operand must be specified in the system common definition.

If you are using the system switchover configuration, even if the IP address of the network used between the HiRDB servers is not inherited, make sure to use a configuration that inherits the IP address of the network used between the HiRDB client and server (the IP address that the client connects to). For an example of a definition in such a case, see (c) *HiRDB parallel server configuration (with inheritance of IP addresses during system switchover)*.

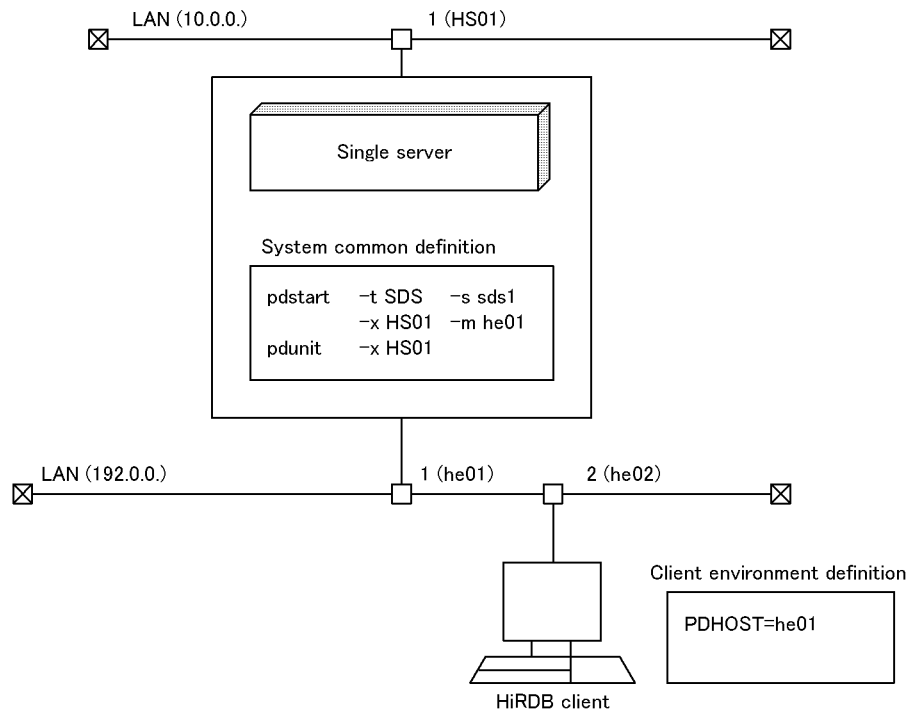
The HiRDB client specifies in the `-m` and `-n` options the host name of the HiRDB system to which connection can be established in the network; this does not have to be the host name specified in the `pdunit` operand.

(2) Examples of network configurations and definitions using the multi-connection address facility

(a) HiRDB single server configuration

The following figure shows an example of a network configuration and definition using the multi-connection address facility (for a HiRDB single server configuration).

Figure 22–2: Example of network configuration and definition: HiRDB single server configuration



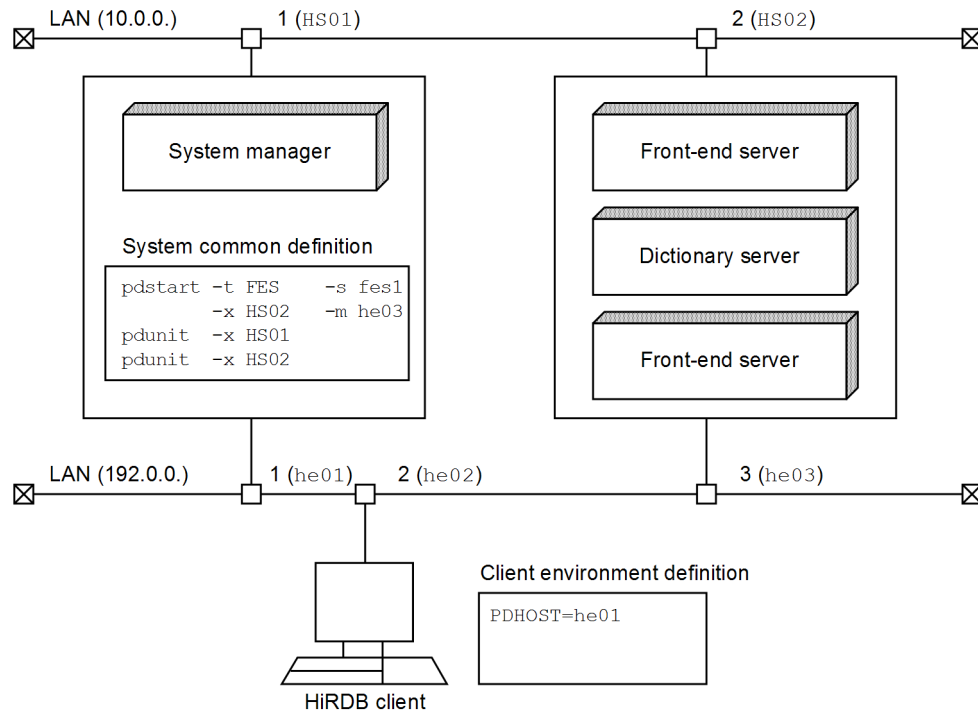
Explanation:

- The host name (HS01) used for the network communications between the HiRDB systems is specified in the `-x` option of the `pdunit` operand.
- The host name (he01) in the network that is used between the HiRDB client and the HiRDB single server configuration is specified in the `-m` option of the `pdstart` operand.
- The host name (he01) in the network that is used between the HiRDB client and the HiRDB single server configuration is specified in the `PDHOST` operand in the client environment definition.

(b) HiRDB parallel server configuration

The following figure shows an example of a network configuration and definition using the multi-connection address facility (for a HiRDB parallel server configuration).

Figure 22–3: Example of network configuration and definition: HiRDB parallel server configuration



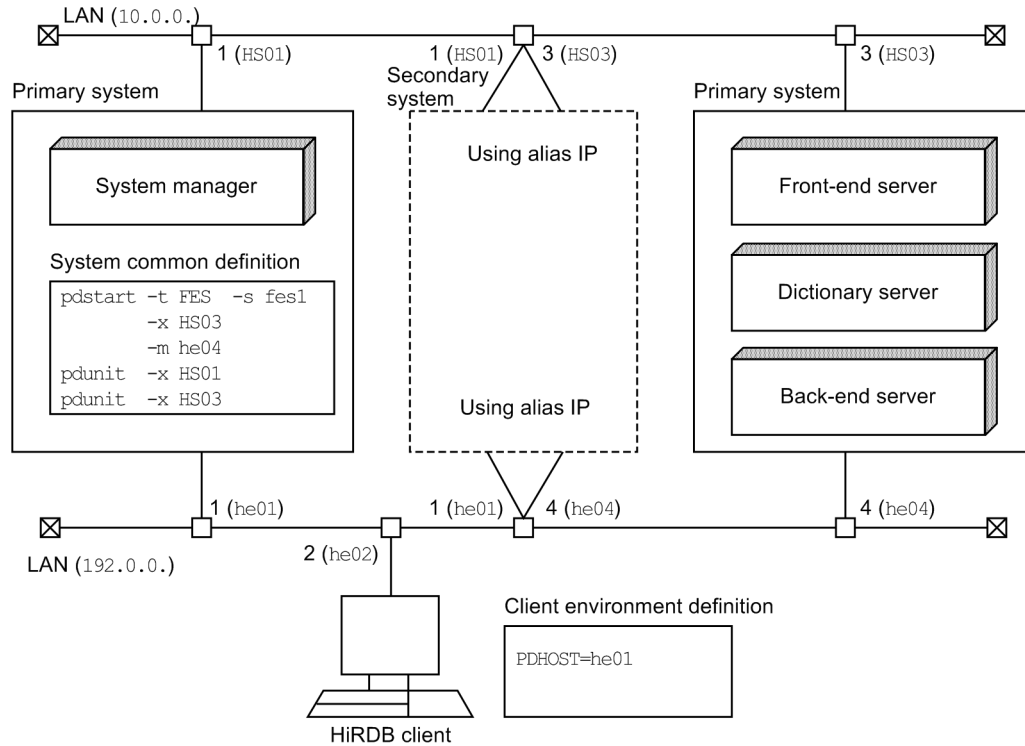
Explanation:

- The host names (HS01 and HS02) used for the network communications between the HiRDB systems are specified in the `-x` option of the `pdunit` operand.
- The host name (he03) in the network that is used between the HiRDB client and the HiRDB system is specified in the `-m` option of the `pdstart` operand (for defining the front-end server).
- The host name (he01) where the system manager is located in the network that is used between the HiRDB client and the HiRDB system is specified in the `PDHOST` operand in the client environment definition.

(c) HiRDB parallel server configuration (with inheritance of IP addresses during system switchover)

The following figure shows an example of a network configuration and definition using the multi-connection address facility (with inheritance of IP addresses during system switchover).

Figure 22–4: Example of network configuration and definition: With inheritance of IP addresses during system switchover



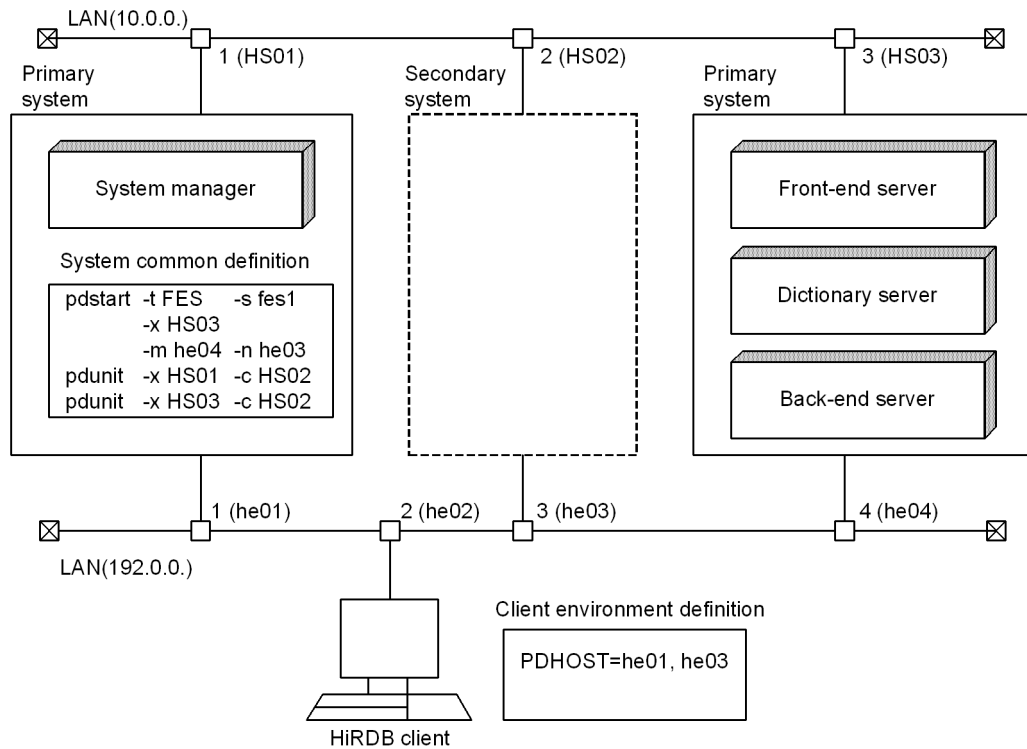
Explanation:

- The host names (HS01 and HS03) used for the network communications between the HiRDB systems are specified in the `-x` option of the `pdunit` operand.
- The host name (he04) corresponding to the IP address of the network that is used between the HiRDB client and the HiRDB system is specified in the `-m` option of the `pdstart` operand (for defining the front-end server). The `-n` option is omitted.
- The host name (he01) where the system manager is located in the network that is used between the HiRDB client and the HiRDB system is specified in the `PDHOST` operand in the client environment definition.

(d) HiRDB parallel server configuration (without inheritance of IP addresses during system switchover)

The following figure shows an example of a network configuration and definition using the multi-connection address facility (without inheritance of IP addresses during system switchover).

Figure 22–5: Example of network configuration and definition: Without inheritance of IP addresses during system switchover



Explanation:

- The host names (HS01 and HS03) used for the network communications between the HiRDB systems are specified in the `-x` option of the `pdunit` operand. The host name of the secondary system (HS02) is specified in the `-c` option.
- The host name (he04) in the network that is used between the HiRDB client and the HiRDB system is specified in the `-m` option of the `pdstart` operand (for defining the front-end server). The host name of the secondary system (he03) is specified in the `-n` option.
- The host name (he01) where the system manager is located in the network that is used between the HiRDB client and the HiRDB system is specified in the `PDHOST` operand in the client environment definition. The host name of the secondary system (he03) is also specified.

22.2 Settings for a DNS server to manage IP addresses

There are two ways for a HiRDB system to use a DNS server to manage IP addresses:

- The server machines reside in the same domain
- The server machines that make up a single HiRDB reside across multiple domains

This section describes how to manage each of these configurations.

(1) How to set up HiRDB in a single domain

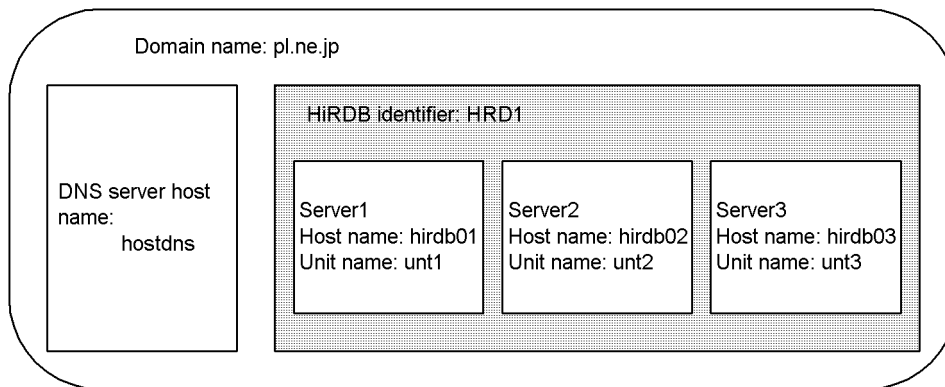
If the server machines reside in the same domain, specify either the host name or the FQDN (fully-qualified domain name; maximum of 32 characters) as the host name in the `pdunit` and `pdstart` operands. In this way, a DNS server can manage IP addresses, thereby making the `hosts` file unnecessary.

Basically, the host name or FQDN is specified using the following options:

- `pdunit` operand: `-x` and `-c` options
- `pdstart` operand: `-x`, `-m`, and `-n` options

The following figure shows an example of a system configuration using a single domain.

