
For Windows Systems
Scalable Database Server

HiRDB Version 8

Installation and Design Guide

3020-6-352(E)

HITACHI

■ Relevant program products

List of program products:

For the Windows 2000, Windows XP Professional, Windows XP x64 Edition, Windows Server 2003, Windows Server 2003 x64 Edition, Windows Server 2003 R2, or Windows Server 2003 R2 x64 Edition operating system:

P-2462-7187 HiRDB/Single Server Version 8 08-00

P-2462-7387 HiRDB/Parallel Server Version 8 08-00

P-2462-7H87 HiRDB Non Recover Front End Server Version 8 08-00

P-2462-7J87 HiRDB Advanced High Availability Version 8 08-00

P-2462-7K87 HiRDB Advanced Partitioning Option Version 8 08-00

For the Windows XP x64 Edition or Windows Server 2003 x64 Edition operating system:

P-2962-7187 HiRDB/Single Server Version 8 08-00

P-2962-7387 HiRDB/Parallel Server Version 8 08-00

P-2962-1187 HiRDB/Run Time Version 8 08-00

P-2962-1287 HiRDB/Developer's Kit Version 8 08-00

For the Windows Server 2003 (IPF) operating system:

P-2862-7187 HiRDB/Single Server Version 8 08-00

P-2862-7387 HiRDB/Parallel Server Version 8 08-00

P-2862-1187 HiRDB/Run Time Version 8 08-00

P-2862-1287 HiRDB/Developer's Kit Version 8 08-00

P-2862-7H87 HiRDB Non Recover Front End Server Version 8 08-00

P-2862-7J87 HiRDB Advanced High Availability Version 8 08-00

P-2862-7K87 HiRDB Advanced Partitioning Option Version 8 08-00

For the Windows 2000, Windows XP, Windows XP x64 Edition, Windows Server 2003, or Windows Server 2003 x64 Edition operating system:

P-2662-1187 HiRDB/Run Time Version 8 08-00

P-2662-1287 HiRDB/Developer's Kit Version 8 08-00

This edition of the manual is released for the preceding program products, which have been developed under a quality management system that has been certified to comply with ISO9001 and TickIT. This manual may also apply to other program products; for details, see *Before Installing* or *Readme file*.

■ Trademarks

ActiveX is a trademark of Microsoft Corp. in the U.S. and other countries.

AIX is a registered trademark of the International Business Machines Corp. in the U.S.

CORBA is a registered trademark of Object Management Group, Inc. in the United States.

DataStage, MetaBroker, MetaStage and QualityStage are trademarks of International Business Machines Corporation in the United States, other countries, or both.

DB2 is a registered trademark of the International Business Machines Corp. in the U.S.

HACMP/6000 is a trademark of the International Business Machines Corp. in the U.S.

HP-UX is a product name of Hewlett-Packard Company.

IBM is a registered trademark of the International Business Machines Corp. in the U.S.

Itanium is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

JBuilder is a trademark of Borland Software Corporation in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Lotus, 1-2-3 are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

Microsoft Excel is a product name of Microsoft Corp.
Microsoft is a registered trademark of Microsoft Corp. in the U.S. and other countries.
Motif is a registered trademark of the Open Software Foundation, Inc.
MS-DOS is a registered trademark of Microsoft Corp. in the U.S. and 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 is a registered trademark of Oracle Corporation.
Oracle8i is a trademark of ORACLE Corporation.
Oracle9i is a trademark of ORACLE Corporation.
Oracle 10g is a trademark of ORACLE Corporation.
OS/390 is a trademark of the International Business Machines Corp. in the U.S.
POSIX stands for Portable Operating System Interface for Computer Environment, which is a set of standard specifications published by the Institute of Electrical and Electronics Engineers, Inc.
RISC System/6000 is a registered trademark of the International Business Machines Corp. in the U.S.
Solaris is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.
Sun is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.
Sun Microsystems is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.
The right to use the trademark DCE in Japan is sub-licensed from OSF.
UNIFY2000 is a product name of Unify Corp.
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 a registered trademark of Microsoft Corp. in the U.S. and other countries.
Visual C++ is a registered trademark of Microsoft Corp. in the U.S. and other countries.
Visual Studio is a registered trademark of Microsoft Corp. in the U.S. and other countries.
WebLogic is a registered trademark of BEA Systems, Inc.
Windows is a registered trademark of Microsoft Corp. in the U.S. and other countries.
Windows NT is a registered trademark of Microsoft Corp. in the U.S. and other countries.
Windows Server is a registered trademark of Microsoft Corp. in the U.S. and other countries.
X/Open is a registered trademark of X/Open Company Limited in the U.K. and other countries.
X Window System is a trademark of X Consortium, Inc.

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.

■ Edition history

Edition 1 (3020-6-352(E)): March 2007

■ Copyright

All Rights Reserved. Copyright (C) 2007, Hitachi, Ltd.

Preface

This manual describes how to construct a HiRDB Version 8 scalable database server, how to design the system and databases, and how to create the databases.

Intended readers

This manual is intended for users who will construct and/or operate a relational database system using HiRDB Version 8 (hereafter referred to as HiRDB).

The manual assumes that you have the following:

- A basic knowledge of managing Windows systems
- A basic knowledge of SQL

Please read the *HiRDB Version 8 Description* manual before reading this manual.

Organization of this manual

This manual consists of the following 23 chapters and 3 appendixes:

Chapter 1. *Overview of HiRDB System Construction*

Describes the HiRDB system construction procedure, organization of the HiRDB files, and upgrading procedure.

Chapter 2. *Installation*

Describes the preparations required before installation, the HiRDB installation procedure, and uninstallation of HiRDB. The chapter also provides notes on installation of option program products.

Chapter 3. *Setting Up an Environment Using the Simple Setup Tool*

Describes use of the simple setup tool for environment setup for a HiRDB system.

Chapter 4. *Setting Up an Environment Using Commands*

Describes the procedure for setting up the HiRDB environment using commands.

Chapter 5. *Setting Up the Plug-in Environment*

Describes the procedures for setting up the plug-in environment, along with the procedures for upgrading and deleting (uninstalling) it.

Chapter 6. *Creating Databases*

Describes the procedures from schema, table, and index creation through data storage.

Chapter 7. *Linking to Other Products*

Describes how to link HiRDB to other products.

Chapter 8. *Designing a HiRDB/Single Server*

Describes the system configuration of a HiRDB/Single Server, the procedure for designing its HiRDB file system areas and system files, and provides notes about placement of RDAREAs.

Chapter 9. *Designing a HiRDB/Parallel Server*

Describes the system configuration of a HiRDB/Parallel Server, the procedure for designing its HiRDB file system areas and system files, and provides notes about placement of RDAREAs.

Chapter 10. *Designing a Multi-HiRDB*

Describes the design of a multi-HiRDB.

Chapter 11. *Designing Global Buffers and Local Buffers*

Describes global-buffer and local-buffer design.

Chapter 12. *Designing Tables*

Describes table design.

Chapter 13. *Designing Indexes*

Describes the design of indexes and plug-in indexes.

Chapter 14. *Designing RDAREAs*

Describes the design of the segments and pages that constitute an RDAREA.

Chapter 15. *Storage Requirements for HiRDB*

Explains how to determine the storage requirements for a HiRDB/Single Server and a HiRDB/Parallel Server.

Chapter 16. *Determining RDAREA Size*

Explains how to determine the size of each type of RDAREA.

Chapter 17. *Determining System File Size*

Explains how to determine the sizes of system files, such as system log files, synchronization point dump files, and status files.

Chapter 18. *Determining Work Table File Size*

Explains how to determine the size of a work-table file.

Chapter 19. *Storage Requirements for Utility Execution*

Explains how to determine the file sizes and storage requirements for executing

utilities.

Chapter 20. *Determining Environment Variables Related to the Number of Resources*

Explains how to determine the environment variables related to the number of resources.

Chapter 21. *Windows Registry Settings*

Explains how to determine the Windows registry settings.

Chapter 22. *Sample Files*

Describes the sample files provided with HiRDB (sample database, configuration, and UOC).

Chapter 23. *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 a firewall.

Appendix A. *HiRDB Maximum and Minimum Values*

Lists the maximum and minimum values for HiRDB system configuration.

Appendix B. *Questions and Answers*

Provides important information concerning the construction, design, and operation of a HiRDB system in question-and-answer format.

Appendix C. *Setting Up an Environment Using a Batch File*

Describes the procedure for setting up the HiRDB server environment using a batch file, which automatically sets up a sample environment.

Related publications

This manual is related to the following manuals, which should be read as required.

HiRDB (for Windows)

- *For Windows Systems HiRDB Version 8 Description* (3020-6-351(E))
- *For Windows Systems HiRDB Version 8 System Definition* (3020-6-353(E))
- *For Windows Systems HiRDB Version 8 System Operation Guide* (3020-6-354(E))
- *For Windows Systems HiRDB Version 8 Command Reference* (3020-6-355(E))

HiRDB (for UNIX)

- *For UNIX Systems HiRDB Version 8 Description* (3000-6-351(E))
- *For UNIX Systems HiRDB Version 8 Installation and Design Guide* (3000-6-352(E))

- *For UNIX Systems HiRDB Version 8 System Definition (3000-6-353(E))*
- *For UNIX Systems HiRDB Version 8 System Operation Guide (3000-6-354(E))*
- *For UNIX Systems HiRDB Version 8 Command Reference (3000-6-355(E))*
- *HiRDB Staticizer Option Version 7 Description and User's Guide (3000-6-282(E))*
- *For UNIX Systems HiRDB Version 8 Disaster Recovery System Configuration and Operation Guide (3000-6-364)**

HiRDB (for both Windows and UNIX)

- *HiRDB Version 8 UAP Development Guide (3020-6-356(E))*
- *HiRDB Version 8 SQL Reference (3020-6-357(E))*
- *HiRDB Version 8 Messages (3020-6-358(E))*
- *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide (3020-6-360(E))*
- *HiRDB Dataextractor Version 8 Description, User's Guide and Operator's Guide (3020-6-362(E))*

* This manual has been published in Japanese only; it is not available in English.

You must use the UNIX or the Windows manuals, as appropriate to the platform you are using.