Figure 22–6: Example system configuration using a single domain



The following is an example of how to specify `pdunit -x` in this case:

- For specifying the host name


```
pdunit -x hirdb01 -u unt1 -d "operating-directory-name" -p port-number ...
pdunit -x hirdb02 -u unt2 -d "directory-name" -p port-number ...
pdunit -x hirdb03 -u unt3 -d "directory-name" -p port-number ...
```
- For specifying the FQDN


```
pdunit -x hirdb01.pl.ne.jp -u unt1 -d "operating-directory-name" -p port-number ...
pdunit -x hirdb02.pl.ne.jp -u unt2 -d "directory-name" -p port-number ...
pdunit -x hirdb03.pl.ne.jp -u unt3 -d "directory-name" -p port-number ...
```

(2) How to set up HiRDB in multiple domains

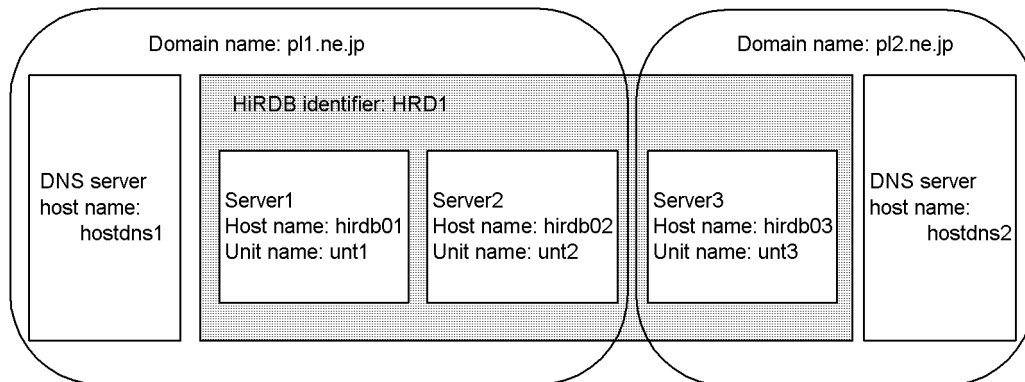
If the server machines reside in multiple domains, specify the FQDN (fully-qualified domain name; maximum of 32 characters) as the host name in the `pdunit` and `pdstart` operands. In this way, a DNS server can manage the IP addresses, thereby making the `hosts` file unnecessary.

Basically, the FQDN is specified using the following options.

- `pdunit` operand: `-x` and `-c` options
- `pdstart` operand: `-x`, `-m`, and `-n` options

The following figure shows an example of a system configuration using multiple domains.

Figure 22–7: Example system configuration using multiple domains



The following is an example of how to specify `pdunit -x` in this case:

- Specifying the FQDN

```
pdunit -x hirdb01.pl1.ne.jp -u unt1 -d "operating-directory-name" -p port-number ...
pdunit -x hirdb02.pl1.ne.jp -u unt2 -d "directory-name" -p port-number ...
pdunit -x hirdb03.pl2.ne.jp -u unt3 -d "directory-name" -p port-number ...
```

22.3 Settings when a firewall and NAT are installed

This section describes the HiRDB environment settings when a firewall and NAT are installed between HiRDB servers and HiRDB clients.

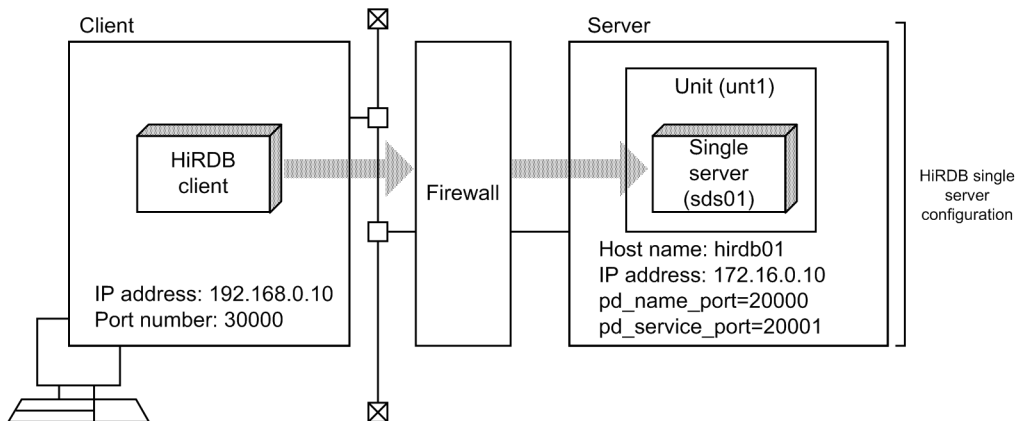
22.3.1 When a firewall is installed on the HiRDB single server configuration side

Here, a firewall is installed on the HiRDB single server configuration side as shown in the figure, with the firewall settings as follows.

Firewall settings

- Direction: Receive
- IP address: 172.16.0.10
- Port numbers: 20000, 20001

Figure 22–8: Network configuration example with a firewall installed on the HiRDB single server configuration side



In this configuration, the settings for the server and client machines are as listed below. When you install the firewall, you must set one of the following operands.

- `pd_service_port` operand
- `pd_scd_port` operand
- `-s` option of the `pdunit` operand

If only a firewall is installed, there is no need to specify the client environment definition (`PDSERVICEPORT` operand).

Server machine settings

- System common definition file


```
set pd_name_port= 20000
set pd_service_port= 20001
pdunit -x hirdb01 -u unt1
pdstart -t SDS -s sds01 -u unt1
```

Client machine settings

- Client environment definition


```
PDHOST hirdb01
PDNAMEPORT 20000
PDCLTRCVPORT 30000#
```

- `hosts` file
172.16.0.10 hirdb01

#: Specify this when there is a firewall on the client side.

22.3.2 When a firewall and NAT are installed on the HiRDB single server configuration side

Here, a firewall and NAT are installed on the HiRDB single server configuration side as shown in the figure, with the two set as follows.

Firewall settings

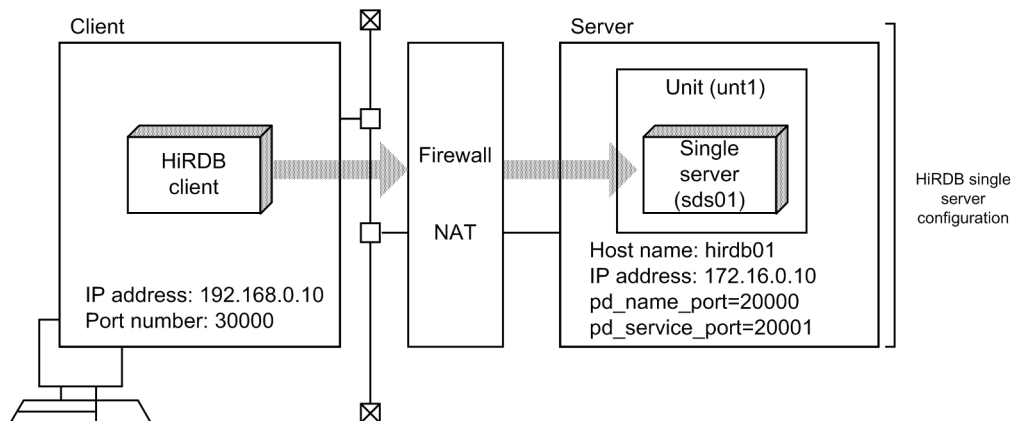
- Direction: Receive
- IP address: 172.16.0.10
- Port numbers: 20000, 20001

NAT address translation

128.1.1.1 \longleftrightarrow 172.16.0.10

Note that HiRDB does not support a function that converts global and local IP addresses as a pair, such as NAPT (IP masquerade). It supports only 1-to-1 conversion.

Figure 22–9: Network configuration example with a firewall and NAT installed on the HiRDB single server configuration side



In this configuration, configure the high-speed connection facility. The settings for the server and client machines are as follows:

Server machine settings

- System common definition file
set pd_name_port= 20000
set pd_service_port= 20001
pdunit -x hirdb01 -u unt1
pdstart -t SDS -s sds01 -u unt1

Client machine settings

- Client environment definition
PDHOST hirdb01
PDNAMEPORT 20000
PDSERVICEGRP sds01
PDSERVICEPORT 20001
PDSRVTYPE WS#1
PDCLTRCVPORT 30000#2

- `hosts` file
128.1.1.1 hirdb01

#1: To operate a HiRDB server on Linux, specify PC.

#2: Specify this when there is a firewall on the client side.

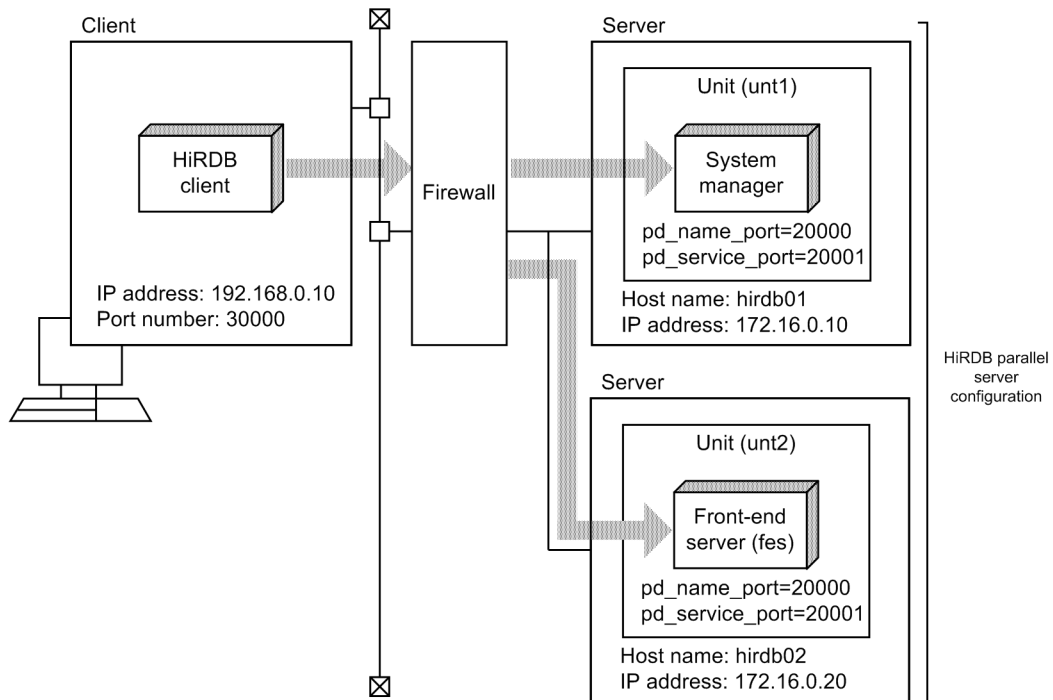
22.3.3 When a firewall is installed on the HiRDB parallel server configuration side

Here, a firewall is installed on the HiRDB parallel server configuration side as shown in the figure, with the firewall settings as follows.

Firewall settings

- Direction: Receive
- IP addresses: 172.16.0.10, 172.16.0.20
- Port numbers: 20000, 20001

Figure 22–10: Network configuration example with a firewall installed on the HiRDB parallel server configuration side



In this configuration, the settings for the server and client machines are as listed below. When you install the firewall, you must set one of the following operands.

- `pd_service_port` operand
- `pd_scd_port` operand
- `-s` option of the `pdunit` operand

If only a firewall is installed, there is no need to specify the client environment definition (`PDSERVICEPORT` operand).

Server machine settings

- System common definition file
set `pd_name_port = 20000`

```

set pd_service_port = 20001
pdunit -x hirdb01 -u unt1
pdunit -x hirdb02 -u unt2
pdstart -t MGR -u unt1
pdstart -t FES -s fes -u unt2

```

Client machine settings

- Client environment definition

```

PDHOST hirdb01
PDNAMEPORT 20000
PDCLTRCVPORT 30000#

```

- hosts file

```

172.16.0.10 hirdb01
172.16.0.20 hirdb02

```

#: Specify this when there is a firewall on the client side.

22.3.4 When a firewall and NAT are installed on the HiRDB parallel server configuration side

Here, a firewall and NAT are installed on the HiRDB parallel server configuration side as shown in the figure, with the two set as follows.

Firewall settings

- Direction: Receive
- IP addresses: 172.16.0.10, 172.16.0.20
- Port numbers: 20000, 20001

NAT address translation

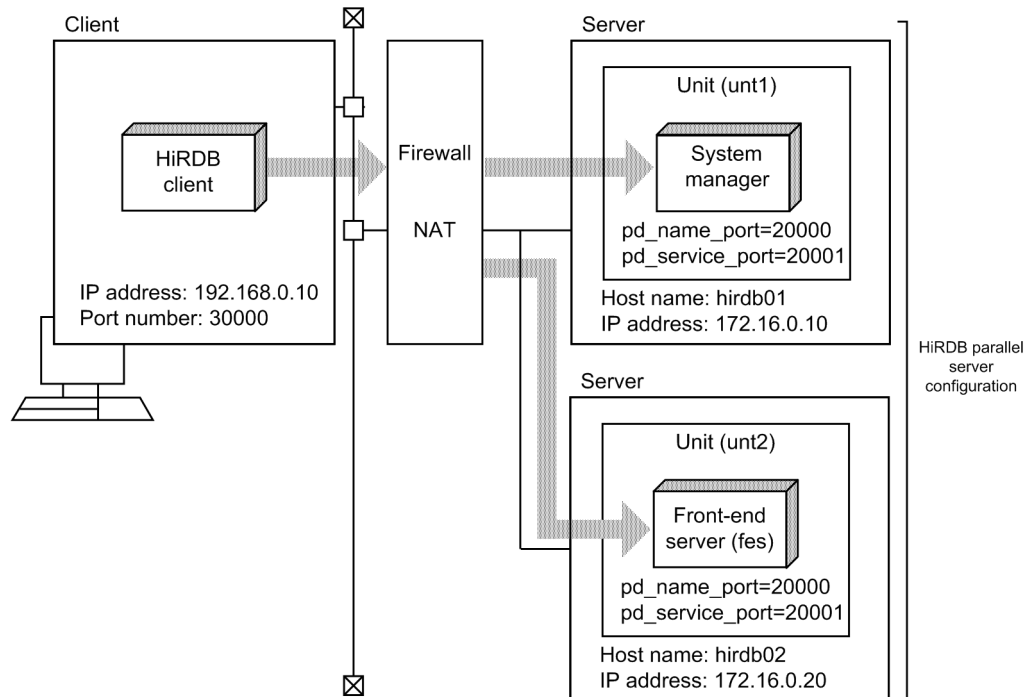
```

128.1.1.1 ← → 172.16.0.10
128.1.1.2 ← → 172.16.0.20

```

Note that HiRDB does not support a function that converts global and local IP addresses as a pair, such as NAPT (IP masquerade). It supports only 1-to-1 conversion.

Figure 22–11: Network configuration example with a firewall and NAT installed on the HiRDB parallel server configuration side



In this configuration, configure the high-speed connection facility. The settings for the server and client machines are as follows:

Server machine settings

- System common definition file


```
set pd_name_port = 20000
set pd_service_port = 20001
pdunit -x hirdb01 -u unt1
pdunit -x hirdb02 -u unt2
pdstart -t MGR -u unt1
pdstart -t FES -s fes -u unt2
```

Client machine settings

- Client environment definition


```
PDHOST hirdb01
PDNAMEPORT 20000
PDSERVICEGRP fes
PDSERVICEPORT 20001
PDFESHOST hirdb02
PDSRVTYPE WS#1
PDCLTRCVPORT 30000#2
```
- hosts file


```
128.1.1.1 hirdb01
128.1.1.2 hirdb02
```

#1: To operate a HiRDB server on Linux, specify PC.

#2: Specify this when there is a firewall on the client side.

22.4 Port numbers used by HiRDB

With HiRDB communication processing, if the `pd_registered_port` operand is not specified, the communication port numbers allocated automatically by the operating system are used. Increase the number of communication ports to be used in accordance with the value of the `pd_max_users` operand and increases in the number of back-end servers. If there are insufficient port numbers, processing can be interrupted or the communication processing of other programs can be influenced.

You can use the `pd_registered_port` operand to specify a range of port numbers for HiRDB to use for communication processing (reserved port facility). For details about the HiRDB reserved port facility, see 22.6 *HiRDB reserved port facility*.

22.4.1 Estimating the number of ports that a unit will use

The following shows the target number of ports that a HiRDB unit will use:

(1) HiRDB single server configuration

Formula

$$\text{value of } pd_max_users \times 4 + 1,000$$

(2) HiRDB parallel server configuration

With a multi-front-end server configuration, decide the values for f and F for each front-end server, which will become the total number of ports. Use the following formula to calculate the target number of port numbers for each front-end server.

Formula

$$\begin{aligned} &\{b \times [k \times (B + F) + 1] \\ &+ f \times (k \times B + D + 2) + d \times (F + 1)\} \\ &\times \text{value of } pd_max_users + 1,000 \end{aligned}$$

b : Number of back-end servers inside the unit

B : Number of back-end servers outside the unit

f : Number of front-end servers inside the unit

Each front-end server counts as either 1 or 0:

If the front-end server is inside the unit: 1

If the front-end server is outside the unit: 0

F : Number of front-end servers outside the unit

Each front-end server counts as either 1 or 0:

If the front-end server is inside the unit: 1

If the front-end server is outside the unit: 0

d : Number of dictionary servers inside the unit

D : Number of dictionary servers outside the unit

k : $2 \div 3$

22.4.2 Notes

- Depending on the SQL statement that is executed, more ports will be required than the calculated target value.

- If the number of ports allocated automatically by the operating system is insufficient, respecify the number in the `pd_registered_port` operand.
- When HiRDB releases port numbers, the OS does not necessarily release the numbers immediately (`TIME_WAIT` status). Therefore, the system will sometimes temporarily use a larger number of port numbers than the target number. For details about how to avoid a shortage of ports caused by the `TIME_WAIT` status, see *22.4.4 Ways to avoid a shortage of ports*.
- When the system configuration includes numerous back-end servers within the system, the calculated value might exceed the operating system's upper limit for port numbers or the port number specified in `pd_registered_port`. For details about the corrective action to take in this situation, see *9.5 Considerations that apply to building a system with many units or servers*.

22.4.3 Calculation examples

Example 1

- A 1-unit configuration of a HiRDB parallel server configuration (with a FES, a DS, and 5 BESSs), where `pd_max_users=1000`

$$\{5 \times [2 \div 3 \times (0 + 0) + 1]$$

$$+ 1 \times (2 \div 3 \times 0 + 0 + 2) + 1 \times (0 + 1)\}$$

$$\times 1,000 + 1,000 = 9,000$$
 The target number of ports is 9,000.

Example 2

- A parallel unit configuration (unit 1: a FES, a DS, and 2BESSs; unit 2: 3 BESSs), where `pd_max_users=1000`
 Unit 1:

$$\{2 \times [2 \div 3 \times (3 + 0) + 1]$$

$$+ 1 \times (2 \div 3 \times 3 + 0 + 2) + 1 \times (0 + 1)\}$$

$$\times 1,000 + 1,000 = 12,000$$
 Unit 2:

$$\{3 \times [2 \div 3 \times (2 + 1) + 1]$$

$$+ 0 \times (2 \div 3 \times 2 + 1 + 2) + 0 (1 + 1)\}$$

$$\times 1,000 + 1,000 = 10,000$$
 The target number of ports is 12,000 for unit 1 and 10,000 for unit 2.

22.4.4 Ways to avoid a shortage of ports

When the following types of operations are performed, a large number of TCP ports might go into the `TIME_WAIT` status, TCP ports in the system as a whole might run short and cause transactions to generate errors, or HiRDB might terminate abnormally.

- Utilities or commands are executed consecutively (in the case of a HiRDB parallel server configuration).
- A UAP makes numerous concurrent connection requests to HiRDB, and the UAP repeatedly executes any of the following types of transactions (in the case of a HiRDB parallel server configuration).
 - Transactions that execute SQL processing on multiple tables
 - Transactions that use a floating server
 - Update transactions that use a recovery-unnecessary front-end server

For these sorts of operations, use the settings shown below to avoid TCP port shortages.

The OS parameters and setting methods described here will differ depending on the OS and kernel versions you are using. Consult the documentation of the OS you are using, and set the corresponding OS parameters to the values that are given as guidelines in this discussion.

(1) Settings to avoid a shortage of TCP ports

You can avoid a shortage of TCP ports by using the `pd_registered_port` operand to expand the range of port numbers that HiRDB uses for communication processing (the HiRDB reserved port facility). For details about the HiRDB reserved port facility, see *22.6 HiRDB reserved port facility*.

(2) Settings that shorten the time a TCP port spends in TIME_WAIT status (AIX and HP-UX only)

To shorten the amount of time a TCP port spends in the `TIME_WAIT` status, specify the following OS parameter.

- **For AIX**
 - OS parameter: `tcp_timewait`
 - Specification guideline: 1
 - Example of parameter setting command: the `no` command
- **For HP-UX**
 - OS parameter: `tcp_time_wait_interval`
 - Specification guideline: 15000
 - Example of parameter setting command: the `ndd` command

(3) Settings that expand the OS's auto-allocated port range for TCP ports

To expand the OS's auto-allocated port range for TCP ports, specify the following OS parameter.

- **For Linux**
 - OS parameter: `ip_local_port_range`
 - Specification guideline: Expand the range of the OS's auto-allocated ports, but without allowing that range to encroach upon the range of reserved ports in the server machine.
 - Option settings file example: `/proc/sys/net/ipv4/ip_local_port_range`
- **For AIX**
 - OS parameters: `tcp_ephemeral_high` and `tcp_ephemeral_low`
 - Specification guideline: Expand the range of the OS's auto-allocated ports, but without allowing that range to encroach upon the range of reserved ports in the server machine.
 - Example of parameter setting command: The `no` command
- **For HP-UX**
 - OS parameters: `tcp_largest_anon_port` and `tcp_smallest_anon_port`
 - Specification guideline: Expand the range of the OS's auto-allocated ports, but without allowing that range to encroach upon the range of reserved ports in the server machine.
 - Example of parameter setting command: the `ndd` command

22.5 Port numbers specified in HiRDB

22.5.1 List of port numbers specified in HiRDB

The following table lists the operands that specify port numbers.

Table 22–2: List of port numbers specified in HiRDB

Setting location	Environmental variable or operand to specify	Description
Client environment definition	PDNAMEPORT	HiRDB port number
	HiRDB_PDNAMEPORT	
	PDSERVICEPORT	Port number for high-speed connection
	PDFESHOST	Port number of unit that has a front-end server
	PDCLTRCVPORT	Client reception port number
	PDASTPORT	HiRDB Control Manager - Agent port number
System common definition	pd_name_port	HiRDB port number
	pdunit -p	
	pd_service_port	Scheduler process port number
	pd_scd_port	
	pdunit -s	
	pd_trn_port	Transaction server process port number
	pdunit -t	
	pd_mlg_port	Message log server process port number
	pdunit -m	
	pd_alv_port	Unit monitoring process port number
	pdunit -a	
	pd_registered_port	Port number that is used by HiRDB reserved port facility
Unit control information definition	pd_service_port	Scheduler process port number
	pd_registered_port	Port number that is used by HiRDB reserved port facility

22.5.2 Specifying port numbers

This section describes the different ways to specify port numbers.

(1) Client environment definition

See *Client environment definition (setting environment variables)* in the *HiRDB Version 9 UAP Development Guide*.

(2) System common definition and unit control information definition

This subsection describes the port numbers that are used, depending on whether system common definition and unit control information definition operands are specified.

To lock the port number specified in an operand, see the *Port number used* column in the table, and specify the applicable operand in a combination that uses that port number. For example, in (a) *HiRDB port numbers*, to lock the port number specified in the `pd_name_port` operand for a HiRDB parallel server configuration, do not specify `pdunit -p`.

(a) HiRDB port numbers

HiRDB port numbers are specified in the `pd_name_port` operand and in the `pdunit` operand's `-p` option. This subsection describes the port numbers that are used when these operands are specified.

For a HiRDB single server configuration

pd_name_port	pdunit -p	Port number used
Specified	Specified	Port specified in <code>pd_name_port</code>
	Not specified	
Not specified	Specified	20,000
	Not specified	

The following table shows the port numbers that are used when the utility special unit is installed.

Location specified		Port number used			
System common definition		At single server		At utility special unit	
pd_name_port	pdunit -p	Unit that has a single server	Utility special unit	Unit that has a single server	Utility special unit
Specified	Specified	Port specified in <code>pd_name_port</code>	Port specified in <code>pdunit -p</code>	Port specified in <code>pdunit -p</code>	Port specified in <code>pd_name_port</code>
	Not specified	Port specified in <code>pd_name_port</code>			
Not specified	Specified	20,000	Port specified in <code>pdunit -p</code>	Port specified in <code>pdunit -p</code>	20,000
	Not specified	20,000	20,000	20,000	20,000

For a HiRDB parallel server configuration

pd_name_port	pdunit -p	Port number used
Specified	Specified	Port specified in <code>pdunit -p</code>
	Not specified	Port specified in <code>pd_name_port</code>
Not specified	Specified	Port specified in <code>pdunit -p</code>
	Not specified	20,000

(b) Scheduler process port numbers

Specify scheduler process port numbers in the following operands.

- `pd_service_port` operand
- `pd_scd_port` operand
- `-s` option of the `pdunit` operand

This subsection describes the port numbers that are used, depending on whether these operands are specified.

Location specified				Port number used
System common definition			Unit control information definition	
pd_service_port	pd_scd_port	pdunit -s	pd_service_port	
Specified	Specified	Specified	Specified	Port specified in <code>pdunit -s</code>
			Not specified	
		Not specified	Specified	Port specified in <code>pd_scd_port</code>
			Not specified	
	Not specified	Specified	Specified	Port specified in <code>pdunit -s</code>
			Not specified	
		Not specified	Specified	Port specified in <code>pd_service_port</code> of unit control information definition
			Not specified	
Not specified	Specified	Specified	Specified	Port specified in <code>pdunit -s</code>
			Not specified	
		Not specified	Specified	Port specified in <code>pd_scd_port</code>
			Not specified	
	Not specified	Specified	Specified	Port specified in <code>pdunit -s</code>
			Not specified	
		Not specified	Specified	Port specified in <code>pd_service_port</code> of unit control information definition
			Not specified	

#

If the `pd_registered_port` operand is specified in the system common definition or the unit control information definition, a port number in the range specified in the `pd_registered_port` operand is used. If the `pd_registered_port` operand is not specified, a port number is allocated automatically by the OS.

(c) Port numbers of transaction server processes, message log server processes, and unit monitoring processes

Specify the port numbers of transaction server processes, message log server processes, and unit monitoring processes in the operands shown below.

- Transaction server processes
`pd_trn_port` operand and `-t` option of `pdunit` operand
- Message log server processes
`pd_mlg_port` operand and `-m` option of `pdunit` operand
- Unit monitoring processes
`pd_alv_port` operand and `-a` option of `pdunit` operand

This section describes the port numbers that are used, depending on whether these operands are specified.

pd_trn_port, pd_mlg_port, or pd_alv_port	pdunit -t, -m or -a	Port number used
Specified	Specified	Port specified in pdunit -t, -m, or -a
	Not specified	Port specified in pd_trn_port, pd_mlg_port, or pd_alv_port
Not specified	Specified	Port specified in pdunit -t, -m, or -a
	Not specified	#

#

If the `pd_registered_port` operand is specified in the system common definition or unit control information definition, a port number in the range specified in the `pd_registered_port` operand is used. If the `pd_registered_port` operand is not specified, a port number is allocated automatically by the OS.

(d) Port numbers used by the HiRDB reserved port facility

Port numbers used by the HiRDB reserved port facility are specified in the `pd_registered_port` operand. This section describes the port numbers that are used, depending on whether these operands are specified.

Location specified		Port number used
System common definition	Unit control information definition	
pd_registered_port	pd_registered_port	
Specified	Specified	Port specified in the unit control information definition
	Not specified	Port specified in system common definition
Not specified	Specified	Port specified in the unit control information definition
	Not specified	--

Legend:

--: N/A

22.5.3 Notes on port number duplication

The following provides notes on duplication of the port numbers specified in HiRDB system definitions (the *System common definition* and *Unit control information definition* sections of *Table 22-2 List of port numbers specified in HiRDB*).

- In a single unit, specify different port numbers in each operand.
- For a HiRDB parallel server configuration, if you start multiple units on a single server machine, specify different port numbers in each operand of the respective units.
- If you start multiple units on the same server machine, specify different port numbers in each operand of the respective HiRDBs.
- If the same server machine has a HiRDB client and a HiRDB server, specify port numbers different from the client environment definition `PDCLTRCVPORT` in each operand.
- Specify port numbers that fulfill the following conditions.
 - They are different from the port numbers used by other program products.
 - They are different from the port numbers registered in the `/etc/services` file (for an NIS or DNS environment, the locations they respectively define)

- They are not included in the range of port numbers auto-allocated by the OS[#]
#: The range of port numbers auto-allocated by the OS differs from OS to OS.
- When no HiRDB port number is specified, 20000 is assumed for the HiRDB port number. For this reason, do not specify 20000 in an operand.

22.6 HiRDB reserved port facility

The HiRDB reserved port facility uses the `pd_registered_port` operand to specify a range of port numbers that can be used for communication. This facility prevents the following:

- A program other than HiRDB communicates with a server using several port numbers allocated automatically by the OS, but processing is interrupted because there are no available port numbers
 - HiRDB uses a large number of communication port numbers, which affects the communication processing of other programs
- Allocated ports that are released by HiRDB cannot be used for a period of time. Therefore, if a large number of ports are in use, it is possible that for a brief period there will be no available port numbers.

If HiRDB is using a large number of ports, see *22.4.4 Ways to avoid a shortage of ports*.

For details about definition examples and notes about using the HiRDB reserved port facility, see the explanation for the `pd_registered_port` operand in the manual *HiRDB Version 9 System Definition*.

22.6.1 Estimation of the HiRDB reserved port facility

(1) Estimating from statistical information

The number of communication port numbers that HiRDB will use can be estimated from statistical information. Using the statistical analysis utility's *Statistical information related to system moving*, you can calculate the number of HiRDB communication ports from the following statistical information (for details, see the manual *HiRDB Version 9 Command Reference*):

- The number of HiRDB reserved ports in use
- When there are excess HiRDB reserved ports, the number of ports allocated automatically by the OS that are in use

Formula

Number of HiRDB reserved ports in use + number of ports allocated automatically by the OS that are in use when there are excess HiRDB reserved ports + 100[#]

[#]: Added as a reserve.

(2) Estimation of the recommended value

The recommended value is determined from the formula in *22.4.1 Estimating the number of ports that a unit will use*. This is the target value. During operation, use the value from *(1) Estimating from statistical information*.

Appendixes

A. HiRDB Maximum and Minimum Values

A.1 Maximum and minimum values for the system configuration

The following table lists maximum and minimum values for the HiRDB system configuration.