Others

- *HiRDB External Data Access Version 7 Description and User's Guide (3000-6-284(E))*
- *OpenTPI Version 7 Programming Guide (3000-3-D51(E))*
- *OpenTPI Version 7 System Definition (3000-3-D52(E))*
- *TPI/LINK USER'S GUIDE (3000-3-951(E))*
- *Job Management Partner I/Integrated Management - Manager System Configuration and User's Guide (3020-3-K01(E))*
- *Job Management Partner I/Base User's Guide (3020-3-K06(E))*
- *Job Management Partner I/Automatic Job Management System 2 Description (3020-3-K21(E))*
- *Job Management Partner I/Integrated Manager - Console (3020-3-F01(E))*
- *Job Management Partner I/Base (3020-3-F04(E))*
- *Job Management Partner I/Automatic Job Management System 2 Description*

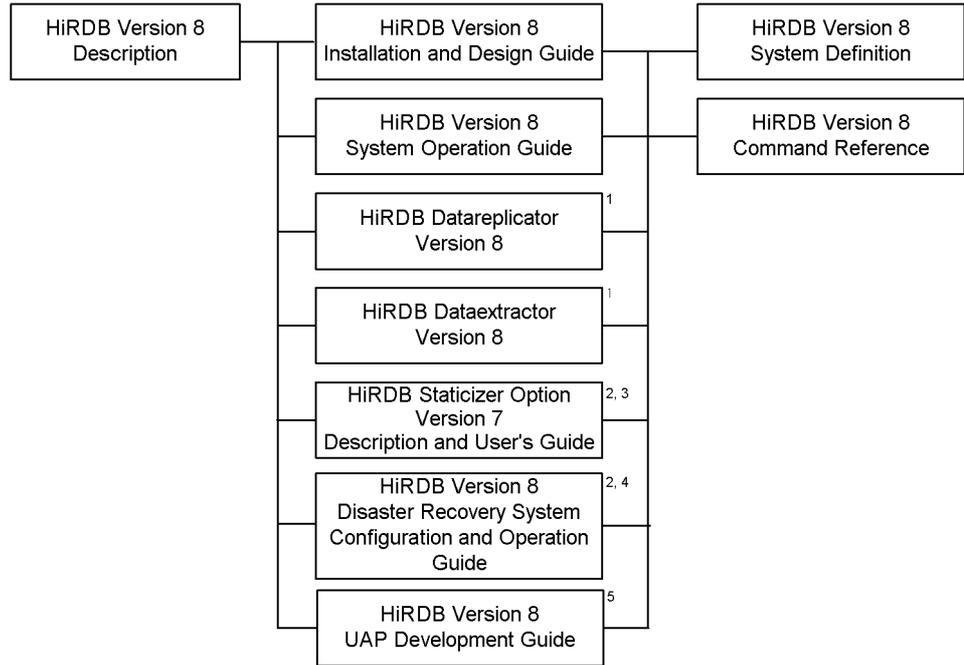
(3020-3-F06(E))

- *JP1 V6 JP1/Automatic Job Management System 2 User's Guide* (3020-3-980(E))
- *JP1 V6 JP1/Base* (3020-3-986(E))

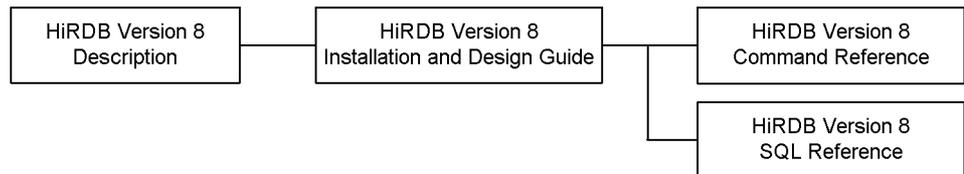
Organization of HiRDB manuals

The HiRDB manuals are organized as shown below. For the most efficient use of these manuals, it is suggested that they be read in the order they are shown, going from left to right.

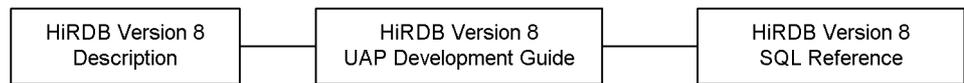
Manuals for system administrators:



Manuals for users who create tables:



Manuals for users who create or execute UAPs:



¹ Read if you use the replication facility to link data.

² Published for UNIX only. There is no corresponding Windows manual.

³ Read if you use the inner replica facility.

⁴ Read if you are configuring a disaster recovery system.

⁵ Must be read if you are linking HiRDB to an OLTP system.

Conventions: Abbreviations

Unless otherwise required, this manual uses the following abbreviations for product and other names.

Name of product or other entity	Representation	
HiRDB/Single Server Version 8	HiRDB/Single Server	HiRDB or HiRDB Server
HiRDB/Single Server Version 8(64)		
HiRDB/Parallel Server Version 8	HiRDB/Parallel Server	
HiRDB/Parallel Server Version 8(64)		
HiRDB/Developer's Kit Version 8	HiRDB/ Developer's Kit	HiRDB Client
HiRDB/Developer's Kit Version 8(64)		
HiRDB/Run Time Version 8	HiRDB/Run Time	
HiRDB/Run Time Version 8(64)		
HiRDB Datareplicator Version 8	HiRDB Datareplicator	
HiRDB Dataextractor Version 8	HiRDB Dataextractor	
HiRDB Text Search Plug-in Version 7	HiRDB Text Search Plug-in	
HiRDB Spatial Search Plug-in Version 3	HiRDB Spatial Search Plug-in	
HiRDB Staticizer Option Version 8	HiRDB Staticizer Option	
HiRDB LDAP Option Version 8	HiRDB LDAP Option	
HiRDB Advanced Partitioning Option Version 8	HiRDB Advanced Partitioning Option	
HiRDB Advanced High Availability Version 8	HiRDB Advanced High Availability	
HiRDB Non Recover Front End Server Version 8	HiRDB Non Recover FES	
HiRDB Disaster Recovery Light Edition Version 8	HiRDB Disaster Recovery Light Edition	
HiRDB External Data Access Version 8	HiRDB External Data Access	
HiRDB External Data Access Adapter Version 8	HiRDB External Data Access Adapter	
HiRDB Adapter for XML - Standard Edition	HiRDB Adapter for XML	
HiRDB Adapter for XML - Enterprise Edition		
HiRDB Control Manager	HiRDB CM	
HiRDB Control Manager Agent	HiRDB CM Agent	

Name of product or other entity	Representation	
Hitachi TrueCopy	TrueCopy	
Hitachi TrueCopy basic		
TrueCopy		
TrueCopy remote replicator		
JP1/Automatic Job Management System 2	JP1/AJS2	
JP1/Automatic Job Management System 2 - Scenario Operation	JP1/AJS2-SO	
JP1/Cm2/Extensible SNMP Agent	JP1/ESA	
JP1/Cm2/Extensible SNMP Agent for Mib Runtime		
JP1/Cm2/Network Node Manager	JP1/NNM	
JP1/Integrated Management - Manager	JP1/Integrated Management or JP1/IM	
JP1/Integrated Management - View		
JP1/Magnetic Tape Access	EasyMT	
EasyMT		
JP1/Magnetic Tape Library	MTguide	
JP1/NETM/DM	JP1/NETM/DM	
JP1/NETM/DM Manager		
JP1/Performance Management	JP1/PFM	
JP1/Performance Management Agent for HiRDB	JP1/PFM-Agent for HiRDB	
JP1/Performance Management - Agent for Platform	JP1/PFM-Agent for Platform	
JP1/Performance Management/SNMP System Observer	JP1/SSO	
JP1/VERITAS NetBackup BS v4.5	NetBackup	
JP1/VERITAS NetBackup v4.5		
JP1/VERITAS NetBackup BS V4.5 Agent for HiRDB License	JP1/VERITAS NetBackup Agent for HiRDB License	
JP1/VERITAS NetBackup V4.5 Agent for HiRDB License		
JP1/VERITAS NetBackup 5 Agent for HiRDB License		
OpenTP1/Server Base Enterprise Option	TP1/EE	
Virtual-storage Operating System 3/Forefront System Product	VOS3/FS	VOS3

Name of product or other entity	Representation		
Virtual-storage Operating System 3/Leading System Product	VOS3/LS		
Extensible Data Manager/Base Extended Version 2 XDM basic program XDM/BASE E2	XDM/BASE E2		
XDM/Data Communication and Control Manager 3 XDM Data communication control XDM/DCCM3	XDM/DCCM3		
XDM/Relational Database XDM/RD	XDM/RD	XDM/RD	
XDM/Relational Database Extended Version 2 XDM/RD E2	XDM/RD E2		
VOS3 Database Connection Server	DB Connection Server		
DB2 Universal Database for OS/390 Version 6	DB2		
DNCWARE ClusterPerfect (Linux Version)	ClusterPerfect		
Microsoft _(R) Excel	Microsoft Excel or Excel		
Microsoft _(R) Visual C++ _(R)	Visual C++ or C++		
Oracle 8i	ORACLE		
Oracle 9i			
Oracle 10g			
Sun Java TM System Directory Server	Sun Java System Directory Server or Directory Server		
HP-UX 11i V2 (IPF)	HP-UX or HP-UX (IPF)		
Red Hat Linux	Linux		
Red Hat Enterprise Linux			
Red Hat Enterprise Linux AS 3 (IPF)	Linux (IPF)	Linux	
Red Hat Enterprise Linux AS 4 (IPF)			
Red Hat Enterprise Linux AS 3 (AMD64 & Intel EM64T)	Linux (EM64T)		
Red Hat Enterprise Linux AS 4 (AMD64 & Intel EM64T)			
Red Hat Enterprise Linux ES 4 (AMD64 & Intel EM64T)			
turbolinux 7 Server for AP8000	Linux for AP8000		