Table A–1: Maximum and minimum values for the HiRDB system configuration

Item	Minimum value	Maximum value	Units
Number of servers that can be created per unit	1	34	Count
Number of single servers	1	1	Count
Number of system managers	1	1	Count
Number of front-end servers	1	1,024	Count
Number of dictionary servers	1	1	Count
Number of back-end servers	1	16,382	Count
Total number of RDAREAs	3	8,388,592	Count
Number of master directory RDAREAs	1	1	Count
Number of data directory RDAREAs	1	1	Count
Number of data dictionary RDAREAs	1	59	Count
Number of data dictionary RDAREAs that store database state analyzed tables and database management tables	1	1	Count
Number of user RDAREAs	0	8,388,589	Count
Number of data dictionary LOB RDAREAs	0	2	Count
Number of user LOB RDAREAs	0	8,388,325	Count
Number of registry RDAREAs	0	1	Count
Number of registry LOB RDAREAs	0	1	Count
Number of list RDAREAs	0	8,388,588	Count
Number of HiRDB files per RDAREA	1	16	Count
Number of base tables and sequence generators per RDAREA	0	500	Count
Number of indexes per RDAREA	0	500	Count
Number of lists per RDAREA	0	50,000	Count
Total number of HiRDB files	1	134,217,728	Count
Number of concurrently accessible tables	4	32,000	Count
Maximum number of concurrent connections	1	For a HiRDB single server configuration: 3,000 For a HiRDB parallel server configuration: 2,000 ^{#1}	Count
Maximum number of users that can be created by HiRDB ^{#2}	1	Unlimited	Count

Item				Minimum value	Maximum value	Units
Number of users who can own the same list				0	32,767	Count
Number of lists created per user				0	32,767	Count
Number of work tables created per transaction				0	255	Count
Number of global buffers per server ^{#3}				1	2,000,000	Count
HiRDB file system area	HP-UX	Large file not used	Regular file	1	2,047	MB
			Character special file	1	2,047	MB
		Large file used	Regular file	1	1,048,575	MB
			Character special file	1	1,048,575	MB
	Solaris	Large file not used	Regular file	1	2,047	MB
			Character special file	1	2,047	MB
		Large file used	Regular file	1	1,048,575	MB
			Character special file	1	1,048,575	MB
	AIX	Large file not used	Regular file	1	2,047	MB
			Character special file	1	2,047	MB
		Large file used	Regular file (JFS)	1	65,411	MB
			Regular file (JFS2)	1	1,048,575	MB
			Character special file	1	1,048,575	MB
	Linux	Large file not used	Regular file	1	2,047	MB
			Character special file	1	2,047	MB
		Large file used	Regular file	1	1,048,575	MB
			Character special file	1	1,048,575	MB

#1: For a configuration with multiple front-end servers, the maximum for *number of front-end servers* × *value of pd_max_users* is 2,000.

#2: The value depends on the size of data dictionary RDAREAs because one row of the data dictionary table (SQL_USERS) is used per user.

#3: However, for the entire system, the maximum is 2,147,483,647.

A.2 Maximum and minimum values for databases

The following table lists maximum and minimum values for databases.

Table A–2: Maximum and minimum values for databases

Item		Minimum value	Maximum value	Units
Length of character data (defined length)	CHAR	1	30,000	Characters (Bytes)
	VARCHAR	1	32,000	Characters (Bytes)
Length of national character data (defined length)	NCHAR	1	15,000	Characters
	NVARCHAR	1	16,000	Characters

A. HiRDB Maximum and Minimum Values

Item		Minimum value	Maximum value	Units
Length of mixed character data (defined length)	MCHAR	1	30,000	Bytes
	MVARCHAR	1	32,000	Bytes
Precision of packed decimal	DECIMAL or NUMERIC	1	38	Digits
Decimal places for packed decimal	DECIMAL or NUMERIC	0	38	Digits
Seconds precision for timestamp data	TIMESTAMP	0	6	Digits
Length of BLOB data		0	2,147,483,647	Bytes
BINARY data length (defined length)		1	2,147,483,647	Bytes
Number of columns in a table		1	30,000	Columns
Number of indexes in a table		0	255	Count
Number of index component columns		1	64	Columns
Number of cluster key component columns		1	64	Columns
Number of RDAREAs for storing table partitions		1	4,096	Count
Number of BESSs where partitioned tables are placed		1	4,096	Count
Total number of literals specified for storage conditions when a row-partitioned table is defined (if storage condition is omitted, 1 is assumed)		1	15,000	Count
Number of tables based on view tables		1	64	Count
Number of columns in a view table		1	30,000	Columns
Number of primary key component columns		1	64	Columns
Number of foreign key component columns		1	64	Columns
Number of foreign keys per table		0	255	Count
Number of foreign keys that reference a single primary key		0	255	Count
Number of check constraints that can be specified per table		0	254	Count
Total number of Boolean operators (AND and OR) and check constraint definitions that can be defined per table (excluding the Boolean operators AND and OR in WHEN search conditions in CASE expressions)		0	254	Count
Identifier length (Applicable to table identifier, column name, data type identifier, index type identifier, attribute name, routine identifier, correlation name, index identifier, cursor name, SQL statement identifier, RDAREA name, embedded variable name, indicator variable name, password, constraint name, condition name, SQL variable name, query name, trigger identifier, statement label, loop variable name, host identifier, list name, SQL parameter name)		1	30	Bytes
Row length of FIX table		1	30,000	Bytes
Number of SQL parameters in a procedure		0	30,000	Count
Number of repetition columns		2	30,000	Count
Number of index information files created per server	HP-UX	1	8,092 ^{#1}	Count
	Solaris	1	1,948 ^{#1}	Count

Item		Minimum value	Maximum value	Units
	Solaris(64-bit mode)	1	8,092 ^{#1}	Count
	AIX	1	8,092 ^{#1}	Count
	Linux	1	8,092 ^{#1}	Count
Number of processed rows that can be displayed in messages by the following utilities: <ul style="list-style-type: none"> • pdload • pdrorg • pdrbal 		0	4,294,967,295 ^{#2}	Count

#1: The maximum value depends on specifications such as the `pd_max_open_fds` operand value, whether plug-ins are used, and the number of RDAREAs in the server. The table value is the maximum value specified in the `pd_max_open_fds` operand. If plug-in index delayed batch creation uses a HiRDB file system area, the maximum value is 4,096.

#2: When the number of processed data items exceeds 4,294,967,295, the displayed row count is reset to 0 and the count starts again from 1.

A.3 Maximum and minimum values for HiRDB file names

The following table lists maximum and minimum values for HiRDB file names.

Table A–3: Maximum and minimum values for HiRDB file names (in characters)

HiRDB file type	HiRDB file name length		Maximum length of path name [#]
	Minimum	Maximum	
System status file	1	30	167
Server status file	1	30	167
System log file	1	30	167
Synchronization point dump file	1	30	167
Unload log file	1	30	167
Audit trail file	16	--	167
Work table file	25	--	167
RDAREA structure file	1	30	167
Backup file	1	30	167
Unload data file	1	30	167
Index information file	30	--	167
Differential backup management file	1	30	167

Legend:

--: Not applicable; the length must be the value shown in the *Minimum* column.

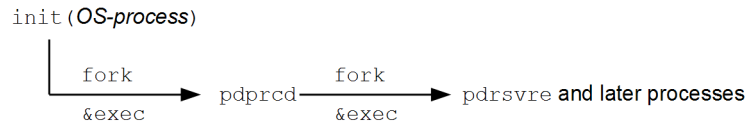
#: Structure of a path name is *HiRDB-file-system-area / HiRDB-file*.

B. Processes Started by HiRDB

This appendix lists and describes processes that are started by HiRDB.

B.1 Processes started by a HiRDB single server configuration

The organization of processes started by HiRDB is shown below.



The following table lists processes started by a HiRDB single server configuration.

Table B–1: Processes started by a HiRDB single server configuration (system server)

Process name		Description	Number of processes	Server name
Notation in manual	Process name			
Process server process	pdprcd	Manages HiRDB related processes	1	_prc
Post-processing process	pdrsvre	Performs cleanup processing after abnormal termination of a process	<p>If <code>fixed</code> is specified in <code>pd_process_terminator</code>: value of <code>pd_process_terminator_max</code></p> <p>If <code>resident</code> is specified in <code>pd_process_terminator</code>: from 3 to the number of processes defined by HiRDB</p> <p>If <code>nonresident</code> is specified in <code>pd_process_terminator</code>: from 0 to the number of processes that terminated abnormally</p>	_admrsvr
HiRDB start process for server mode system switchover	pdstart2d	<p>Controls start of HiRDB processes linked to cluster software</p> <ul style="list-style-type: none"> Standby system switchover that does not perform user server hot standby User server hot standby Rapid system switchover 	1 (none when online)	_pdstrt2
Message log server process	pdmlgd	Controls message output	1	_mlg
System manager process	pdrdmd	<p>Unit start/stop control, connected user management</p> <p>(This might also be notated as a name server or a node manager.)</p>	1	_rdm
Status server process	pdstd	I/O control of status file for units	1	_sts0

Process name		Description	Number of processes	Server name
Notation in manual	Process name			
Scheduler process	pdscdd	Allocates transactions to single server processes (This might also be notated as a lock server.)	1	_scd
Transaction server process	pdtrnd	Controls transactions	1	_trn
Transaction recovery process	pdtrnrvd	Controls transaction completion/recovery	From 1 to the number of crashed pdsds ^{#1}	_trnrcv
Audit trail management server process	pdaudd	Manages audit trails	If pd_aud_file_name is specified, 1; otherwise, 0	_aud
Audit trail automatic load process	pdauld	Start control of pdload for automatic data loading	If pd_aud_file_name is specified and Y is specified in pd_aud_auto_loading, 1; otherwise 0	_auld
XDS log output process ^{#8}	pdprctee	Controls output of XDS log	If the pdxds operand is specified, 1; otherwise, 0	--
Command execution process	pdcmdd	Controls execution of update log reflection command	If the pdxds operand is specified, 1; otherwise, 0	_cmdd
Cluster synchronization monitoring process	pdxc1	Synchronization monitoring of running XDS and standby XDS	If the pdxds operand is specified, 1; otherwise, 0	_xc1
Troubleshooting information acquisition process	pdprfd	Controls the troubleshooting function	1	--
Log server process	pdlogd	Controls system log acquisition and log-related processes	1	_logN
Deferred write process	pd_buf_dfw	Background writes to DB storage disk	1	1dfwN
Asynchronous READ process	pd_ios_ard	Asynchronous READ facility	Value of pd_max_ard_process	1ardN
Process for parallel writes in deferred write processing	pd_buf_awt	Facility for parallel writes in deferred write processing	Value of pd_dfw_awt_process	1awtN
REDO process	pd_rcv_rd	Roll forward of database at full rerun	MIN(number of connected disks, value of pd_max_recover_process) Number of connected disks: Number of character special files that define RDAREAs If Y is specified in pd_rdarea_open_attribute_use, the value of pd_max_recover_process ^{#2}	2rrnM
Log swap process	pdlogswd	Manages allocation, release and I/O of system log-related files and acquires syncpoint dump	1	_logNs

B. Processes Started by HiRDB

Process name		Description	Number of processes	Server name
Notation in manual	Process name			
Deadlock monitoring process	pdckmnd	Detects deadlock when lock processing is distributed	If 2 or more is specified for <code>pd_lck_pool_partition</code> and <code>Y</code> is specified for <code>pd_lck_deadlock_check</code> , 1; otherwise, 0	<code>_lckmnN</code>

Table B–2: Processes started by a HiRDB single server configuration (user server)

Process name		Explanation	Number of processes	Server name
Notation in manual	Process name			
Single server process	pdsds	SQL processing	<i>Value of <code>pd_max_users</code> + value of <code>pd_max_reflect_process_count</code>^{#3}</i>	Server name ^{#4}

Table B–3: Processes started by a HiRDB single server configuration (XDS server)

Process name		Explanation	Number of processes	Server name
Notation in manual	Process name			
XDS process	pdxds	SQL processing, manages memory database	If the <code>pdxds</code> operand is specified, 1; otherwise, 0 ^{#9}	Server name ^{#10}

Table B–4: Processes started by a HiRDB single server configuration (utility server)

Process name ^{#5}		Explanation	Number of processes	Server name
Notation in manual	Process name			
pdinit control process	pdinitd	Initialization utility execution process	1	0minit0
pdcopy backup output process	pdcopyb	Backup file output	<i>Degrees of duplication × number of concurrent command executions</i>	0bcpy0 ^{#6}
pdcopy database read process	pdcopyr	Database reads	Number of concurrent command executions	0rcopy0
pdrstr backup read process	pdrstrb	Backup file reads	Number of concurrent command executions	0brstr0
pdrstr master directory RDAREA read process	pdrstrm	Reads master directory RDAREA	Number of concurrent command executions	0mrstr0
pdrstr unload log read process	pdrstrl	Reads unload log file	Number of concurrent command executions	0lrstr0
pdrstr database write process	pdrstrr	Database writes	Number of concurrent command executions	0brstr0
pdrstr master directory RDAREA write process	pdrstrw	Writes to master directory RDAREA	Number of concurrent command executions	0wrstr0
pdload control process	pdloadm	Controls data loads	Number of concurrent <code>pdload</code> command executions	0mload0

Process name ^{#5}		Explanation	Number of processes	Server name
Notation in manual	Process name			
pdorg control process	pdorgm	Controls database reorganization (unloading, reloading, index reorganization/re-creation, free page release, and making global buffers resident)	Number of concurrent pdorg, pdreclaim, and pdpgbfon command executions	0mrorg0
pdgetcst control process	pdgcstm	Collects optimization information	Number of concurrent pdgetcst command executions	0mgcst0
pddbstd control process	pddbstd1	Controls database condition analysis	Number of concurrent pddbstd command executions	0mdbstd0
pdexp control process	pdexpm	Controls dictionary export/import	1	0mexp0
pdplgexe control process	pdplgexm	Plug-in utility execution control	Number of concurrent executions of commands provided by plug-ins	0mplge0
pdorend control process	pdorendm	Online reorganization Reflection control	1	0more0
pdorend application process	pdorendl	Online reorganization Reflection processing	Degrees of duplication (number of specifications of the pdorend command's -m option)	0lore?0 ^{#7}

Legend

--: N/A

Notes

- The *xxxN* in the server name increases by 1, 2, ..., up to the maximum number of servers in the unit, based on the number of user servers.
- The *xxxM* in the server name increases by 2 to 11 based on the definitions.
- The server name is used in message output, command information output, and so on.

#1

If two or more pdsds have crashed, increases to the number of crashes. As the crashed pdsds transactions complete, the number of processes decreases, returning to 1 when there are no more transactions subject to recovery. The upper limit value is as follows:

$$\text{MIN}(\downarrow \text{value of } pd_trn_rcvmsg_store_buflen \div 72 \downarrow, (\text{value of } pd_max_users + \text{value of } pd_max_reflect_process_count) \times 2 + 7)$$

#2

The REDO process starts when HiRDB starts, and stops when startup is complete.

#3

For utility servers, the number of processes is 0. When it exceeds 2,000 after calculation, everything up to 2,000 is processes that are started. When *pd_process_count* is specified, the number of startup process is the amount of the specification. When the number of access requests exceeds *pd_process_count*, concurrent processing occurs, the number of processes that start is the number of concurrent accesses, up to *value of pd_max_users + value of pd_max_reflect_process_count*.

#4

This is the server name specified by the -s option of the system common definition's *pdstart* operand.

#5

The process only starts up when the corresponding command is executing. The process stops when the command terminates.

#6

? is increased 0, 1, ..., *f* by the backup output process's degree of duplication (the number of specifications of the `-b` option in the control statement file specified by the `pdcopy -f` option).

#7

? is increased 1, 2, ..., 8 by the online reorganization reflection processing's degree of duplication (the number of specifications of the `pdorend` command's `-m` option).

#8

The XDS log output process starts if the `pdstart` command executes and then becomes resident. It does not stop even if the `pdstop` command is executed.

#9

If the `pdxdstart` command is executed, the process starts; it stops if the `pdxdstop` command is executed.

#10

This is the server name specified by the `-s` option of the system common definition's `pdxd` operand.

B.2 Processes started by a HiRDB parallel server configuration

The organization of processes started by HiRDB is shown below.

`init (OS-process)`



The processes started by a HiRDB parallel server configuration are listed in the following table.

Table B-5: Processes started by a HiRDB parallel server configuration (system server) (1/2)

Process name		Description	Number of processes	Server name	Does process start for each server?	
Notation in manual	Process name				MG R	Non - MG R
Process server process	pdprcd	Manages HiRDB related processes	1	<code>_prc</code>	Y	Y
Post-processing process	pdrsvre	Performs cleanup processing after abnormal termination of a process	<p>If fixed is specified in <code>pd_process_terminator:</code> value of <code>pd_process_terminator_max</code></p> <p>If resident is specified in <code>pd_process_terminator:</code> from 3 to the number of processes defined by HiRDB</p> <p>If nonresident is specified in <code>pd_process_terminator:</code> from 0 to the number of processes that terminated abnormally</p>	<code>_admrsvr</code>	Y	Y
HiRDB start process for server mode system switchover 1	pdstart2d	Controls start of HiRDB processes linked to cluster software	1 (none when online)	<code>_pdstrt2</code>	Y	Y

Process name		Description	Number of processes	Server name	Does process start for each server?	
Notation in manual	Process name				MGR	Non-MGR
		<ul style="list-style-type: none"> Standby system switchover that does not perform user server hot standby User server hot standby Rapid system switchover Running system of standby-less system switchover (1:1) 				
HiRDB start process for server mode system switchover 2	pdstart2a	Controls start of HiRDB processes linked to cluster software <ul style="list-style-type: none"> Standby system of standby-less system switchover (1:1) 	1 (none when online)	_pdst2a1	N	Y
HiRDB start process for server mode system switchover 3	pdsvstartd	Controls start of HiRDB processes linked to cluster software <ul style="list-style-type: none"> Standby system of standby-less system switchover (effects distributed) 	If <code>activeunits</code> is specified in <code>pd_ha_agent</code> , 1; otherwise, 0	_pdsvstartd	N	Y
XDS log output process ^{#14}	pdprctee	Controls output of XDS log	If the <code>pdxds</code> operand is specified for a BES within a unit, 1; otherwise, 0	--	Y	Y
Command execution process	pdcmdd	Controls execution of update log reflection command	If the <code>pdxds</code> operand is specified for a BES within a unit, 1; otherwise, 0	_cmdd	Y	Y
Cluster synchronization monitoring process	pdxc1	Synchronization monitoring of running XDS and standby XDS	If the <code>pdxds</code> operand is specified for a BES within a unit, 1; otherwise, 0	_xc1	Y	Y
Troubleshooting information acquisition process	pdprfd	Controls the troubleshooting function	1	--	Y	Y
Message log server process	pdmlgd	Controls message output (starts when <code>local</code> is specified for <code>pd_mlg_msg_log_unit</code>)	1	_mlg	Y	Y
System manager (MGR) process	pdrdmd	Controls unit start/stop and manages connected users (This might also be notated as a name server or a node manager.)	1	_rdm	Y	N

B. Processes Started by HiRDB

Process name		Description	Number of processes	Server name	Does process start for each server?	
Notation in manual	Process name				MGR	Non-MGR
Node manager (non-MGR unit) process	pdndmd	Controls unit start/stop and manages connected users (This might also be notated as a name server.)	1	_ndm	N	Y
Status server process	pdstd	Controls I/O of status files for units	1	_sts0	Y	Y
Scheduler process	pdsd	Allocates back-end server, dictionary server, and front-end server processes (This might also be notated as a lock server.)	1	_scd	Y	Y
Transaction server process	pdtrnd	Controls transactions	1	_trn	Y	Y
Transaction recovery process	pdtrnrvd	Controls transaction completion/recovery	For an FES: from 1 to the number of crashed pdfes ^{#1} For a DS: from 1 to the number of crashed pddic ^{#1} For a BES: from 1 to the number of crashed pdbes ^{#1}	_trnrcv	Y	Y
Audit trail management server process	pdaudd	Manages audit trails	If pd_aud_file_name is specified, 1; otherwise, 0	_aud_aud ^{#2}	Y	Y
Audit trail automatic load process	pdauld	Controls start of pdload for automatic data load	If Y is specified for pd_aud_auto_loading in keeping with the conditions of the audit trail management server process, 1; when N or nothing is specified, 0	_auld	Y	N
Unit monitoring process	pdrdma	Monitors whether the HiRDB unit is running	1 (when the number of units is 1, 0)	_rdmck	Y	N

Table B–6: Processes started by a HiRDB parallel server configuration (system server) (2/2)

Process name		Description	Number of processes	Server name	Does process start for each server?		
Notation in manual	Process name				BES	DS	FES
Log server process	pdlogd	Controls system log acquisition and log-related processes	1	_logN _logN ^{#2}	Y	Y	Y

Process name		Description	Number of processes	Server name	Does process start for each server?		
Notation in manual	Process name				BES	DS	FES
Deferred write process	pd_buf_dfw	Background write to DB storage disk	1	1dfwN	Y	Y	N
Asynchronous READ process	pd_ios_ard	Asynchronous READ facility	Value of <code>pd_max_ard_process</code>	1ardN	Y	Y	N
Parallel WRITE process for deferred write processing	pd_buf_awt	Facility for parallel writes in deferred write processing	Value of <code>pd_dfw_awt_process</code>	1awtN	Y	Y	N
REDO process	pd_rcv_rd	DB roll forward at full rerun	MIN(<i>number of connected disks, value of <code>pd_max_recover_process</code></i>) Number of connected disks: Number of character special files that define RDAREAs If the value of <code>pd_rdarea_open_attribute_use</code> is Y, the value of <code>pd_max_recover_process</code> ^{#3}	2rrnM	Y	Y	N
Log swap process	pdlogswd	Manages allocation, release and I/O of system log-related files and acquires syncpoint dump	1	_logsN _lozsN ^{#2}	Y	Y	Y
Deadlock monitoring process	pdlckmnd	Detects deadlock when lock processing is distributed	For a FES: If 2 or more is specified for <code>pd_fes_lck_pool_partition</code> and if Y is specified for <code>pd_lck_deadlock_check</code> , 1; otherwise, 0 For a DS or BES: If 2 or more is specified for <code>pd_lck_pool_partition</code> and if Y is specified for <code>pd_lck_deadlock_check</code> , 1; otherwise, 0	_lckmnN _lckmzN ^{#2}	Y	Y	Y

Table B–7: Processes started by a HiRDB parallel server configuration (user server)

Process name		Explanation	Number of processes	Server name	Does process start for each server?		
Notation in manual	Process name				BES	DS	FES
Back-end server process	pdbes	Access to database	MAX (<i>value of <code>pd_max_users</code>, value of <code>pd_max_bes_process</code></i>)	Server name ^{#4}	Y	N	N
Dictionary server process	pddic	Batch management of dictionary table	MAX (<i>value of <code>pd_max_users</code>, value of <code>pd_max_dic_process</code></i>)	Server name ^{#4}	N	Y	N

B. Processes Started by HiRDB

Process name		Explanation	Number of processes	Server name	Does process start for each server?		
Notation in manual	Process name				BES	DS	FES
Front-end server process	pdfes	SQL processing, instructions to back-end server	Value of <code>pd_max_users</code>	Server name ^{#4}	N	N	Y

Table B–8: Processes started by a HiRDB parallel server configuration (XDS server)

Process name		Explanation	Number of processes	Server name	Does process start for each server?		
Notation in manual	Process name				BES	DS	FES
XDS process	pdxds	SQL processing, manages memory database	For each <code>pdxds</code> operand specification, 1 ^{#12}	Server name ^{#13}	Y	N	N

Table B–9: Processes started by a HiRDB parallel server configuration (utility server)

Process name		Explanation	Number of processes	Server name	Does process start for each server?		
Notation in manual	Process name				BES	DS	FES
pdinit control process	pdinitd	Initialization utility execution process	1	0minit0	N	Y	N
pdinit execution process	pdinitb	Initialization utility BES-side execution process	1 to 2	0sinit0	Y	N	N
pdcopy backup output process	pdcopyb	Backup file output (starts at the backup output destination specified in <code>pdcopy</code> (<code>-b</code> option))	<i>Degree of duplication</i> × <i>number of concurrent pdcopy command executions</i>	0bcpy?0 ^{#5}	N	N	N
pdcopy database read process	pdcopyr	Database reads	Number of servers subject to copy ^{#6}	0rcopyN	Y	Y	N
pdrstr backup read process	pdrstrb	Backup file reads (starts at the backup read destination specified in <code>pdrstr</code> (<code>-b</code> option))	Number of concurrent <code>pdrstr</code> command executions	0brstr0	N	N	N
pdrstr master directory RDAREA read process	pdrstrm	Reads master directory RDAREA ^{#7}	Number of concurrent <code>pdrstr</code> command executions	0mrstr0	N	Y	N
pdrstr unload log read process	pdrstrl	Reads unload log file ^{#8}	Number of concurrent <code>pdrstr</code> command executions	0lrstr0	N	N	N
pdrstr database write process	pdrstrr	Database writes	Number of servers subject to recovery ^{#9}	0brstrN	Y	Y	N
pdrstr master directory RDAREA write process	pdrstrw	Writes to master directory RDAREA ^{#10}	1	0wrstr0	N	Y	N

Process name		Explanation	Number of processes	Server name	Does process start for each server?		
Notation in manual	Process name				BES	DS	FES
pdload control process	pdloadm	Controls data loads	Number of concurrent pdload command executions	0mload0	Y	Y	Y
pdrorg control process	pdrorgm	Controls database reorganization (unloading, reloading, index reorganization/re-creation, free page release, and making global buffers resident)	Number of concurrent pdrorg, pdreclaim, and pdpgbfon command executions	0mrorg0	Y	Y	Y
pdrbal control process	pdrbalm	Controls rebalancing	Number of concurrent pdrbal command executions	0mrbal0	N	Y	N
pdgetcst control process	pdgcstm	Collects optimization information	Number of concurrent pdgetcst command executions	0mgcst0	N	Y	N
pddbstd control process	pddbstd1	Controls database condition analysis (starts at MGR unit)	Number of concurrent pddbstd command executions	0mdbstd0	N	N	N
pdexp control process	pdexpm	Controls dictionary export/import (starts in unit that has export file)	1	0mexp0	N	N	N
pdplgexe control process	pdplgexm	Plug-in utility execution control	Number of concurrent executions of commands provided by plug-ins	0mplge0	N	Y	N
pdorend control process	pdorendm	Online reorganization Reflection control	1	0more0	Y	N	N
pdorend application process	pdorendl	Online reorganization Reflection processing	Degrees of duplication (number of specifications of the pdorend command's -m option)	0lore?0 ^{#11}	Y	N	N
pdtlcecx control process	pdtlcexm	Controls utility execution	<i>Number of parallel executions (pd_import_export_parallel operand value of XDS database definition) × number of starting and terminating XDSs</i>	0mucex	N	N	N

Legend

Y: Process starts.

N: Process does not start.

--: N/A

Notes

- The xxxN in the server name increases by 1, 2, ..., up to the maximum number of servers in the unit, based on the number of user servers.

- The *xxxM* in the server name increases by 2 to (maximum number of units \times 11) based on the definitions.

#1

When two or more `pdbes`, `pdfes`, or `pddic` have crashed, this value increases to the number of crashes. As the crashed `pdbes`, `pdfes`, or `pddic` transactions complete, the number of processes decreases, returning to 1 when there are no more transactions subject to recovery. Note that the upper limit value is the following:

For an FES: (Value of `pd_max_users` + value of `pd_max_reflect_process_count`) \times 2 + 7

For a DS: (value of `pd_max_dic_process` + value of `pd_max_reflect_process_count`) \times 2 + 7

For a BES: (value of `pd_max_bes_process` + value of `pd_max_reflect_process_count`) \times 2 + 7

Note that the maximum value per unit is \downarrow value of `pd_trn_rcvmsg_store_buflen` \div 72 \downarrow . The maximum number of processes per unit will be this value even if the sum of the values obtained for each FES, DS, and BES in the unit is greater than this.

#2

For a standby-less system switchover (1:1) configuration, this is the name used when an alternate BES unit is started.

#3

The REDO process starts when HiRDB starts, and stops when startup is complete.

#4

This is the server name specified by the `-s` option of the system common definition's `pdstart` operand.

#5

? is increased 0, 1, ..., *f* by the backup output process's degree of duplication (the number of specifications of the `-b` option in the control statement file specified by the `pdcopy -f` option).

#6

The process starts the same number of processes as there are servers that have RDAREAs subject to copying, as specified in `pdcopy` (`-r`, `-s`, `-u`, and `-a` options), as members.

#7

The process starts when the host that has the backup file specified in `pdrstr` (`-b` option) differs from the host that has the dictionary server. It also starts when the output destination host of the backup file specified in `pdrstr` (`-b` option) differs from the host that has the dictionary server.

#8

When `pdrstr` has specified an unload log file (`-l` option) or directory (`-d` option), the process starts if there are two or more servers to recover, or if the host that stores the unload log file is different from the host that has the server that RDAREAs subject to recovery are members of.

#9

The process starts the same number of processes as there are servers that have RDAREAs subject to recovery, as specified in `pdrstr` (`-r`, `-s`, `-u`, `-c`, and `-a` options), as members.

#10

The process starts if the host that specifies the master directory RDAREA as subject to recovery, and that has the backup file, is different from the host that has the dictionary server.

#11

? is increased 1, 2, ..., 8 by the online reorganization reflection processing's degree of duplication (the number of specifications of the `pdorend` command's `-m` option).

#12

If the `pdxsstart` command is executed, the process starts; it stops when the `pdxsstop` command is executed.

#13

This is the server name specified by the `-s` option of the system common definition's `pdxds` operand.

#14

The XDS log output process starts if the `pdstart` command executes and then becomes resident. It does not stop even if the `pdstop` command is executed.

C. Questions and Answers

This appendix discusses in question-and-answer format some of the topics concerning HiRDB system construction that can be easily misunderstood.

(1) HiRDB/Developer's Kit

Question

When do I need a HiRDB/Developer's Kit?

Answer

When creating a UAP on a machine on which a HiRDB server is installed, you do not need a HiRDB/Developer's Kit because the HiRDB server provides its facilities. You need a HiRDB/Developer's Kit when you create a UAP on a machine on which a HiRDB server is not installed.

You also need a HiRDB/Developer's Kit to create a UAP for a platform that is different from that for the HiRDB server.

(2) Execution of the database definition utility (pddef)

Question

Why didn't anything happen when I executed `CREATE TABLE` with the database definition utility (pddef)?

Answer

Check for spaces after the semicolon (;) in a pddef control statement. If there is a space, that SQL statement will not execute.

Wrong: `CREATE TABLE; Δ [Δ : space]`

Correct: `CREATE TABLE;`

(3) Maximum table size

Question

What is the maximum size of a HiRDB table?

Answer

A single table can be divided and stored in a maximum of 4,096 RDAREAs, and one RDAREA can consist of a maximum of 16 HiRDB files.

Because the maximum size of a HiRDB file is approximately 2 GB, the maximum size of a table is as follows:

- Maximum size of a table
 $4,096 \times 16 \times 2 \text{ gigabytes} = \text{approximately } 128 \text{ terabytes}$

HiRDB supports large files. When you use large files, the maximum size of a HiRDB file is 64 gigabytes. Therefore, the maximum size of a table is as follows:

- Maximum size of a table
 $4,096 \times 16 \times 64 \text{ gigabytes} = \text{approximately } 4 \text{ petabytes}$

(4) OpenTP1 and XA interface

Question

When OpenTP1 is linked to HiRDB, why does transaction commit processing for a referencing-only SQL not seem to be passed via the XA interface?

Answer

When a referencing-only SQL is executed, the process is passed to HiRDB via the XA interface at the time of commit processing. However, it might not be apparent because fewer steps are required than in the case of an updating SQL.

(5) Performance of FIX tables

Question

How much difference in terms of performance is there between `FIX` tables and non-`FIX` tables?

Answer

It is difficult to say because performance depends on the number of columns and rows subject to manipulation; however, there has been a case where the execution time for retrieval of a large amount of data in one row of a `FIX` table was approximately two-thirds of the execution time for a table for which `FIX` was not specified.

Performance is never reduced by specifying `FIX`; therefore, `FIX` should always be specified when all the following apply:

- There are no variable-length columns
- There are no column with the `NULL` value

(6) Duplicate key index

Question

Is it permissible to define an index with duplicate keys? If it's permissible, are there any problems in doing so?