Name of product or other entity	Representation	
Microsoft _(R) Windows NT _(R) Workstation Operating System Version 4.0	Windows NT	
Microsoft _(R) Windows NT _(R) Server Network Operating System Version 4.0		
Microsoft _(R) Windows _(R) 2000 Professional Operating System	Windows 2000	
Microsoft _(R) Windows _(R) 2000 Server Operating System		
Microsoft _(R) Windows _(R) 2000 Datacenter Server Operating System		
Microsoft _(R) Windows _(R) 2000 Advanced Server Operating System	Windows 2000 or Windows 2000 Advanced Server	
Microsoft _(R) Windows Server TM 2003, Standard Edition	Windows Server 2003	
Microsoft _(R) Windows Server TM 2003, Enterprise Edition		
Microsoft _(R) Windows Server TM 2003 R2, Standard Edition	Windows Server 2003 R2 or Windows Server 2003	
Microsoft _(R) Windows Server TM 2003 R2, Enterprise Edition		
64 bit Version Microsoft _(R) Windows Server TM 2003, Enterprise Edition (IPF)	Windows Server 2003 (IPF) or Windows Server 2003	
Microsoft _(R) Windows Server TM 2003, Standard x64 Edition	Windows Server 2003 or Windows Server 2003 x64 Editions	Windows (x64)
Microsoft _(R) Windows Server TM 2003, Enterprise x64 Edition		
Microsoft _(R) Windows Server TM 2003 R2, Standard x64 Edition	Windows Server 2003, Windows Server 2003 R2 or Windows Server 2003 x64 Editions	
Microsoft _(R) Windows Server TM 2003 R2, Enterprise x64 Edition		
Microsoft _(R) Windows _(R) XP Professional x64 Edition	Windows XP or Windows XP x64 Edition	
Microsoft _(R) Windows _(R) XP Professional Operating System	Windows XP Professional	

Name of product or other entity	Representation	
Microsoft _(R) Windows _(R) XP Home Edition Operating System	Windows XP Home Edition	
Single server	SDS	
System manager	MGR	
Front-end server	FES	
Dictionary server	DS	
Back-end server	BES	

- Windows 2000, Windows XP, and Windows Server 2003 may be referred to collectively as *Windows*.
- The HiRDB directory path is represented as %PDDIR%. The path of the Windows installation directory is represented as %windir%.
- The hosts file means the `hosts` file stipulated by TCP/IP. As a rule, a reference to the hosts file means the `%windir%\system32\drivers\etc\hosts` file.

This manual also uses the following abbreviations:

Abbreviation	Full name or meaning
ACK	Acknowledgement
ADM	Adaptable Data Manager
ADO	ActiveX Data Objects
ADT	Abstract Data Type
AP	Application Program
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
BES	Back End Server
BLOB	Binary Large Object
BOM	Byte Order Mark
CD-ROM	Compact Disc - Read Only Memory
CGI	Common Gateway Interface
CLOB	Character Large Object

Abbreviation	Full name or meaning
CMT	Cassette Magnetic Tape
COBOL	Common Business Oriented Language
CORBA(R)	Common ORB Architecture
CPU	Central Processing Unit
CSV	Comma Separated Values
DAO	Data Access Object
DAT	Digital Audio Taperecorder
DB	Database
DBM	Database Module
DBMS	Database Management System
DDL	Data Definition Language
DF for Windows NT	Distributing Facility for Windows NT
DF/UX	Distributing Facility/for UNIX
DIC	Dictionary Server
DLT	Digital Linear Tape
DML	Data Manipulate Language
DNS	Domain Name System
DOM	Document Object Model
DS	Dictionary Server
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DWH	Data Warehouse
EUC	Extended UNIX Code
EX	Exclusive
FAT	File Allocation Table
FD	Floppy Disk
FES	Front End Server

Abbreviation	Full name or meaning
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GUI	Graphical User Interface
HBA	Host Bus Adapter
HD	Hard Disk
HTML	Hyper Text Markup Language
ID	Identification number
IP	Internet Protocol
IPF	Itanium(R) Processor Family
JAR	Java Archive File
Java VM	Java Virtual Machine
JDBC	Java Database Connectivity
JDK	Java Developer's Kit
JFS	Journaled File System
JFS2	Enhanced Journaled File System
JIS	Japanese Industrial Standard code
JP1	Job Management Partner 1
JRE	Java Runtime Environment
JTA	Java Transaction API
JTS	Java Transaction Service
KEIS	Kanji processing Extended Information System
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LIP	loop initialization process
LOB	Large Object
LRU	Least Recently Used
LTO	Linear Tape-Open

Abbreviation	Full name or meaning
LU	Logical Unit
LUN	Logical Unit Number
LVM	Logical Volume Manager
MGR	System Manager
MIB	Management Information Base
MRCF	Multiple RAID Coupling Feature
MSCS	Microsoft Cluster Server
NAFO	Network Adapter Fail Over
NAPT	Network Address Port Translation
NAT	Network Address Translation
NIC	Network Interface Card
NIS	Network Information Service
NTFS	New Technology File System
ODBC	Open Database Connectivity
OLAP	Online Analytical Processing
OLE	Object Linking and Embedding
OLTP	On-Line Transaction Processing
OOCOBOL	Object Oriented COBOL
ORB	Object Request Broker
OS	Operating System
OSI	Open Systems Interconnection
OTS	Object Transaction Service
PC	Personal Computer
PDM II E2	Practical Data Manager II Extended Version 2
PIC	Plug-in Code
PNM	Public Network Management
POSIX	Portable Operating System Interface for UNIX

Abbreviation	Full name or meaning
PP	Program Product
PR	Protected Retrieve
PU	Protected Update
RAID	Redundant Arrays of Inexpensive Disk
RD	Relational Database
RDB	Relational Database
RDB1	Relational Database Manager 1
RDB1 E2	Relational Database Manager 1 Extended Version 2
RDO	Remote Data Objects
RiSe	Real time SAN replication
RM	Resource Manager
RMM	Resource Manager Monitor
RPC	Remote Procedure Call
SAX	Simple API for XML
SDS	Single Database Server
SGML	Standard Generalized Markup Language
SJIS	Shift JIS
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SQL/K	Structured Query Language / VOS K
SR	Shared Retrieve
SU	Shared Update
TCP/IP	Transmission Control Protocol / Internet Protocol
TM	Transaction Manager
TMS-4V/SP	Transaction Management System - 4V / System Product
UAP	User Application Program
UOC	User Own Coding

Abbreviation	Full name or meaning
VOS K	Virtual-storage Operating System Kindness
VOS1	Virtual-storage Operating System 1
VOS3	Virtual-storage Operating System 3
WS	Workstation
WWW	World Wide Web
XDM/BASE E2	Extensible Data Manager / Base Extended Version 2
XDM/DF	Extensible Data Manager / Distributing Facility
XDM/DS	Extensible Data Manager / Data Spreader
XDM/RD E2	Extensible Data Manager / Relational Database Extended Version 2
XDM/SD E2	Extensible Data Manager / Structured Database Extended Version 2
XDM/XT	Extensible Data Manager / Data Extract
XFIT	Extended File Transmission program
XML	Extensible Markup Language

Log representations

The application log that is displayed by Windows Event Viewer is referred to as the *event log*. The following procedure is used to view the event log.

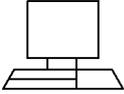
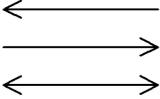
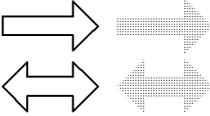
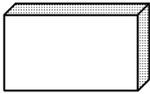
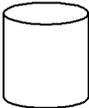
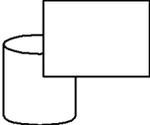
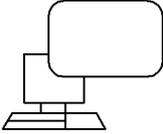
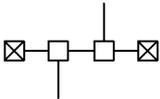
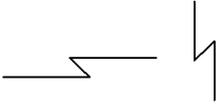
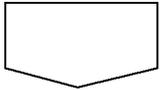
To view the event log:

1. Choose **Start, Programs, Administrative Tools (Common)**, and then **Event Viewer**.
2. Choose **Log**, and then **Application**.
3. The application log is displayed. Messages with **HiRDBSingleServer** or **HiRDBParallelServer** displayed in the **Source** column were issued by HiRDB.

If you specified a setup identifier when you installed HiRDB, the specified setup identifier follows **HiRDBSingleServer** or **HiRDBParallelServer**.

Conventions: Diagrams

This manual uses the following conventions in diagrams:

• Personal computer or workstation	• I/O operation	• Flow of control	• Flow of data
			
• Program or server ¹	• File or magnetic disk ²	• File contents	• Screen display
			
• Network	• LAN	• Communication line	• Flow of procedure or work item
			

¹ In some figures, a program is simply enclosed in a rectangle (without shading).

² Input-data files, unload files, and backup files can be stored on magnetic tape, magnetic cassette tape (CMT), and digital audio tape (DAT), as well as on magnetic disk; only magnetic disk storage is described in this manual.

Conventions: Fonts and symbols

Font and symbol conventions are classified as:

- General font conventions
- Conventions in syntax explanations

These conventions are described below.

General font conventions

The following table lists the general font conventions:

Font	Convention
Bold	Bold type indicates text on a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example, bold is used in sentences such as the following: <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.
<i>Italics</i>	Italics are used to indicate a placeholder for some actual text provided by the user or system. Italics are also used for emphasis. For example: <ul style="list-style-type: none"> • Write the command as follows: <i>copy source-file target-file</i> • Do <i>not</i> delete the configuration file.
Code font	A code font indicates text that the user enters without change, or text (such as messages) output by the system. For example: <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>

Examples of coding and messages appear as follows (although there may be some exceptions, such as when coding is included in a diagram):

```
MakeDatabase
...
StoreDatabase temp DB32
```

In examples of coding, an ellipsis (...) indicates that one or more lines of coding are not shown for purposes of brevity.

Conventions in syntax explanations

Syntax definitions appear as follows:

```
StoreDatabase [temp|perm] (database-name ...)
```

The following table lists the conventions used in syntax explanations:

Example font or symbol	Convention
<code>StoreDatabase</code>	Code-font characters must be entered exactly as shown.
<i>database-name</i>	This font style marks a placeholder that indicates where appropriate characters are to be entered in an actual command.
SD	Bold code-font characters indicate the abbreviation for a command.
<u>perm</u>	Underlined characters indicate the default value.
[]	Square brackets enclose an item or set of items whose specification is optional.

Example font or symbol	Convention
	Only one of the options separated by a vertical bar can be specified at the same time.
...	An ellipsis (...) indicates that the item or items enclosed in () or [] immediately preceding the ellipsis may be specified as many times as necessary.
()	Parentheses indicate the range of items to which the vertical bar () or ellipsis (...) is applicable.