Answer

Such an index can be defined (non-`UNIQUE` attribute). However, an index with many instances of keys duplicated many times (more than 200 keys) in a column is not desirable in terms of performance because a special storage structure is required and there are many index pages to be accessed.

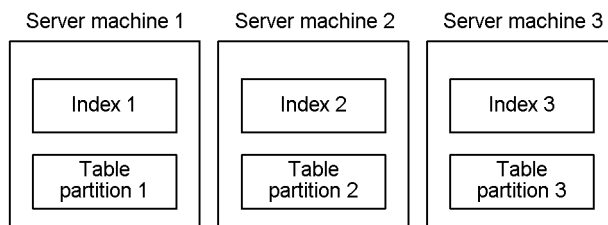
(7) Index definition for a partitioned table

Question

When an index is defined for a table that is partitioned among multiple server machines, how should the index be placed?

Answer

The index should be defined in units of table partitions, shown as follows.



(8) Response to `pdsetup -d` command

Question

Which response, `y` or `n`, should be entered when `pdsetup -d` has been entered and the `KFPS00036-Q` message is displayed requesting a response?

Answer

If you respond with `y`, you will delete files and directories that HiRDB requires for execution. In such a case, the next time you execute the `pdsetup` command, copy the files required for execution from the installation directory. If you respond with `n`, the files and directories are not deleted.

In the following cases, you should respond with `y`:

- When you are replacing the currently running HiRDB with the HiRDB that ran at the time of installation (when upgrading HiRDB, you must perform a normal shutdown using the `pdstop` command)
- When you change the HiRDB administrator's authorization identifier
- When there is an error in the files in the HiRDB operation directory, with a directory owner, or with a file mode, and they were changed or deleted

Note

If `y` is entered when the operation directory is different from the installation directory, the entire set of loaded files will be deleted from the operation directory. The set of loaded files will then be copied from the installation directory at the next `pdsetup`. Therefore, it takes some time to execute the command.

(9) Handling of synchronization point dumps**Question**

How many guaranteed-valid generations of synchronization point dump files should be provided?

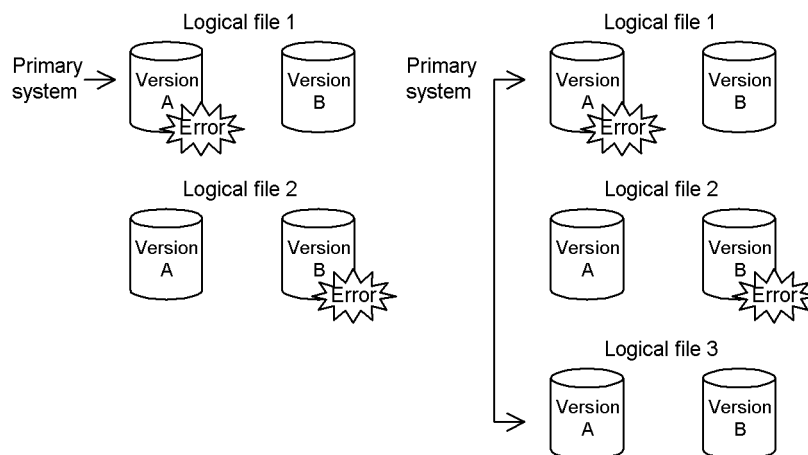
Answer

Information such as the read operation start location in the system log file is acquired in the synchronization point dump file each time a synchronization point dump is collected in order to prepare for a full rerun. The portion of the system log file beginning at the location indicated in the synchronization point dump file is overwrite-protected, so that it will not be used during a full rerun.

The number of guaranteed-valid generations is the number of generations of the synchronization point dump file that are used to overwrite-protect the system log file. In other words, if the number of guaranteed-valid generations is 1, the system log file beginning at the location indicated by the most recent synchronization point dump file is placed in overwrite-protected status. If the number of guaranteed-valid generations is 2, the system log file beginning at the location indicated by the synchronization point dump file generation immediately preceding the most recent synchronization point dump file will be placed in overwrite-protected status.

(10) Handling of status files (dual status files)**Question**

How is a pair of dual status files formed? In the figure below, if errors occur in version A of logical file 1 and in version B of logical file 2, can version B of logical file 1 and version A of logical file 2 be used to constitute a pair?

**Answer**

A pair of system files is never formed using different logical files. A logical file whose versions A and B are both normal is selected. If there is no logical file with normal versions A and B, either the unit is terminated abnormally or the system is placed in the single operation mode in order to continue processing according to the specification of `pd_sts_singleoperation`.

(11) Handling of status files (when an error occurs)**Question**

There are two operands that determine the method for handling status files in the event there are no more logical files with normal versions A and B (either A or B is erroneous):

- `pd_syssts_singleoperation=stop|continue` (for unit status file)
- `pd_sts_singleoperation=stop|continue` (for server status file)

Which option (`stop` or `continue`) should be specified?

Answer

If you specify `stop`, HiRDB (the unit for a HiRDB parallel server configuration) terminates abnormally. If you specify `continue`, the single operation takes effect on the status files.

Status files are important because they contain information that is needed for a full recovery. If you specify `continue` and an error occurs in the status file during single operation, the unit is shut down because the error is on both versions. In this case, full recovery is not possible because both versions of the current file are inaccessible. This operand should be specified according to the following guidelines:

- Specify `stop` if guaranteeing a successful full recovery is more important than avoiding online shutdown.
- Specify `continue` if an application requires uninterrupted online operation (in the worst case, the database must be rolled back to its backup copy or data loading must be re-executed instead of executing a full recovery).

(12) Handling of status files (status file definition)

Question

You say that 1-7 logical status files can be defined, but what if there is not enough room on the disk? What is a reasonable number of logical status files that should be provided?

Answer

Considering maintenance of disk integrity from error to recovery, at least three logical status files should be provided ($\text{dual files} \times 3 = 6$ files).

If there is not enough room on the disk, two logical status files are reasonable ($\text{dual files} \times 2 = 4$ physical files). If an error occurs in such a case, it must be recovered immediately.

If only one logical status file is provided and an error occurs in the status file, the database will have to be re-created.

(13) Handling of status files (status file placement)

Question

How should status files be placed on disks?

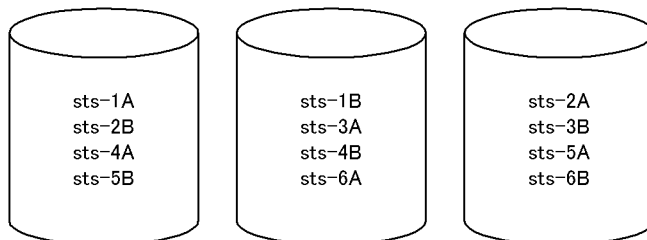
Answer

As a rule, no more than one physical status file should be placed on the same disk (if multiple files are placed on the same disk, using dual files or multiple logical files serves no purposes).

If two logical files are defined, the files should be distributed among four disks; if three logical files are defined, they should be distributed among six disks.

Reliability can be achieved with fewer disks by placing physical status files in a ring format, as shown in the following figure.

Example of placement of three logical files:



This placement configuration provides two generations with both versions available in the event of an error on one of the volumes.

(14) Peak capacity of a HiRDB file system area for work table files**Question**

How do I determine the maximum utilization value, maximum number of files that can be used, and maximum utilization increment count of a HiRDB file system area for work table files (the back-end server definition of the `pdwork` operand)?

Answer

Use the `pdfstatfs` command:

```
pdfstatfs -d name-of-HiRDB-file-system-area-for-work-table-files
```

The `-d` option displays the maximum utilization value, maximum number of files that can be used, and maximum utilization increment count for the allocated HiRDB file system area. The maximum utilization value is output as `peak capacity`, the maximum number of files that can be used is output as `peak file count`, and the maximum utilization increment count is output as `peak expand count`.

Note that the maximum utilization value can be cleared using the `pdfstatfs` command.

```
pdfstatfs -c name-of-HiRDB-file-system-area-for-work-table-files
```

The `-c` option clears (sets to 0) the maximum utilization value, maximum number of files that can be used, and maximum utilization increment count for the allocated HiRDB file system area.

Notes

The `-c` option of the `pdfstatfs` command is applicable only when the usage of the HiRDB file system area is `WORK` or `UTL`.

The `-k` option of the `pdfmkfs` command is used to specify the usage of a HiRDB file system area.

(15) HiRDB cannot be started by the `pdstart` command**Question**

What is happening when HiRDB won't start when the `pdstart` command is entered and `-prc` results in abnormal termination with `Psp4017`?

Answer

Possible causes are:

- HiRDB has not been set up correctly.
- There is no access privilege to the `/dev/HiRDB/pth` directory or this directory does not exist.

Set up HiRDB correctly. After setting up HiRDB, check the access privilege for `/dev/HiRDB/pth`.

(16) Any particular unit cannot be started by the `pdstart` command**Question**

The `pdstart` command was entered, but a unit (other than the system manager unit) doesn't start. Why?

When the OS's `ps` command is executed in a unit other than the system manager, only `pdprcd` is running (other HiRDB processes have not been started).

Answer

Check in the system common definition (`pdsys`) for the unit that cannot be started. The information defined with `pdunit` or `pdstart` probably does not match the information defined with `pdsys` for the MGR unit.

Correct `pdsys` for this unit and then restart the unit with `pdstart -u`.

(17) HiRDB startup using the `pdstart` command is slow**Question**

The `pdstart` command terminated with the message, `KFPS05078-I Unable to recognize HiRDB initialization Completion`, but why does it take so long time (up to two hours) for all units to start?

Answer

1. If KFPS00608-W-314 is output more than once, check that the same host name is specified for `pdunit` and `pdstart` in `pdsys` for all units and that the specified host is correct (existing).
2. Check that all the hosts and networks specified with HiRDB are active.

(18) pdstart command results in an error (reason code=SETUP)

Question

The `pdstart` command results in an error with the KFPS01801-E message. Why?

Answer

Following are possible reasons:

1. A HiRDB directory set in the `PDDIR` environment variable was not registered in the OS with the `pdsetup` command.
2. A prerequisite program process is not installed (applicable only to 02-00).
3. A process server process cannot start because of a shortage of semaphores in the kernel.

Take the following action:

1. Register the HiRDB directory into the OS with the `pdsetup` command.
2. Install the prerequisite program process.
3. Increase the number of semaphores defined in the system. Note that the specified value does not take effect until the system is restarted.

(19) pdstart command results in an error (reason code=TIMEOUT)

Question

The `pdstart` command results in an error with the KFPS01861-E message (reason code=TIMEOUT). Why?

Answer

Following are possible reasons:

1. It took more time to start a unit than was expected.
2. There is a specification error in the server common definition or in an individual server definition.

Take the following action:

1. Increase the value specified in the `pd_start_time_out` operand, and re-enter the `pdstart` command.
2. Check the HiRDB message output to `syslogfile` and correct the definition. Use `pdsetup -d` and enter `y` in response to the KFPS00036-Q message to delete HiRDB from the OS, then re-execute `pdsetup`.

(20) A unit cannot be started by the pdstart command

Question

When `pdstart` was executed, the KFPS01815-E message (`errno=11, 13, 22`) was output indicating that semaphore manipulation (`semop, semctl`) failed, making it impossible to start the unit. Why?

Answer

Following are possible reasons:

1. HiRDB is not installed on the corresponding machine.
2. HiRDB was not registered into the OS with the `pdsetup` command.
3. The installation directory is linked to a shared disk.

Take the following action:

1. Restart the machine, execute `pdsetup -d` to delete the machine from the OS, and then re-execute `pdsetup`.

2. If HiRDB is installed on that machine, execute `pdsetup`.
3. Place the installation directories on a local disk at the local node.

Remarks

HiRDB cannot be used simply by copying its files from another environment; it must actually be installed.

(21) Database definition utility (pddef) results in an error**Question**

`pdinit` (database initialization utility) executed successfully, but `pddef` (definition utility) results in an error. What causes this?

Answer

Possible causes are:

- *If no response or connection error results, some information is probably missing in the environment variables.* Check the values set in `PDHOST` and `PDNAMEPORT`.

PDHOST

Specify the name of the host where HiRDB was started. This is the host name specified with `pdstart` in the HiRDB system common definition (`pdsys`).

PDNAMEPORT

This is the port number specified with `pd_name_port` in the HiRDB system common definition (`pdsys`).

- *If a connection error results, an invalid value might be set in the PDUSER environment variable.* Only one authorization identifier having the DBA privilege exists immediately after executing `pdinit`. Specify the authorization identifier and password in the `PDUSER` environment variable as follows:

Bourne shell

```
PDUSER="authorization-identifier" / "password"
export PDUSER
```

C shell

```
setenv PDUSER "authorization-identifier" / "password"
```

Notes

1. For details about the authorization identifier and password immediately after executing `pdinit`, see *Options in Database Initialization Utility (pdinit)* in the manual *HiRDB Version 9 Command Reference*.
2. When the `PDUSER` environment variable is set, the authorization identifier and password must each be enclosed in double quotation marks and the entire string must be enclosed in apostrophes. The same applies when any other HiRDB utility or UAP is executed.

(22) CREATE TABLE statement LOB column definition**Question**

In a column definition using the `CREATE TABLE` statement, what is the difference in the memory requirements of HiRDB servers and HiRDB clients and in the volume of data transfer if you specify the LOB column maximum length (for example, 300 megabytes) as opposed to using the default (2 gigabytes)?

Answer

Regardless of whether the maximum length of LOB columns is specified, the memory requirements and the data transfer volume are the same. The memory requirements and data transfer volume when LOB columns are used for retrieval or updating depend not on the maximum length in the column definition, but on the actual length and the defined length of the embedded variables during retrieval or updating. If there is no limit on the storage of binary data, it can be limited by the maximum length of LOB columns.

(23) Antivirus software

Question

After I installed antivirus software, a UAP could no longer connect to HiRDB. The antivirus software's firewall seems to be the problem. Which port numbers should I specify as exceptions on the firewall's exceptions list?

Answer

22.5.1 List of port numbers specified in HiRDB lists port numbers used by HiRDB. Specify exceptions for the following port numbers, which are found on these lists.

Note that the port numbers that you need to specify as exceptions vary depending on where the firewall was installed.

- If the firewall was installed on the HiRDB server side

Specify exceptions for the following port numbers, which are specified in the system common definition and unit control information definition.

- HiRDB port numbers
- Scheduler process port numbers

If the operands that specify these port numbers have not been specified, specify them, including the port numbers as exceptions.

- If the firewall was installed on the client side

Specify exceptions for the following port number, which is specified in the client environment definition.

- Client's reception port number

If the client environment definition that specifies this port number has not been specified, specify it, including the port number as an exception.