Notations used in formulas

The following notations are used in formulas:

Notation	Explanation
↑ ↑	Round up the result to the next integer. Example: The result of $\uparrow 34 \div 3 \uparrow$ is 12.
↓ ↓	Discard digits following the decimal point. Example: The result of $\downarrow 34 \div 3 \downarrow$ is 11.
MAX	Select the largest value as the result. Example: The result of MAX (3 × 6, 4 + 7) is 18.
MIN	Select the smallest value as the result. Example: The result of MIN (3 × 6, 4 + 7) is 11.
mod	mod (a, b) indicates the remainder of a divided by b. Example: The result of mod (9, 2) is 1.

Notes on Windows path names

- In this manual, the Windows terms *directory* and *folder* are both referred to as *directory*.
- Include the drive name when you specify an absolute path name.

Example: C:\win32app\hitachi\hirdb_s\spool\tmp

- When you specify a path name in a command argument, in a control statement file, or in a HiRDB system definition file, and that path name includes a space or a parenthesis, you must enclose the entire path name in double quotation marks (").

Example: pdinit -d "C:\Program Files(x86)\hitachi\hirdb_s\conf\mkinit"

However, double quotation marks are not necessary when you use the `set` command in a batch file or at the command prompt to set an environment variable or when you

specify the installation directory. If you do use double quotation marks in such a case, the double quotation marks become part of the value assigned to the environment variable.

Example: `set PDCLTPATH=C:\Program Files\hitachi\hirdb_s\spool`

- HiRDB cannot use files on a networked drive, so you must install HiRDB and configure the HiRDB environment on a local drive. Files used by utilities, such as utility input and output files, must also be on the local drive.

Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is 1,024² bytes.
- 1 GB (gigabyte) is 1,024³ bytes.
- 1 TB (terabyte) is 1,024⁴ bytes.

Conventions: Version numbers

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

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

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

Important notes on this manual

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

- Distributed database facility
- Server mode system switchover facility
- User server hot standby
- Rapid system switchover facility
- Standby-less system switchover (1:1) facility
- Standby-less system switchover (effects distributed) facility
- HiRDB External Data Access facility

- Inner replica facility (when described for the Windows version of HiRDB)
- Updatable online reorganization (when described for the Windows version of HiRDB)
- Sun Java System Directory Server linkage facility
- Simple setup tool

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

- HiRDB Control Manager
- HiRDB Disaster Recovery Light Edition
- HiRDB External Data Access
- HiRDB LDAP Option

Contents

Preface	i
Intended readers	i
Organization of this manual	i
Related publications	iii
Organization of HiRDB manuals	v
Conventions: Abbreviations	vi
Log representations	xvi
Conventions: Diagrams	xvii
Conventions: Fonts and symbols	xvii
Notes on Windows path names	xix
Conventions: KB, MB, GB, and TB	xx
Conventions: Version numbers	xx
Important notes on this manual	xx
1. Overview of HiRDB System Construction	1
1.1 System construction procedures	2
1.1.1 System construction procedure for installing a new HiRDB	2
1.1.2 Setting up a HiRDB environment	2
1.1.3 Environment setup for linking to other products	4
1.1.4 Using Windows Terminal Service	5
1.2 Organization of HiRDB directories and files	7
1.3 Upgrading HiRDB	12
1.3.1 Before upgrading	12
1.3.2 Replacing an existing version with the new version	16
1.3.3 Upgrading the HiRDB plug-ins	18
1.3.4 Using Java stored procedures and functions	18
1.3.5 In the event of an upgrading error	19
1.3.6 Restoring HiRDB to the previous version when version upgrade fails	19
1.4 Updating to HiRDB update version	22
1.4.1 Updating HiRDB	22
1.4.2 Prerequisites	23
1.4.3 Actions performed prior to updating	24
1.4.4 Update procedure	24
1.4.5 Update procedure when the system switchover facility is used	27
1.4.6 Cautions	28
1.4.7 Operation considerations	30
1.4.8 Related product limitations and considerations	31
1.4.9 Operation when an error occurs during installation	31

2. Installation	35
2.1 Pre-installation procedure	36
2.1.1 Checking the server machine environment	36
2.1.2 Registering the HiRDB administrator	40
2.1.3 Setting up the OS environment files.....	40
2.2 HiRDB installation procedure	42
2.2.1 Notes before installation.....	42
2.2.2 HiRDB installation procedure	42
2.2.3 Environment variables set after installation	44
2.2.4 Notes after installation.....	45
2.2.5 What to do if error 1073 occurs during installation.....	45
2.3 Post-installation procedures.....	47
2.3.1 Selection of the character codes	47
2.3.2 Deleting files from the HiRDB directory	47
2.4 Notes about option program products installation	49
2.5 HiRDB uninstallation procedure	51
3. Setting Up an Environment Using the Simple Setup Tool	55
3.1 Overview of the simple setup tool	56
4. Setting Up an Environment Using Commands	57
4.1 Overview of environment setup using commands.....	58
4.2 Creating the HiRDB system definitions	60
4.2.1 Creating HiRDB system definitions (HiRDB/Single Server)	60
4.2.2 Creating HiRDB system definitions (HiRDB/Parallel Server)	62
4.2.3 Modifying HiRDB system definitions (excluding UAP environment definitions)	69
4.2.4 Modifying a UAP environment definition.....	71
4.3 Creating HiRDB file system areas.....	72
4.3.1 Types of HiRDB file system areas.....	72
4.3.2 Creating a HiRDB file system area using the raw I/O facility	73
4.3.3 Example 1 (creating a HiRDB file system area for RDAREAs).....	74
4.3.4 Example 2 (creating a HiRDB file system area for system files).....	75
4.3.5 Example 3 (creating a HiRDB file system area for work table files).....	76
4.3.6 Example 4 (creating a HiRDB file system area for utilities).....	77
4.3.7 Example 5 (creating a HiRDB file system area for list RDAREAs).....	78
4.3.8 Example 6 (creating a HiRDB file system area that uses the raw I/O facility)	78
4.4 Creating system files.....	80
4.4.1 Creating system log files	80
4.4.2 Creating synchronization point dump files.....	81
4.4.3 Creating status files	82
4.4.4 Example of system file creation (HiRDB/Single Server).....	82

4.4.5 Example of system file creation (HiRDB/Parallel Server)	86
4.5 Creating system RDAREAs	96
4.5.1 Basics	96
4.5.2 Example 1 (HiRDB/Single Server)	97
4.5.3 Example 2 (HiRDB/Parallel Server)	98
4.6 Starting HiRDB for the first time	100
4.7 Creating user RDAREAs	101
4.7.1 Basics	101
4.7.2 Example 1 (HiRDB/Single Server)	101
4.7.3 Example 2 (HiRDB/Parallel Server)	102
4.8 Creating user LOB RDAREAs	105
4.8.1 Basics	105
4.8.2 Example 1 (HiRDB/Single Server)	105
4.8.3 Example 2 (HiRDB/Parallel Server)	107
4.9 Creating data dictionary LOB RDAREAs	109
4.9.1 Basics	109
4.9.2 Example 1 (HiRDB/Single Server)	109
4.9.3 Example 2 (HiRDB/Parallel Server)	111
4.10 Creating list RDAREAs	113
4.10.1 Basics	113
4.10.2 Example 1 (HiRDB/Single Server)	113
4.10.3 Example 2 (HiRDB/Parallel Server)	114
5. Setting Up the Plug-in Environment	117
5.1 Overview of plug-in environment setup	118
5.1.1 Environment setup procedure	118
5.1.2 Notes on using plug-ins	123
5.2 Upgrading plug-ins	125
5.2.1 Notes about upgrading	125
5.2.2 Upgrading procedure	125
5.3 Deleting plug-ins	127
6. Creating Databases	129
6.1 Overview of database creation	130
6.1.1 Preparing for database creation	130
6.1.2 Database creation procedure	131
6.1.3 Database update log acquisition mode	132
6.1.4 Notes on data storage for a table for which an index with the unique attribute has been defined	136
6.1.5 Loading a large amount of data (data loading with the synchronization point specification)	136
6.1.6 Loading data into a row-partitioned table (Creating divided-input data files)	137
6.1.7 Input data file UOC	137

6.1.8	Deleting unneeded RDAREAs	137
6.2	Creating a row partitioned table.....	138
6.3	Creating a table with a LOB column	142
6.4	Creating a table containing a plug-in-provided abstract data type	146
6.5	Creating a table containing a user-defined abstract data type.....	151
6.5.1	Defining an abstract data type	151
6.5.2	Defining a table	155
6.5.3	Defining an index	157
6.5.4	Storing data in a table	157
6.5.5	Database update log acquisition methods.....	158
6.5.6	Checking the data storage status.....	160
6.6	Handling errors during batch index creation	161
6.6.1	When data was loaded in log acquisition mode or pre-update log acquisition mode.....	161
6.6.2	When data was loaded in no-log mode.....	164
6.7	Handling utility abnormal termination errors during data loading with the synchronization point specification.....	167
6.7.1	Overview of error handling procedure	167
6.7.2	Example	168
7.	Linking to Other Products	171
7.1	Linking to the replication facility	172
7.1.1	Linking to HiRDB Datareplicator	172
7.1.2	Linking to HiRDB Dataextractor	173
7.2	Linking with an OLTP system	174
7.2.1	OLTP products supported for linking.....	174
7.2.2	HiRDB XA library.....	174
7.2.3	Example of HiRDB system configuration with OLTP linkage	176
7.2.4	Transaction transfer	180
7.2.5	Registering HiRDB in the transaction manager	183
7.2.6	Information to be registered in the transaction manager	185
7.2.7	Example of registering in the transaction manager	190
7.2.8	Modifying the registration information in the transaction manager.....	192
7.2.9	Methods for re-establishing connection between the transaction manager and HiRDB.....	193
7.2.10	Notes.....	195
7.3	Linking to the HiRDB External Data Access facility.....	197
7.4	Linking to JP1	199
7.4.1	Reporting events to JP1/Base	199
7.4.2	Managing events by JP1/IM	201
7.4.3	Automatic job execution using JP1/AJS2 linkage.....	203
8.	Designing a HiRDB/Single Server	205
8.1	System design for a HiRDB/Single Server.....	206