Index

A

abstract data type 365
 defining 176
 null value for 178
abstract data type column structure base table 179
AFTER trigger 333
AIX, specifying parameters unique to 728
alias (for command) 752
ALTER TRIGGER 333
ARRAY option (CREATE TABLE) 364
asynchronous READ facility 289
asynchronous XA call 194
automatic opening
 HiRDB parallel server configuration 257, 261
 HiRDB single server configuration 228, 231

B

back-end server definition, creating 86
base row log information, determining amount of 640
basic attribute (JP1 linkage) 212
batch index creation, handling errors during 183
BEFORE trigger 333
BINARY type 350
BLOB type 350
boundary value, specifying 312
branch row log information, determining amount of 643
buffer hit rate 287

C

candidate key 344
character encoding
 specifying using the pdsetup command 67
character special file, using 92
check constraint 412
 table 412
check pending status 394, 413
client environment definition
 registering, in transaction manager 203
 setting (database creation) 136
 specifying (multi-HiRDB) 276
close character string 203
cluster key, specifying 345
CLUSTER KEY option (CREATE TABLE) 345
column information work table, obtaining size of 669
column information work tables, obtaining maximum
 number of 670
command
 environment setup using 79
 overview of environment setup using 80
commit-time reflection processing 292
compressed column 422
compressed table 422
configuration file, creating 749
constructor function 177, 367

conventions

diagrams I
fonts and symbols I
version numbers II

CREATE INDEX 439
create rdarea statement 107, 111, 478, 480
CREATE SCHEMA 136
CREATE TRIGGER 333
CREATE TYPE 176

D

database
 creating 135
 maximum value for 785
database compression facility 422
database definition utility
 execution of (pddef) 799
 results in error (pddef) 805
database initialization utility 107
database load utility 137
database structure modification utility 111
database update log acquisition method 181
database update log acquisition mode 137
 types of 137
data conversion facility, specifying (database creation) 137
data dictionary LOB RDAREA
 creating 117
 determining size of 621
 page length of 621, 622
 placing (HiRDB parallel server configuration) 266
 placing (HiRDB single server configuration) 236
 total number of pages in 621, 622
data dictionary RDAREA
 determining size of 594
 determining size of (for storing database state
 analyzed tables and database management tables)
 617
 page length of 594
 total number of pages in 594
data directory RDAREA
 determining size of 620
 page length of 620
 total number of pages in 620
data global buffer 285
 allocating 282
data in table, storing 180
data integrity
 check constraint 414
 referential constraint 400
data length
 for abstract data type 582
 for repetition columns 582
 for variable-length character string type 581
 of abstract data type column, obtaining 581
 of repetition columns, obtaining 582
data lengths, list of 578

data loading
 with synchronization point specification 140
 with synchronization point specification (handling utility abnormal termination errors) 188

data local buffer 300
 allocating 300

data manipulation
 check constraint 414
 referential constraint 400

data page 288

data storage status, checking 162, 182

data types for which indexes cannot be defined 439

deadlock between referenced and referencing tables 389

DECIMAL signed normalized number, facility for conversion to 137

default constructor function 177, 367

deferred write processing 290
 facility for parallel writes in 291

deferred write trigger 290
 updated output page rate for 290

deletion prevented duration 357

diagram conventions I

dictionary server definition, creating 85

directory
 created by HiRDB 6
 created by HiRDB administrator 6

disaster recovery 4

distributed transaction processing 193

divided-input data file 150
 creating 150

DROP TRIGGER 333

DTP 193

duplicate key index 800

dynamic registration 199

dynamic transaction registration 194

dynamic updating of global buffers 282

E

EBCDIK 353

encapsulation level 177, 368

environment
 overview of setting up, using command 80
 setting up, using command 79

environment variable
 setting 68
 setting (multi-HiRDB) 276

error occurs during updating to HiRDB update version, operation when 46

event
 managing by JP1/IM 214
 monitoring by JP1/IM (overview) 214

event attribute definition file 214

event notice, sending (JP1 linkage) 212

exceptional key 444

EXCEPT VALUES option (CREATE INDEX) 445

extended attribute (JP1 linkage) 212

extended system-defined scalar function
 amount of system log information output during definition of 652

F

facility for monitoring free space remaining for system log files (HiRDB parallel server configuration) 257

facility for monitoring free space remaining for system log files (HiRDB single server configuration) 228

failed page searches by free space reuse facility, number of 475

falsification prevented table 356

falsification prevention facility 356

FES host direct connection facility 245

file
 created by HiRDB 6
 created by HiRDB administrator 6

file size
 required for execution of access path display utility (pdvwopt) 692
 required for execution of database condition analysis utility (pddbstd) 688
 required for execution of database copy utility (pdcopy) 688
 required for execution of database load utility (pdload) 678
 required for execution of database reorganization utility (pdrorg) 680
 required for execution of dictionary import/export utility (pdexp) 691
 required for execution of integrity check utility (pdconstck) 694
 required for execution of optimizing information collection utility (pdgetcst) 692
 required for execution of rebalancing utility (pdrbal) 693
 required for execution of statistics analysis utility (pdstedit) 685
 required for utility execution, determining 678

first dimension partitioning column 327

FIX attribute, specifying 342

FIX hash partitioning 313
 examples of 318

FIX table, performance of 800

flexible hash partitioning 313
 examples of 318

floating machine 325

floating server 325
 placement of 240

font conventions I

foreign key 385

FOREIGN KEY 386

free page reuse mode 470

free space required for upgrading 30

free space reuse facility 470

front-end server definition, creating 85

full segment 461

G

global buffer
 allocating 282, 284
 designing 281
 dynamic updating of 282
 LRU management of 293

- pre-writing of 299
- setting number of sectors of 287
- global buffer definition, example 285
- global buffer pool, notes about allocating to list RDAREA 284
- global buffer residence utility 299
- guaranteed-valid generations
 - number of (HiRDB parallel server configuration) 260
 - number of (HiRDB single server configuration) 231

H

- hash facility for hash row partitioning 307, 326
- hash functions, types of 313
- hash partitioning 312
- high-speed connection facility 245
- HiRDB
 - 64-bit mode migration error of 56
 - continuous 24-hour operation 3
 - directory organization of 6
 - file organization of 6
 - installing 57, 61
 - installing new version while retaining old version 37
 - in transaction manager, registering 198
 - linking to other products 191
 - linking with OLTP system 193
 - maximum value of 784
 - minimum value of 784
 - notes about upgrading 279
 - online distribution of 48
 - port number used by 773
 - post-installation procedure 63
 - registering, in operating system 66
 - replacing existing version with new version 35
 - starting, for the first time 110
 - terminating normally 32
 - uninstalling 75
 - update version 42
 - updating to update version of 42
 - upgrading 30
 - upgrading error 38
 - upgrading plug-in of 37
- HiRDB/Developer's kit 799
- HiRDB administrator
 - registering 58
 - registering (multi-HiRDB) 276
- HiRDB Advanced High Availability 327
- HiRDB client to HiRDB server, connecting 760
- HiRDB Dataextractor, linking to 192
- HiRDB Datareplicator, linking to 192
- HiRDB directory
 - backing up 64
 - backing up disk containing 64
 - backing up files in (upgrading) 34
 - creating 63
 - creating (multi-HiRDB) 276
 - deleting files from 64
 - information to be specified in 63
 - making backup copy of 64
 - notes when creating 63
- HiRDB environment setup 3
- HiRDB event
 - that can be sent 212
 - that can be sent to JP1/Base 213
- HiRDB file name
 - maximum value for 787
 - minimum value for 787
- HiRDB file system area
 - creating 92
 - creating, for list RDAREA 95
 - creating, for RDAREA 93
 - creating, for system file 93
 - creating, for utility 94
 - creating, for work table file 94
 - creating in a character special file or block special file 71
 - designing, for list RDAREA (HiRDB parallel server configuration) 253
 - designing, for list RDAREA (HiRDB single server configuration) 224
 - designing, for RDAREA (HiRDB parallel server configuration) 250
 - designing, for RDAREA (HiRDB single server configuration) 221
 - designing, for system file (HiRDB parallel server configuration) 251
 - designing, for system file (HiRDB single server configuration) 222
 - designing, for utility (HiRDB parallel server configuration) 252
 - designing, for utility (HiRDB single server configuration) 223
 - designing, for work table file (HiRDB parallel server configuration) 252
 - designing, for work table file (HiRDB single server configuration) 223
 - designing (HiRDB parallel server configuration) 250
 - designing (HiRDB single server configuration) 221
 - determining size of (work table file) 668
 - for work table files, peak capacity of 803
 - maximum size of 225, 254
 - types of 92
- HiRDB group, setting up 59
- HiRDB parallel server configuration, estimating size of memory required for 518
- HiRDB reserved port facility 781
- HiRDB server
 - using multi-connection address facility to connect to 761
 - with FQDN specified, connecting to 760
- HiRDB setup automation shell, creating 50
- HiRDB single server configuration, estimating size of memory required for 484
- HiRDB Staticizer Option 4
- HiRDB status, checking (upgrading) 32
- HiRDB system construction with OLTP linkage 195
- HiRDB system definition
 - creating 82
 - creating (HiRDB parallel server configuration) 84
 - creating (HiRDB single server configuration) 82
 - modifying (excluding UAP environment definitions) 89
 - sharing 88

- specifying (multi-HiRDB) 276
- HiRDB system definition file
 - configuring (HiRDB parallel server configuration) 87
 - configuring (HiRDB single server configuration) 83
 - sharing 87
- HiRDB Text Search Plug-in 159
- HiRDB XA library 193
 - functions supported by 194
- host name 760

I

- index 439
 - creating 439
 - defining 180
 - definition for partitioned table 800
 - designing 437
 - having B-tree structure 439
 - items to be examined during design of 438
 - key lengths, list of 587
 - maximum key length of 439
 - multicolumn 443
 - non-partitioning key index 446
 - partitioning guideline of 447
 - partitioning key index 446
 - row partitioning of 446
 - single-column 443
 - storage pages, calculating number of 585, 588
 - storage pages, number of 611
 - using, with exceptional key value set 444
- index global buffer 284
 - allocating 282
- index local buffer 300
 - allocating 300
- index log information
 - determining amount of 646
 - determining amount of, for index page splitting 646
- index storage RDAREA, notes on estimating size of 459
- inheritance 177, 365
- inner replica facility 4, 211
- input data file UOC 152
- insert history maintenance column 357
- INSERT ONLY operand (CREATE TABLE) 356
- installation directory, creating 59
- installer to update HiRDB, using 42
- installing
 - multi-HiRDB 276
 - plug-in 124
- InstallTiming parameter 49
- instantiating (temporary table) 428
- inter-process memory communication shared memory
 - HiRDB parallel server configuration 523
- Inter-process memory communication shared memory 488
- IP address 760

J

- JP1/Automatic Job Management System 3 212
- JP1/Base 212
- JP1/IM 214
- JP1/Integrated Management - Manager 212

- JP1/Software Distribution 48
- JP1/Software Distribution Workstation
 - preparation on 49
 - specifying registration file for 49
- JP1/System Event Service 212

K

- kernel parameter 58
 - estimating (AIX) 728
 - estimating (HP-UX) 720
 - estimating (Linux) 732
 - estimating (Solaris) 725
- key range partitioning 312
 - examples of (with boundary values specified) 317
 - examples of (with storage condition specified) 316
- KFPS00036-Q message 800
- KFPS01801-E message 804
- KFPS01815-E message 804
- KFPS01861-E message 804
- KFPS05078-I message 803

L

- large file
 - creating 93
- LD_LIBRARY_PATH 69
- LIBPATH 69
- libraries for multi-thread, notes about 209
- library sharing
 - multi-HiRDB 277
 - releasing (upgrading) 32
- listen queue 739
- list RDAREA
 - creating 120
 - creating HiRDB file system area for 95
 - designing 468
 - designing HiRDB file system area for 224, 253
 - determining size of 631
 - placing (HiRDB parallel server configuration) 267
 - placing (HiRDB single server configuration) 238
- LOB column 350
- LOB global buffer 285
 - allocating 284
- local buffer 300
 - designing 281
- location information work table, obtaining size of 671
- location information work tables, obtaining maximum number of 672
- log acquisition mode 137, 181
- LRU management method 293

M

- master directory RDAREA
 - determining size of 619
 - page length of 619
 - total number of pages in 619
- matrix-partitioned table 327
- matrix partitioning 327
- maximum utilization value (HiRDB file system area for work table) 803

- memory
 - allocation of (HiRDB parallel server configuration) 518
 - allocation of (HiRDB single server configuration) 484
 - memory requirement
 - calculation of (HiRDB parallel server configuration) 521
 - calculation of (HiRDB single server configuration) 487
 - checking (upgrading) 33
 - during array FETCH (front-end server) 571
 - during array FETCH (HiRDB single server configuration) 516
 - during BLOB data retrieval (back-end server) 570
 - during BLOB data retrieval (dictionary server) 570
 - during BLOB data retrieval (front-end server) 569
 - during BLOB data retrieval (HiRDB single server configuration) 515
 - during BLOB data updating (back-end server) 570
 - during BLOB data updating (dictionary server) 570
 - during BLOB data updating (front-end server) 569
 - during BLOB data updating (HiRDB single server configuration) 515
 - during block transfer (front-end server) 571
 - during block transfer (HiRDB single server configuration) 516
 - during execution of rapid grouping facility (HiRDB parallel server configuration) 562
 - during execution of rapid grouping facility (HiRDB single server configuration) 508
 - during hash join 509, 564
 - during SQL execution (HiRDB parallel server configuration) 562
 - during SQL execution (HiRDB single server configuration) 508
 - during SQL preprocessing (HiRDB parallel server configuration) 568
 - during SQL preprocessing (HiRDB single server configuration) 514
 - during subquery hash execution 509, 564
 - estimating, for HiRDB parallel server configuration 518
 - estimating, for HiRDB single server configuration 484
 - for Java virtual machine (HiRDB parallel server configuration) 524
 - for Java virtual machine (HiRDB single server configuration) 489
 - memory size
 - required for execution of access path display utility (pdvwopt) 711
 - required for execution of database condition analysis utility (pddbst) 705
 - required for execution of database copy utility (pdcopy) 706
 - required for execution of database definition utility (pddef) 698
 - required for execution of database initialization utility (pdinit) 697
 - required for execution of database load utility (pdload) 698
 - required for execution of database recovery utility (pdrstr) 707
 - required for execution of database reorganization utility (pdorg) 701
 - required for execution of database structure modification utility (pdmod) 703
 - required for execution of dictionary import/export utility (pdexp) 710
 - required for execution of optimizing information collection utility (pdgetest) 705
 - required for execution of statistics analysis utility (pdstedit) 704
 - required for utility execution, determining 697
 - message queue, estimating size of 736
 - multicolumn partitioning 446
 - multi-connection facility 194
 - X/Open XA interface environment 200
 - multi-HiRDB
 - installing 276
 - setting environment for 276
 - system design for 276
 - multiple front-end server 240
 - setting up 243
 - using 240
 - multi-thread, XA interface supporting 194
- ## N
-
- network configuration
 - example of (FQDN) 760
 - example of (multi-connection address facility) 762
 - network definition
 - example of (FQDN) 760
 - example of (multi-connection address facility) 762
 - new page allocate mode 470
 - new values correlation name 333
 - NFS 87
 - no-log mode 138, 181
 - non-full segment 461
 - non-UNIQUE attribute 800
 - normalizing table 308
 - NO SPLIT option 349
 - no-split option, specifying 348
 - NOT NULL constraint 344
 - number reserved 473
- ## O
-
- old correlation name 333
 - old or new values alias 333
 - OLTP products supported for linking 193
 - open character string 200
 - OpenTP1 193, 799
 - OpenTP1/Server Base Enterprise Option 193
 - operand, specifying port number 776
 - operating system parameter 58
 - checking (upgrading) 33
 - optimizing based on cost 440
 - option program product
 - installing 61
 - registering, in operating system 66
 - OS parameter 58

estimating (AIX) 728
 estimating (HP-UX) 720
 estimating (Linux) 732
 estimating (Solaris) 725
 specifying 719
 OTS 194
 override 367