8.1.1	System design.....	206
8.1.2	System configuration.....	206
8.2	Designing HiRDB file system areas.....	207
8.2.1	Designing HiRDB file system areas for RDAREAs.....	207
8.2.2	Designing HiRDB file system areas for system files.....	208
8.2.3	Designing HiRDB file system areas for work table files.....	208
8.2.4	Designing HiRDB file system areas for utilities.....	209
8.2.5	Designing HiRDB file system areas for list RDAREAs.....	210
8.2.6	Maximum sizes of HiRDB file system areas.....	210
8.2.7	Notes on the use of the system switchover facility.....	210
8.3	Designing system files.....	212
8.3.1	Designing system log files.....	212
8.3.2	Designing synchronization point dump files.....	216
8.3.3	Designing status files.....	219
8.4	Placing RDAREAs.....	225
8.4.1	Placing system RDAREAs.....	225
8.4.2	Placing data dictionary LOB RDAREAs.....	226
8.4.3	Placing user RDAREAs.....	227
8.4.4	Placing user LOB RDAREAs.....	227
8.4.5	Placing list RDAREAs.....	228

9. Designing a HiRDB/Parallel Server 229

9.1	System design for a HiRDB/Parallel Server.....	230
9.1.1	System design.....	230
9.1.2	System configuration of HiRDB/Parallel Server.....	231
9.1.3	Setting up multiple front-end servers.....	232
9.1.4	Recovery-unnecessary front-end server.....	236
9.2	Designing HiRDB file system areas.....	241
9.2.1	Designing HiRDB file system areas for RDAREAs.....	241
9.2.2	Designing HiRDB file system areas for system files.....	242
9.2.3	Designing HiRDB file system areas for work table files.....	242
9.2.4	Designing HiRDB file system areas for utilities.....	243
9.2.5	Designing HiRDB file system areas for list RDAREAs.....	244
9.2.6	Maximum sizes of HiRDB file system areas.....	245
9.2.7	Notes on the use of the system switchover facility.....	245
9.3	Designing system files.....	246
9.3.1	Designing system log files.....	246
9.3.2	Designing synchronization point dump files.....	250
9.3.3	Designing status files.....	253
9.4	Placing RDAREAs.....	259
9.4.1	Placing system RDAREAs.....	259
9.4.2	Placing data dictionary LOB RDAREAs.....	260
9.4.3	Placing user RDAREAs.....	261
9.4.4	Placing user LOB RDAREAs.....	262

9.4.5 Placing list RDAREAs	263
9.5 Heterogeneous system configuration for HiRDB/Parallel Server	264
10. Designing a Multi-HiRDB	267
10.1 System design for a multi-HiRDB	268
10.1.1 Installing a multi-HiRDB	268
10.1.2 Setting the environment for a multi-HiRDB	270
10.2 Notes about operation	271
11. Designing Global Buffers and Local Buffers	273
11.1 Allocating global buffers	274
11.1.1 Allocating index global buffers	274
11.1.2 Allocating data global buffers	275
11.1.3 Allocating LOB global buffers	278
11.1.4 Global buffer allocation procedures	278
11.2 Setting the number of global buffer sectors	281
11.3 Specifying the prefetch facility	283
11.4 Specifying the asynchronous READ facility	285
11.5 Specifying deferred write processing	286
11.6 Specifying the facility for parallel writes in deferred write processing	288
11.7 Setting the commit-time reflection processing	289
11.8 Global buffer LRU management	290
11.8.1 LRU management methods	290
11.8.2 LRU management suppression settings for a UAP	291
11.9 Page access using the snapshot method	293
11.10 Global buffer pre-writing	295
11.11 Local buffers	297
11.11.1 Allocating index local buffers	297
11.11.2 Allocating data local buffers	298
11.11.3 Allocating local buffers	298
11.11.4 Considerations about local buffers	299
12. Designing Tables	301
12.1 Items to be examined during table design	302
12.2 Normalizing a table	308
12.3 Table row partitioning	314
12.3.1 Table row partitioning	314
12.3.2 Types of table row partitioning	314
12.3.3 Forms of table row partitioning	325
12.3.4 Effects of table row partitioning	326
12.3.5 Design considerations	327
12.3.6 Notes on table row partitioning	334
12.4 Table matrix partitioning	336
12.5 Defining a trigger	343

12.5.1	Application standards.....	344
12.5.2	Defining a trigger.....	344
12.5.3	Trigger considerations.....	349
12.5.4	Trigger management.....	350
12.5.5	Error recovery.....	354
12.6	Creating a view table.....	356
12.7	Specifying the FIX attribute.....	359
12.8	Specifying a primary key.....	360
12.9	Specifying a cluster key.....	361
12.10	Specifying the suppress option.....	363
12.11	Specifying the no-split option.....	364
12.12	Specifying a binary data column.....	366
12.12.1	BLOB type.....	367
12.12.2	BINARY type.....	368
12.12.3	BLOB type and BINARY type usage.....	368
12.13	Specifying the WITHOUT ROLLBACK option.....	371
12.14	Specifying the falsification prevention facility.....	374
12.14.1	Specification.....	374
12.14.2	Restrictions.....	376
12.14.3	Changing a falsification-unprevented table to a falsification prevented table.....	379
12.14.4	Error operation.....	381
12.15	Table containing a repetition column.....	382
12.16	Table containing an abstract data type.....	385
12.17	Shared tables.....	391
12.17.1	Effects and criteria.....	393
12.17.2	Definition method.....	393
12.17.3	Manipulating shared tables.....	393
12.17.4	Limitations on shared tables.....	396
12.17.5	Notes about execution of definition SQL statements, utilities, and operation commands.....	396
12.17.6	Using shared tables with a HiRDB/Single Server.....	396
12.18	Referential constraints.....	399
12.18.1	About referential constraints.....	399
12.18.2	Defining referential constraints.....	400
12.18.3	Check pending status.....	412
12.18.4	Data manipulation and integrity.....	420
12.18.5	Procedure for checking table integrity.....	423
12.18.6	Referential constraints and triggers.....	430
12.18.7	Notes about linkage with related products.....	432
12.19	Check constraints.....	434
12.19.1	About check constraints.....	434
12.19.2	Defining check constraints.....	434
12.19.3	Check pending status.....	436

12.19.4	Data manipulation and integrity	438
12.19.5	Procedure for checking table integrity.....	438
12.19.6	Notes about linkage with related products	443
13.	Designing Indexes	445
13.1	Items to be examined during index design	446
13.2	Index	447
13.2.1	Creating an index.....	447
13.2.2	Index creation taking into account optimizing based on cost.....	449
13.2.3	Single-column index vs. multicolumn index.....	453
13.2.4	Using multiple indexes	455
13.2.5	Using an index with an exceptional key value set.....	455
13.2.6	Effects on performance of the number of indexes.....	456
13.3	Index row partitioning	457
13.4	Plug-in index.....	463
13.5	Plug-in index row partitioning.....	464
14.	Designing RDAREAs	471
14.1	Items to be examined during RDAREA design.....	472
14.2	Segments.....	476
14.2.1	Determining the segment size.....	476
14.2.2	Setting the percentage of free pages in a segment.....	478
14.2.3	Allocating and releasing segments	479
14.3	Pages	480
14.3.1	Determining the page length.....	480
14.3.2	Setting the percentage of unused space in a page.....	482
14.3.3	Allocating and releasing pages	483
14.4	Designing list RDAREAs	485
14.5	Free space reusage facility.....	488
14.5.1	Data storage search modes	488
14.5.2	Free space reusage facility.....	488
14.5.3	Effects and applicability	490
14.5.4	Considerations	492
14.5.5	Environment settings	493
14.5.6	Checking execution status	494
14.5.7	Notes.....	495
14.6	Shared RDAREAs (HiRDB/Parallel Server only).....	496
15.	Storage Requirements for HiRDB	501
15.1	Estimating the memory size required for a HiRDB/Single Server	502
15.1.1	Memory allocation.....	502
15.1.2	Calculation of required memory	505
15.1.3	Formulas for shared memory used by a unit controller.....	514
15.1.4	Formulas for shared memory used by a single server	524

15.1.5	Formula for size of shared memory used by global buffers.....	529
15.1.6	Formulas for size of memory required during SQL execution	531
15.1.7	Formula for size of memory required during SQL preprocessing	539
15.1.8	Formula for size of memory required during BLOB data retrieval or updating (HiRDB/Single Server).....	540
15.1.9	Formula for size of memory required during block transfer or array FETCH.....	541
15.2	Estimating the memory size required for a HiRDB/Parallel Server.....	543
15.2.1	Memory allocation	543
15.2.2	Calculation of required memory	547
15.2.3	Formulas for shared memory used by a unit controller	556
15.2.4	Formulas for shared memory used by each server.....	588
15.2.5	Formula for size of shared memory used by global buffers.....	599
15.2.6	Formulas for size of memory required during SQL execution	603
15.2.7	Formula for size of memory required during SQL preprocessing	612
15.2.8	Formula for size of memory required during BLOB data retrieval or updating (front-end server)	614
15.2.9	Formula for size of memory required during BLOB data retrieval or updating (back-end server or dictionary server)	614
15.2.10	Formula for size of memory required during block transfer or array FETCH (front-end server)	615
16.	Determining RDAREA Size	617
16.1	Determining the size of a user RDAREA.....	618
16.1.1	Calculating the size of a user RDAREA	618
16.1.2	Calculating the number of table storage pages.....	619
16.1.3	Calculating the number of index storage pages	633
16.2	Determining the size of a data dictionary RDAREA	646
16.2.1	Determining the size of a normal data dictionary RDAREA.....	646
16.2.2	Determining the size of a data dictionary RDAREA for storing database state analyzed tables and database management tables.....	679
16.3	Determining the size of the master directory RDAREA	682
16.4	Determining the size of the data directory RDAREA	683
16.5	Determining the size of a data dictionary LOB RDAREA	684
16.6	Determining the size of a user LOB RDAREA.....	693
16.7	Determining the size of the registry RDAREA.....	694
16.8	Determining the size of the registry LOB RDAREA.....	696
16.9	Determining the size of the list RDAREA	697
17.	Determining System File Size	699
17.1	Determining the size of system log files	700
17.1.1	Total size of system log files	700
17.1.2	Amount of system log information output during table definition	703
17.1.3	Amount of system log information output during index definition	704

17.1.4	Amount of system log information output during table data updating.....	705
17.1.5	Amount of system log information output during database creation by a utility	719
17.1.6	Amount of system log information that is output depending on the SQL manipulation.....	722
17.1.7	Amount of system log information that is output during the execution of the RDAREA automatic extension facility	722
17.1.8	Amount of system log information output when the PURGE TABLE statement is executed.....	723
17.1.9	Amount of system log information output when the free page release utility (pdreclaim) is executed	723
17.1.10	Amount of system log information that is output during execution of the facility for predicting reorganization time.....	724
17.2	Determining the size of synchronization point dump files	726
17.3	Determining the size of status files.....	727
18.	Determining Work Table File Size	735
18.1	Overview of determining the size of a work table file	736
18.2	Determining the size of a HiRDB file system area (pdfmkfs -n command).....	737
18.2.1	Size of a work table file used by an SQL statement.....	738
18.2.2	Size of a work table file used by a utility	744
18.3	Determining the maximum number of files (pdfmkfs -l command)	747
18.4	Determining the maximum number of extensions (pdfmkfs -e command).....	749
19.	Storage Requirements for Utility Execution	751
19.1	Determining the file sizes required for utility execution	752
19.1.1	File sizes required for the execution of the database load utility (pload).....	752
19.1.2	File sizes required for the execution of the database reorganization utility (pdrgorg).....	754
19.1.3	File sizes required for the execution of the statistics analysis utility (pdstedit).....	761
19.1.4	File sizes required for the execution of the database condition analysis utility (pddbst).....	763
19.1.5	File sizes required for the execution of the database copy utility (pdcopy)	764
19.1.6	File sizes required for the execution of the dictionary import/export utility (pdexp)	768
19.1.7	File sizes required for the execution of the optimizing information collection utility (pdgetcst)	769
19.1.8	File sizes required for the execution of the access path display utility (pdvwopt)	769
19.1.9	File sizes required for execution of the rebalancing utility (pdrbal)	770
19.1.10	File sizes required for execution of the integrity check utility (pdconstck)	772

19.2	Determining the memory size required for utility execution	773
19.2.1	Memory size required for the execution of the database initialization utility (pdinit).....	773
19.2.2	Memory size required for the execution of the database definition utility (pddef).....	774
19.2.3	Memory size required for the execution of the database load utility (pload).....	774
19.2.4	Memory size required for the execution of the database reorganization utility (pdrorg).....	779
19.2.5	Memory size required for the execution of the database structure modification utility (pdmod).....	781
19.2.6	Memory size required for the execution of the statistics analysis utility (pdstedit).....	783
19.2.7	Memory size required for the execution of the database condition analysis utility (pddbst).....	784
19.2.8	Memory size required for the execution of optimizing the information collection utility (pdgetcst).....	785
19.2.9	Memory size required for the execution of the database copy utility (pdcopy).....	786
19.2.10	Memory size required for the execution of the database recovery utility (pdrstr).....	788
19.2.11	Memory size required for the execution of the dictionary import/export utility (pdexp).....	790
19.2.12	Memory size required for the execution of the access path display utility (pdvwopt).....	792
19.2.13	Memory size required for the execution of the rebalancing utility (pdrbal).....	792
19.2.14	Memory size required for execution of the free page release utility (pdreclaim) and global buffer residence utility (pdpgbfon).....	797
19.2.15	Memory size required for execution of the integrity check utility (pdconstck).....	798
20.	Determining Environment Variables Related to the Number of Resources	801
20.1	HiRDB/Single Server	802
20.2	HiRDB/Parallel Server	804
21.	Windows Registry Settings	807
21.1	Estimating desktop heap settings.....	808
22.	Sample Files	811
22.1	Overview of sample files.....	812
22.1.1	Names of sample files	812
22.2	Table definition information.....	815
22.3	Use of the sample files	819

22.3.1	Creating the sample database	819
22.3.2	Customizing the sample database	820
22.3.3	HiRDB file system area names and user-created file names used with sample database	824
23.	Communication Between HiRDB Servers and HiRDB Clients	827
23.1	Connecting to a HiRDB server with an FQDN specified	828
23.1.1	Connection to a HiRDB server with an FQDN specified	828
23.1.2	Using the multi-connection address facility to connect to a HiRDB server	830
23.2	Settings for a DNS server to manage IP addresses	835
23.3	Settings when a firewall and NAT are installed	838
23.3.1	When a firewall is installed on the HiRDB/Single Server side	838
23.3.2	When a firewall and NAT are installed on the HiRDB/Single Server side	839
23.3.3	When a firewall is installed on the HiRDB/Parallel Server side	841
23.3.4	When a firewall and NAT are installed on the HiRDB/Parallel Server side	842
23.4	Port numbers used by HiRDB	845
23.4.1	Estimating the number of ports that a unit will use	845
23.4.2	Notes	846
23.4.3	Calculation examples	846
23.5	HiRDB reserved port facility	848
23.5.1	Estimation of the HiRDB reserved port facility	848
Appendixes		851
A.	HiRDB Maximum and Minimum Values	852
A.1	Maximum and minimum values for the system configuration	852
A.2	Maximum and minimum values for databases	853
A.3	Maximum and minimum values for HiRDB file names	855
B.	Questions and Answers	857
C.	Setting Up an Environment Using a Batch File	868
C.1	Overview of environment setup using a batch file	868
C.2	Modifying the HiRDB system definitions	874
Index		877

List of figures

Figure 1-1: System construction procedure for installing a new HiRDB.....	2
Figure 1-2: Procedure for replacing an existing version with the new version	17
Figure 1-3: Procedure to restore HiRDB to the previous version (when version upgrade fails).....	20
Figure 1-4: Procedure for deleting and re-creating the system files created by the new version	21
Figure 1-5: Handling errors during updating to the HiRDB update version	32
Figure 4-1: Procedure for using commands to set up the HiRDB environment.....	59
Figure 4-2: Configuration of HiRDB system definition files: HiRDB/Single Server	62
Figure 4-3: Configuration of HiRDB system definition files: HiRDB/Parallel Server	67
Figure 4-4: Configuration of HiRDB system definition files (when the HiRDB External Data Access facility is used).....	68
Figure 5-1: Plug-in registration procedure.....	120
Figure 5-2: Procedure for creating a registry RDAREA and registry LOB RDAREA	122
Figure 6-1: Database creation procedure	132
Figure 6-2: Differences in operating procedure based on the database update log acquisition mode (data storage)	134
Figure 6-3: Example of table containing abstract data type using substitutability	154
Figure 6-4: Procedure for inserting data in a table that contains an abstract data type column.....	158
Figure 6-5: Status of index in the event of an error during batch index creation by database load utility (pload).....	161
Figure 6-6: Error handling procedure in the event of abnormal termination of a utility during data loading with the synchronization point specification	167
Figure 7-1: Relationship between HiRDB and X/Open DTP model.....	175
Figure 7-2: Linking HiRDB/Single Server with an OLTP system (OpenTP1)	177
Figure 7-3: Linking HiRDB/Parallel Server with an OLTP system (OpenTP1)	178
Figure 7-4: Linking between multiple OLTPs (OpenTP1) and a single HiRDB.....	179
Figure 7-5: Linking between one OLTP (OpenTP1) and multiple HiRDBs	180
Figure 7-6: Overview of transaction transfer.....	181
Figure 7-7: Overview of event monitoring by JP1/IM	202
Figure 7-8: Automatic control achieved by JP1 linkage when system log files are unloaded.....	203
Figure 8-1: System configuration for a HiRDB/Single Server	206
Figure 8-2: Example of creating adjacent generations on separate hard disks (HiRDB/Single Server)	217
Figure 8-3: Number of guaranteed-valid generations for synchronization point dump files ..	218
Figure 8-4: Example of placing six sets of status files on four sets of disks	221
Figure 8-5: Example of system RDAREA placement (HiRDB/Single Server)	226
Figure 8-6: Example of data dictionary LOB RDAREA placement (HiRDB/Single Server).....	226
Figure 8-7: Example of user RDAREA placement (HiRDB/Single Server).....	227

Figure 8-8: Example of user LOB RDAREA placement (HiRDB/Single Server)	228
Figure 8-9: Example of list RDAREA placement (HiRDB/Single Server)	228
Figure 9-1: System configuration for a HiRDB/Parallel Server	232
Figure 9-2: System configuration for a HiRDB/Parallel Server with multiple front-end servers	233
Figure 9-3: Operation based on whether or not a recovery-unnecessary front-end server is used	237
Figure 9-4: Example of a system configuration using a recovery-unnecessary front-end server	238
Figure 9-5: Example of creating adjacent generations on separate hard disks (HiRDB/Parallel Server)	251
Figure 9-6: Number of guaranteed-valid generations for synchronization point dump file (HiRDB/Parallel Server)	252
Figure 9-7: Example of placing six sets of status files on four sets of disks	255
Figure 9-8: Example of system RDAREA placement (HiRDB/Parallel Server)	260
Figure 9-9: Example of data dictionary LOB RDAREA placement (HiRDB/Parallel Server)	261
Figure 9-10: Example of user RDAREA placement (HiRDB/Parallel Server)	262
Figure 9-11: Example of user LOB RDAREA and user RDAREA placement (HiRDB/Parallel Server)	263
Figure 9-12: Example of list RDAREA placement (HiRDB/Parallel Server)	263
Figure 9-13: Example of heterogeneous system configuration for HiRDB/Parallel Server ..	264
Figure 10-1: Example of service name and service port number for communication port....	269
Figure 11-1: Overview of global buffer for an index	275
Figure 11-2: Example of data global buffer allocation	276
Figure 11-3: Overview of the facility for parallel writes in deferred write processing.....	288
Figure 11-4: Overview of the snapshot method	293
Figure 11-5: Overview of global buffer pre-writing	295
Figure 11-6: Overview of index local buffers	298
Figure 12-1: Multiple columns in a table containing similar data	309
Figure 12-2: Same table used by multiple applications	310
Figure 12-3: Columns with different access frequencies	312
Figure 12-4: Example of key range partitioning with storage condition specified.....	322
Figure 12-5: Example of key range partitioning with boundary values specified	323
Figure 12-6: Example of flexible hash partitioning and FIX hash partitioning	324
Figure 12-7: Table row partitioning form for a HiRDB/Single Server	325
Figure 12-8: Table row partitioning form for a HiRDB/Parallel Server	326
Figure 12-9: Overview of row partitioning taking into account contention among disk accesses.....	328
Figure 12-10: Overview of row partitioning taking into account operability	329
Figure 12-11: Overview of input/output processing performance based on number of back-end servers used for table row partitioning	331
Figure 12-12: Performance of parallel processing depending on table access frequency	332