P

page 464
 allocating 466
 determining length of 464
 releasing 467
 page search mode switchovers, number of 475
 partitioning key 312
 selecting 313
 password, changing (database creation) 136
 PATH 69
 PCTFREE option (CREATE TABLE, CREATE INDEX, ALTER TABLE) 466
 PCTFREE option (CREATE TABLE) 463
 pd_assurance_table_no operand 473
 pd_check_pending operand 385, 412
 pd_dbbuff_lru_option operand 293
 pd_dbbuff_rate_uppage operand 291
 pd_dbsync_point operand 290, 292
 pd_dfw_awt_process operand 291
 pd_inner_replica_control operand 211
 pd_inner_replica_lock_shift operand 211
 pd_jpl_event_level operand 212
 pd_jpl_use operand 212
 pd_log_dual operand 227, 256
 pd_log_org_no_standby_file_opr operand 211
 pd_log_org_reflected_logpoint operand 211
 pd_log_rerun_reserved_file_open operand 228, 257
 pd_log_rpl_no_standby_file_opr operand 192
 pd_log_singleoperation operand 227, 256
 pd_lv_mirror_use operand 211
 pd_max_reflect_process_count operand 211
 pd_max_temporary_object_no operand 480
 pd_max_tmp_table_rdarea_no operand 480
 pd_pageaccess_mode operand 297
 pd_registered_port operand 773, 781
 pd_rpl_hdepath operand 192
 pd_rpl_init_start operand 192
 pd_shared_rdarea_use operand 477
 pd_spd_assurance_count operand 231, 260
 pd_spd_dual operand 230, 260
 pd_spd_reduced_mode operand 231, 260
 pd_spd_reserved_file_auto_open operand 232, 261
 pd_spool_cleanup_interval operand 64
 pd_spool_cleanup operand 64
 pd_sts_singleoperation operand 234, 263
 pd_syssts_singleoperation operand 234, 263
 pd_tmp_table_initialize_timing operand 481
 pdadmvr command 43
 pdbes (HiRDB parallel server configuration) 795
 pdbufmod command 282
 pdchgconf command 90
 pdconfchk command 82
 PDCONFPATH 69
 pdcopyb (HiRDB parallel server configuration) 796
 pdcopyb (HiRDB single server configuration) 790
 pdcopyr (HiRDB parallel server configuration) 796
 pdcopyr (HiRDB single server configuration) 790
 pdcspool command 64
 pddbst1 (HiRDB parallel server configuration) 797
 pddbst1 (HiRDB single server configuration) 791
 pddef control statement 799
 pddic (HiRDB parallel server configuration) 795
 PDDIR 68
 pdexpm (HiRDB parallel server configuration) 797
 pdexpm (HiRDB single server configuration) 791
 pdfes (HiRDB parallel server configuration) 796
 pdfmkfs command 92, 477, 480
 pdgcstm (HiRDB parallel server configuration) 797
 pdgcstm (HiRDB single server configuration) 791
 pdinit 107
 pdinitb (HiRDB parallel server configuration) 796
 pdinitd (HiRDB parallel server configuration) 796
 pdinitd (HiRDB single server configuration) 790
 pdload 137
 pdloadm (HiRDB parallel server configuration) 797
 pdloadm (HiRDB single server configuration) 790
 pdloginit command 96
 pdls command 44
 pdls -d ust command
 when upgrading HiRDB 32
 pdmod 111
 pdopsetup command 68
 pdorendl (HiRDB parallel server configuration) 797
 pdorendl (HiRDB single server configuration) 791
 pdorendm (HiRDB parallel server configuration) 797
 pdorendm (HiRDB single server configuration) 791
 pdpgbfon 299
 pdplgexm (HiRDB parallel server configuration) 797
 pdplgexm (HiRDB single server configuration) 791
 pdplgrgst command 126
 pdplgset command 124
 pdplugin operand 128
 pdprgcopy command 44
 pdprgrefresh command 44
 pdrbalm (HiRDB parallel server configuration) 797
 pdrorgm (HiRDB parallel server configuration) 797
 pdrorgm (HiRDB single server configuration) 791
 pdrplstart command 247
 pdrplstop command 247
 pdrstrb (HiRDB parallel server configuration) 796
 pdrstrb (HiRDB single server configuration) 790
 pdrstrl (HiRDB parallel server configuration) 796
 pdrstrl (HiRDB single server configuration) 790
 pdrstrm (HiRDB parallel server configuration) 796
 pdrstrm (HiRDB single server configuration) 790
 pdrstrr (HiRDB parallel server configuration) 796
 pdrstrr (HiRDB single server configuration) 790
 pdrstrw (HiRDB parallel server configuration) 796
 pdrstrw (HiRDB single server configuration) 790
 pdsds (HiRDB single server configuration) 790
 pdsetup command 66
 pdsetup -d command, response to 800
 pdstart command
 fails to start any particular unit by 803
 fails to start HiRDB by 803

- fails to start unit by 804
 - results in error 804
 - when HiRDB startup is slow using 803
- pdstsinit command 97
- pdtlexm (HiRDB parallel server configuration) 797
- pdxds (HiRDB parallel server configuration) 796
- pdxds (HiRDB single server configuration) 790
- percentage of free pages in a segment 462
- percentage of unused space in a page 465
 - obtaining 466
 - setting 465
- plug-in
 - canceling setup of 133
 - deleting 132
 - installing 62, 124
 - owner of 126
 - registering 126
 - setting up 124
 - uninstalling 133
 - upgrading 130
- plug-in environment, setting up 123
- plug-in index 451
 - defining 161
 - effects of row partitioning of 452
 - row partitioning of 452
- port number 773
- port numbers, list of 776
- POSIX library version 68
- prefetch facility 288, 461
- pre-update log acquisition mode 137, 181
- primary key, specifying 344
- PRIMARY KEY option (CREATE TABLE) 344
- PRIVATE 368
- process private area (memory requirement) 521
- PROTECTED 368
- Psp4017 803
- PUBLIC 368

Q

questions and answers 799

R

RDAREA

- creating HiRDB file system area for 93
- deleting unneeded 152
- designing 457
- designing HiRDB file system area for 221, 250
- determining size of 573
- items to be examined during design of 458
- maximum value for 459
- minimum value for 459
- placing (HiRDB parallel server configuration) 265
- placing (HiRDB single server configuration) 236
- temporary table 480

read only 194

Real Time SAN Replication 4

reason code=SETUP 804

reason code=TIMEOUT 804

rebalancing facility 307, 326

RECOVERY operand 182

CREATE TABLE 138

recovery-unnecessary front-end server 246

recovery-unnecessary front-end server unit 246

reduced mode operation

- HiRDB parallel server configuration 260
- HiRDB single server configuration 231

reference buffer 282

referenced table 385

reference-only back-end server 369

referencing table 385

referential constraint 385

registry, deleting 133

registry facility, initializing 127

registry information, registering 128

registry LOB RDAREA

- determining size of 630
- page length of 630

registry RDAREA, determining size of 628

reload-not-completed data status 358

remote installation 48

remote shell execution environment, setting 70

repetition column 363

replication facility, linking to 192

resource manager 193

RM 193

RMM 208

RM name 200

RM-related object name 203

RM switch name 200

routine 367

row-partitioned table 312

- creating 153

S

sample configuration file, name of 743

sample database file, name of 743

sample file 741

- use of 749

sample UOC file, name of 744

schema, defining (database creation) 136

second dimension partitioning column 327

segment 461

- allocating 463
- determining size of 461
- free 461
- releasing 463
- setting percentage of free pages in 462
- unused 461
- used 461

SEGMENT REUSE option 473

- ALTER TABLE 473

segments, number of 621, 622, 627

semaphore, estimating size of 736

server name operand 478

setting

- when firewall is installed 768
- when NAT is installed 768

shared directory 89

shared memory

- formula for, used by each server (HiRDB parallel server configuration) 552

- formula for, used by single server (HiRDB single server configuration) 503
 - memory requirement 523
 - used by back-end server 555
 - used by dictionary server 554
 - used by front-end server 552
 - used by global buffer (HiRDB parallel server configuration) 559
 - used by global buffer (HiRDB single server configuration) 507
 - used by single server 503
 - used by unit controller (HiRDB parallel server configuration) 529
 - used by unit controller (HiRDB single server configuration) 494
- shared RDAREA 477
- shared table 369
- shell script for executing operation commands under aliases, creating 752
- SHLIB_PATH 69
- simple setup tool 77
- single-column partitioning 446
- single operation
 - HiRDB parallel server configuration 256
 - HiRDB single server configuration 227
- single-phase optimization 194
 - notes on 209
- single server definition, creating 82, 84
- skipped effective synchronization point dump monitoring facility (HiRDB parallel server configuration) 257
- skipped effective synchronization point dump monitoring facility (HiRDB single server configuration) 228
- snapshot method 297
- space conversion facility 137
- split compression size 423
- SQL, notes about (X/Open XA interface environment) 209
- SQL reserved word definition
 - creating (HiRDB parallel server configuration) 86
 - creating (HiRDB single server configuration) 83
- SQL session shared attribute 480
- SQL session-specific attribute 480
- SQL session-specific temporary table 428
- static registration 199
- status file
 - creating 97
 - designing (HiRDB parallel server configuration) 261
 - designing (HiRDB single server configuration) 232
 - determining number of records in 658
 - determining size of 658
 - handling of (dual status file) 801
 - handling of (status file definition) 802
 - handling of (status file placement) 802
 - handling of (when error occurs) 801
 - single operation of (HiRDB parallel server configuration) 262
 - single operation of (HiRDB single server configuration) 233
- status-file double operation
 - HiRDB parallel server configuration 262
 - HiRDB single server configuration 233
- status-file single operation
 - HiRDB parallel server configuration 262
 - HiRDB single server configuration 233
- storage condition, specifying 312
- storage requirement
 - for HiRDB 483
 - for utility execution 677
- substitutability 178, 366
- subtype 365
- supertype 365
- suppress option, specifying 347
- SUPPRESS option (CREATE TABLE) 347
- symbol conventions I
- symbolic link 73
- synchronization point, line number of 140
- synchronization point dump, handling of 801
- synchronization point dump file
 - automatic opening of (HiRDB parallel server configuration) 261
 - automatic opening of (HiRDB single server configuration) 231
 - creating 96
 - designing (HiRDB parallel server configuration) 259
 - designing (HiRDB single server configuration) 230
 - determining number of records in 657
 - determining size of 657
 - duplexing of 230, 259
 - number of guaranteed-valid generations for (HiRDB parallel server configuration) 260
 - number of guaranteed-valid generations for (HiRDB single server configuration) 231
 - reduced mode operation for (HiRDB parallel server configuration) 260
 - reduced mode operation for (HiRDB single server configuration) 231
- system common definition
 - creating (HiRDB parallel server configuration) 84
 - creating (HiRDB single server configuration) 82
- system configuration
 - maximum value for 784
 - of HiRDB parallel server configuration 242
 - of HiRDB single server configuration 220
 - sample file 745
- system design
 - for HiRDB parallel server configuration 240
 - for HiRDB single server configuration 218
 - for multi-HiRDB 276
- system file
 - creating 96
 - creating HiRDB file system area for 93
 - designing (HiRDB parallel server configuration) 255
 - designing (HiRDB single server configuration) 226
 - designing HiRDB file system area for 222, 251
 - example of creating (HiRDB parallel server configuration) 100
 - example of creating (HiRDB single server configuration) 97
- system log file
 - automatic opening of (HiRDB parallel server configuration) 257
 - automatic opening of (HiRDB single server configuration) 228
 - creating 96
 - designing (HiRDB parallel server configuration) 255

- designing (HiRDB single server configuration) 226
 - determining size of 634
 - double operation of (HiRDB parallel server configuration) 256
 - double operation of (HiRDB single server configuration) 227
 - duplexing of (HiRDB parallel server configuration) 256
 - duplexing of (HiRDB single server configuration) 227
 - record length of 634
 - record length of (HiRDB parallel server configuration) 258
 - record length of (HiRDB single server configuration) 229
 - records, number of 634
 - single operation of (HiRDB parallel server configuration) 256
 - single operation of (HiRDB single server configuration) 227
 - total size of 634
 - system log file automatic extension facility (HiRDB parallel server configuration) 257
 - system log file automatic extension facility (HiRDB single server configuration) 228
 - system log information
 - determining size of 635
 - output depending on SQL manipulation, amount of 652
 - output during database creation by utility, amount of 649
 - output during execution of RDAREA automatic extension facility, amount of 652
 - output during index definition, amount of 637
 - output during table data updating, amount of 639
 - output during table definition, amount of 636
 - system manager, placement of 240
 - system RDAREA
 - backing up (upgrading) 31
 - creating 107
 - placing (HiRDB parallel server configuration) 265
 - placing (HiRDB single server configuration) 236
 - system reconfiguration command 90
 - system switchover facility, using 4
- ## T
-
- table
 - actual 340
 - containing abstract data type 365
 - containing repetition column 363
 - creating, containing plug-in-provided abstract data type 159
 - creating, containing user-defined abstract data type 176
 - creating, with LOB column 156
 - defining 179
 - definition information of (sample file) 746
 - designing 303
 - design method for row partitioning of 312
 - effects of row partitioning of 320
 - forms of row partitioning of 319
 - items to be examined during design of 304
 - matrix partitioning of 327
 - maximum size of 799
 - normalizing 308
 - partitioning, among multiple servers 449
 - partitioning, within one server 449
 - procedure for creating, with abstract data type (SGMLTEXT type) 159
 - row partitioning of 312
 - storage pages, calculating number of 575, 583, 594
 - storing data in 180
 - view 340
 - table integrity
 - how to check (check constraint) 415
 - how to check (referential constraint) 403
 - temporary table 428
 - temporary table index 428
 - TIMEOUT 804
 - TM 193
 - TP1/Resource Manager Monitor 208
 - TPBroker for C++ 193
 - transaction completion type 209
 - transaction manager 193
 - example of registering in 204
 - information to be registered in 200
 - modifying registration information in 206
 - registering HiRDB in 198
 - transaction-specific temporary table 428
 - transaction transfer 194, 197
 - trigger
 - defining 332
 - management of 336
 - trigger action search conditions 332
 - trigger event 332
 - trigger event SQL 332
 - trnstring operand 198
 - TUXEDO 193
- ## U
-
- UAP environment definition
 - creating (HiRDB parallel server configuration) 86
 - creating (HiRDB single server configuration) 83
 - modifying 90
 - uninstalling plug-in 133
 - unique index 140
 - uniqueness constraint 344
 - unit control information definition
 - creating (HiRDB parallel server configuration) 84
 - creating (HiRDB single server configuration) 82
 - UOC 152
 - updatable back-end server 369
 - updatable column 356
 - update buffer 282
 - update patch to update HiRDB, using 43
 - upgrading
 - backing up files in HiRDB directory before 34
 - backing up system RDAREAs before 31
 - before 30
 - checking HiRDB status before 32
 - checking memory requirement before 33
 - checking operating system parameters before 33

- checking to see whether HiRDB is online before 31
- checking total number of records in system log file
 - before 33
- free space checking before 30
- HiRDB 30
- HiRDB plug-in 37
- plug-in 130
- releasing library sharing before 32
- terminating HiRDB normally before 32
- used free pages, release of 467
- used free segment 461
- user's own coding 152
- user LOB RDAREA
 - creating 114
 - determining size of 627
 - placing (HiRDB parallel server configuration) 267
 - placing (HiRDB single server configuration) 237
- user RDAREA
 - creating 111
 - determining size of 574
 - formula for calculating total number of pages in 574
 - placing (HiRDB parallel server configuration) 266
 - placing (HiRDB single server configuration) 237
- UTF16 353
- utility
 - creating HiRDB file system area for 94
 - designing HiRDB file system area for 223, 252
- utility special unit
 - installing 219
 - installing (multi-HiRDB) 276

V

- valid period of data (temporary table) 429
- version number conventions II
- view table, creating 340

W

- WebLogic Server 193
- WITHOUT ROLLBACK option, specifying 354
- work disk 325
- work file for sorting
 - buffer size for determining size of 695
- work table file 666
 - creating HiRDB file system area for 94
 - designing HiRDB file system area for 223, 252
 - determining maximum number of extensions for 676
 - determining size of 665
 - SQL that requires 666
 - used by SQL statement, size of 668
 - used by utility, size of 673
- work table files, determining maximum number of 675

X

- X/Open XA interface 193
- xa_switch_t structure-name 200
- XA interface 799
 - supporting multi-thread 194