Figure 12-13: System configuration involving table row partitioning taking into account complicated retrieval processing.....	334
Figure 12-14: Matrix partitioning example (combination of key range partitioning with boundary values specified).....	339
Figure 12-15: Example of matrix partitioning (combination of key range partitioning with boundary values specified and hash partitioning).....	341
Figure 12-16: Overview of triggers	343
Figure 12-17: Relationship between base table and view table.....	356
Figure 12-18: Overview of overhead when data is added to a table with a cluster key specified	362
Figure 12-19: Data storage method when actual variable-length character string data is 256 bytes or greater	364
Figure 12-20: Data storage method when the no-split option is specified	365
Figure 12-21: Example of a numbering application	372
Figure 12-22: Reloading to another table	378
Figure 12-23: Example of tables defined without repetition columns.....	382
Figure 12-24: Example of table containing repetition columns	383
Figure 12-25: Data structure of a table containing an abstract data type.....	385
Figure 12-26: Hierarchical structure based on the relationship between supertype and subtype abstract data types	387
Figure 12-27: Data structure of a table containing an abstract data type (using substitutability).....	388
Figure 12-28: Join processing without using a shared table.....	391
Figure 12-29: Join processing using a shared table	392
Figure 12-30: Example of referenced and referencing tables.....	399
Figure 12-31: Example of the actions that occur when update SQL code is executed on a referenced table (with CASCADE specified)	401
Figure 12-32: Example of the actions that occur when update SQL code is executed on a referencing table (with CASCADE specified).....	401
Figure 12-33: Example of the actions that occur when update SQL code is executed on a referenced table (with RESTRICT specified).....	404
Figure 12-34: Data manipulation availability when managing check pending status for each RDAREA in a partitioned table	419
Figure 12-35: Overview of procedure for checking data integrity (referential constraint)	423
Figure 12-36: Procedure for checking data integrity when check pending status is used (referential constraint).....	426
Figure 12-37: Procedure for checking data integrity when check pending status is not used	429
Figure 12-38: Overview of the procedure for checking data integrity (check constraint).....	439
Figure 12-39: Procedure for checking data integrity when check pending status is used (check constraint).....	441
Figure 13-1: Partitioning key index (single-column partitioning).....	457
Figure 13-2: Partitioning key index (multicolumn partitioning)	458
Figure 13-3: Example of index row partitioning (HiRDB/Single Server).....	460
Figure 13-4: Example of index row partitioning (within one server).....	461

Figure 13-5: Example of index row partitioning (among multiple servers)	462
Figure 13-6: Form of plug-in index row partitioning (HiRDB/Single Server)	464
Figure 13-7: Example of plug-in index row partitioning (key range partitioning) (HiRDB/Single Server)	465
Figure 13-8: Form of plug-in index row partitioning (HiRDB/Parallel Server)	467
Figure 13-9: Example of plug-in index row partitioning (key range partitioning) (HiRDB/Parallel Server)	468
Figure 14-1: Overview of RDAREAs depending on segment size	477
Figure 14-2: Overview of percentage of free pages in a segment	478
Figure 14-3: Overview of unused space in pages	482
Figure 14-4: Overview of the free space reusage facility	489
Figure 14-5: Overview of a shared RDAREA	496
Figure 15-1: Memory allocation for a HiRDB/Single Server	502
Figure 15-2: Memory allocation for each unit of a HiRDB/Parallel Server	544
Figure 17-1: Concept of index page splitting	715
Figure 22-1: Directory structure of sample files	812
Figure 22-2: Sample batch file	823
Figure 23-1: Example of network configuration and definition for connecting to HiRDB system with an FQDN specified	829
Figure 23-2: Example of network configuration and definition: HiRDB/Single Server	831
Figure 23-3: Example of network configuration and definition: HiRDB/Parallel Server	832
Figure 23-4: Example of network configuration and definition: With inheritance of IP addresses during system switchover	833
Figure 23-5: Example of network configuration and definition: Without inheritance of IP addresses during system switchover	834
Figure 23-6: Example system configuration using a single domain	835
Figure 23-7: Example system configuration using multiple domains	836
Figure 23-8: Network configuration example with a firewall installed on the HiRDB/Single Server side	838
Figure 23-9: Network configuration example with a firewall and NAT installed on the HiRDB/Single Server side	840
Figure 23-10: Network configuration example with a firewall installed on the HiRDB/Parallel Server side	841
Figure 23-11: Network configuration example with a firewall and NAT installed on the HiRDB/Parallel Server side	843
Figure C-1: Environment setup procedure using a batch file	868
Figure C-2: Sample system configuration created by SPsetup.bat	870
Figure C-3: Organization of sample HiRDB file system areas created by SPsetup.bat	871

List of tables

Table 1-1: Advantages and disadvantages of each environment setup method.....	3
Table 1-2: Items supported in the Windows Server 2003 (IPF) system switchover facility when Windows Terminal Service is used	6
Table 1-3: Organization of directories and files that the HiRDB administrator creates	7
Table 1-4: Directories and files that HiRDB creates	7
Table 1-5: Free space required for upgrading	13
Table 2-1: Required virtual memory size (HiRDB/Single Server).....	38
Table 2-2: Required virtual memory size (HiRDB/Parallel Server).....	39
Table 2-3: System environment variables to be set	44
Table 2-4: Option program products facility and target installation servers.....	49
Table 4-1: Types of HiRDB file system areas	72
Table 5-1: HiRDB (unit) startup conditions and availability of plug-ins	123
Table 5-2: SQL statements for deleting database resources	127
Table 6-1: Database update log acquisition mode for user LOB RDAREAs depending on the -l option value in pload	133
Table 6-2: Database update log acquisition method for user LOB RDAREAs depending on a specification in the client environment definition.....	159
Table 7-1: Functions supported by HiRDB XA library	175
Table 7-2: Relationship between OpenTP1's trnstring operand and HiRDB's PDXAMODE operand	182
Table 7-3: Differences between dynamic and static registration	184
Table 7-4: Library names for RM-related object names.....	189
Table 7-5: HiRDB events that can be sent to JP1/Base	200
Table 8-1: Maximum sizes of HiRDB file system areas.....	210
Table 8-2: Effects on operations of the number of generations of system log files.....	214
Table 8-3: HiRDB operation and HiRDB administrator's action that depend on whether or not single operation is used	223
Table 9-1: Number of permitted servers	230
Table 9-2: Server that acquires the current timestamp depending on whether or not USING BES is specified.....	235
Table 9-3: Client environment definitions required for multiple front-end servers.....	235
Table 9-4: Maximum sizes of HiRDB file system areas.....	245
Table 9-5: Effects on operations of the number of generations of system log files.....	248
Table 9-6: HiRDB operation and HiRDB administrator's action that depend on whether or not single operation is used	257
Table 11-1: Applicability of the snapshot method for retrievals	294
Table 12-1: Items to be examined for database table design	302
Table 12-2: Hash functions	316
Table 12-3: Differences among key range partitioning, flexible hash partitioning, and FIX hash partitioning	319

Table 12-4: Recommended data type for each binary data usage	369
Table 12-5: Operations permitted on falsification prevented tables	374
Table 12-6: Restricted definition SQL statements and the restrictions	376
Table 12-7: Restricted utilities and the restrictions	377
Table 12-8: Restricted commands and their restrictions	379
Table 12-9: Meanings of values in the SQL_TABLES table	381
Table 12-10: Encapsulation levels and privileges	390
Table 12-11: Primary key operations and the resulting actions that occur in the referencing table (with CASCADE specified)	402
Table 12-12: Foreign key operations and the resulting actions that occur in the referenced table (with CASCADE specified)	402
Table 12-13: Primary key operations and the resulting actions that occur in referenced and referencing tables.....	404
Table 12-14: Storage locations of check pending status information and contents (referential constraint)	413
Table 12-15: Availability of operations on tables in check pending status	414
Table 12-16: Restricted operations when table T2 is in check pending status.....	415
Table 12-17: Restricted operations when table T3 is in check pending status.....	416
Table 12-18: Restricted operations when tables T2 and T3 are in check pending status.....	417
Table 12-19: Operations on referenced tables that nullify the guarantee of data integrity and the conditions under which loss of data integrity occurs	420
Table 12-20: Operations on referencing tables that nullify the guarantee of data integrity and the conditions under which loss of data integrity occurs	421
Table 12-21: Conditions that cause moving of mismatched data in a table between RDAREAs when the target table is a partitioned table	422
Table 12-22: Storage locations of check pending status information and their contents (check constraint)	437
Table 12-23: Operations on check constraint tables that nullify the guarantee of data integrity and the conditions under which loss of data integrity occurs.....	438
Table 13-1: Items to be examined during index design	446
Table 13-2: Priority order of index usage by HiRDB	449
Table 13-3: Whether or not HiRDB uses an index whose exception key value is the null value	453
Table 13-4: Priority among single-column and multicolumn indexes	455
Table 13-5: Index partitioning guidelines	459
Table 14-1: Items to be examined during RDAREA design	472
Table 14-2: Maximum and minimum values for RDAREAs.....	474
Table 14-3: Segment statuses	476
Table 14-4: Page statuses	480
Table 15-1: HiRDB/Single Server shared memory details	503
Table 15-2: Size of memory required for a HiRDB/Single Server	506
Table 15-3: Procedure for obtaining the number of work tables for each SQL statement.....	513
Table 15-4: Length of column subject to grouping and length of operation area for set functions	531

Table 15-5: Data length of embedded variables and ? parameters	538
Table 15-6: HiRDB/Parallel Server shared memory details per unit.....	544
Table 15-7: Size of memory required for each unit of a HiRDB/Parallel Server	548
Table 15-8: Procedure for obtaining the number of work tables for each SQL statement	556
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) 600	600
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)	601
Table 15-11: Length of column subject to grouping and length of operation area for set functions	604
Table 16-1: List of data lengths	622
Table 16-2: Data lengths for the variable-length character string type (except abstract data type and repetition columns).....	627
Table 16-3: Data lengths for the variable-length character string type (abstract data type) ...	628
Table 16-4: Data lengths for the variable-length character string type (repetition columns) .	629
Table 16-5: List of index key lengths	637
Table 16-6: Variables used in the formulas for obtaining the number of index storage pages.....	670
Table 17-1: Types of system log information that are output when table rows are manipulated	706
Table 17-2: Amount of log information depending on type of database manipulation	707
Table 17-3: Formulas for determining the amount of base row log information per data item.....	708
Table 17-4: Formulas for determining the amount of log information per BLOB column data item.....	709
Table 17-5: Formulas for determining the amount of log information per branch row (1)....	711
Table 17-6: Formulas for determining the amount of log information per branch row (2)....	712
Table 17-7: Amount of index log information per index	714
Table 17-8: Amount of index log information by split type	716
Table 17-9: Amount of log information for page allocation and segment allocation	718
Table 17-10: Amount of event log information that is output when a single row is manipulated	718
Table 17-11: Formulas for determining the amount of system log information output during database creation by a utility.....	719
Table 17-12: Values added when calculating the amount of system log information, and conditions required to perform this addition.....	721
Table 17-13: Amount of system log information that is output during the execution of the RDAREA automatic extension facility	722
Table 17-14: Amount of system log information output during execution of the PURGE TABLE statement.....	723
Table 17-15: Amount of system log information output when the free page release utility is executed.....	724

Table 18-1: Options for which values need to be specified	736
Table 18-2: Obtaining the number of rows in a column information work table.....	739
Table 18-3: Obtaining the data length for each column and the number of columns in a work table	740
Table 18-4: Obtaining the maximum number of column information work tables	742
Table 18-5: Obtaining the maximum number of location information work tables.....	743
Table 19-1: Maximum lengths of converted character strings for columns (in DAT or extended DAT format)	758
Table 19-2: Maximum lengths of converted character strings for columns (fixed-length character format).....	760
Table 20-1: Values of environment variables related to the number of resources (HiRDB/Single Server)	802
Table 20-2: Values of environment variables related to the number of resources (HiRDB/Parallel Server)	804
Table 22-1: Directories and files used with sample database.....	813
Table 22-2: Contents of sample configuration	813
Table 22-3: Contents of sample UOC	814
Table 22-4: Tables provided as sample database	815
Table 22-5: Column attributes and indexes of the tables	815
Table 22-6: Batch files for creating the sample database.....	819
Table 22-7: Sample batch file for executing commands under aliases	822
Table 22-8: Names and sizes of HiRDB file system areas created by SPsetup.bat	824
Table 22-9: Names of the user-created files created by SPsetup.bat.....	825
Table 23-1: Names allowed in client environment definition	828
Table A-1: Maximum and minimum values for the HiRDB system configuration	852
Table A-2: Maximum and minimum values for databases	854
Table A-3: Maximum and minimum values for HiRDB file names (in characters).....	856
Table C-1: Contents of sample configuration	869
Table C-2: Names and sizes of HiRDB file system areas created by SPsetup.bat.....	872
Table C-3: Names of files created by SPsetup.bat	873

Chapter

1. Overview of HiRDB System Construction

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

This chapter contains the following sections:

- 1.1 System construction procedures
- 1.2 Organization of HiRDB directories and files
- 1.3 Upgrading HiRDB
- 1.4 Updating to HiRDB update version

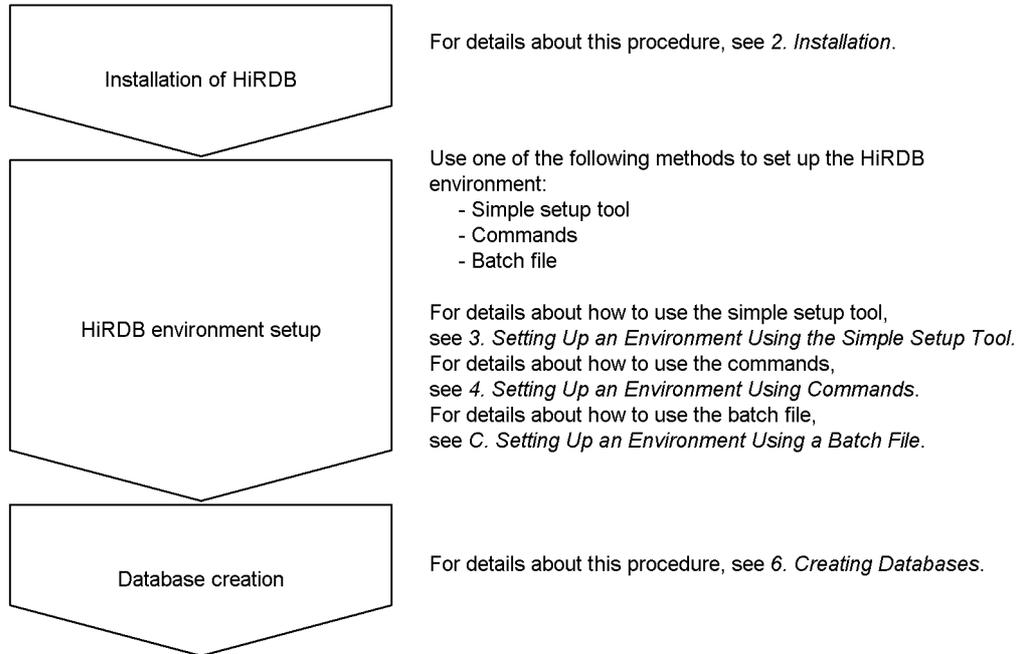
1.1 System construction procedures

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

1.1.1 System construction procedure for installing a new HiRDB

Figure 1-1 shows the system construction procedure for installing a new 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 manual *HiRDB Version 8 System Operation Guide*.

1.1.2 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
- A batch file (`SPsetup.bat`)

Hint:

Normally, Hitachi recommends that you use the simple setup tool to set up the HiRDB environment.

Table 1-1 shows 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. For details about how to use the simple setup tool, see 3. <i>Setting Up an Environment Using the Simple Setup Tool.</i>	Easier than using other methods to set up the HiRDB environment. The simple setup tool allows you to start using HiRDB immediately. You can also use it to change existing system definition settings.	The HiRDB system construction is limited to the range of configurations that can be set up using the simple setup tool.
Commands ^{#1}	Use HiRDB commands to set up the HiRDB environment. For details about how to use commands, see 4. <i>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.
Batch files (SPsetup.bat) ^{#2}	Execute a batch file to set up the HiRDB environment. For details about how to use the batch file, see C. <i>Setting Up an Environment Using a Batch File.</i>	Easier than using commands to set up the HiRDB environment. Use of the batch file allows you to start using HiRDB immediately.	The HiRDB system construction is limited to the range of configurations that can be set up using the settings in the batch file.

#1

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.

For details about how to perform a simple installation, see 22. *Sample Files*. For details about how to install a HiRDB/Single Server, see the manual *HiRDB First Step Guide*.

#2

The batch file method can be used only for installing a HiRDB/Single Server. For details about environment setup for the HiRDB/Parallel Server, see %PDDIR%\HiRDEF\readme.txt.

Note:

- With the simple setup tool method, a plug-in environment cannot be set up.
- With the batch file method, the HiRDB environment settings are configured automatically. Once setup is complete, the HiRDB administrator can modify these settings to achieve an environment better suited to his or her needs.

1.1.3 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 manual *HiRDB Version 8 System Operation Guide*.

(4) Using the Directory Server linkage facility

HiRDB can be linked to the following directory servers:

- Sun Java System Directory Server

For details about the environment setup procedure for the Hitachi Directory Server linkage facility, see the manual *HiRDB Version 8 System Operation Guide*.

(5) Using the HiRDB External Data Access facility

HiRDB External Data Access is required in order to use the HiRDB External Data Access facility. For details about the environment setup procedure for the HiRDB External Data Access facility, see *7.3 Linking to the HiRDB External Data Access facility*.

(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*.

1.1.4 Using Windows Terminal Service

You can use Windows Terminal Service to operate HiRDB. This capability enables you to operate HiRDB on machines in remote areas and on machines with no console. For details about Windows Terminal Service, see the operating system documentation.

Windows Terminal Service includes console sessions and virtual sessions.

- Console sessions

A console session switches input and output devices (such as the display, keyboard and mouse) from the server that you are using to the client device of a terminal server. Note the following:

- Specify the `/console` option when you execute the command for connecting a remote desktop to the server.
- Only one session can be created on the client and server (the original window becomes disabled (logged off)).
- Windows 2000 does not support this session.

- Virtual sessions

A virtual session controls the server you are operating in the background (a client operation window is created in the background). Note the following:

- Execute the command for connecting a remote desktop to the server without any options specified.
- Multiple sessions can be created.

(1) Notes**(a) Accessing files under the operating system installation directory**

When you operate an application using Windows Terminal Service, the initialization files (file extension: `.INI`) created under the directory for each user (for example, `c:\windows\Documents and Settings\log-n user-name\Windows`) may be used rather than the initialization files under the directory where the operating system is installed (for example, `c:/windows`).

In such a case, you need to copy the contents of the `HiRDB.INI` file located under the operating system installation directory into the `HiRDB.INI` file under the directory created for each user. However, to prevent `HiRDB.INI` files in different locations from

affecting the execution environment of command prompts to execute operating commands, utilities, and client programs, you can set client environment definitions for the execution environment. For details about how to set client environment definitions, see the manual *HiRDB Version 8 UAP Development Guide*.

(b) Notes on security

When Windows Terminal Server is used, even though the server console is not in login status, the client side window may still run in login status. To avoid this, you must do the following in the server or client:

- Use a screensaver password for security.
- Log off when operation from the client is completed.

(c) Temporary limitations

Support for the Windows Server 2003 (IPF) system switchover facility is limited.

Table 1-2 lists the items supported in the Windows Server 2003 (IPF) system switchover facility when Windows Terminal Service is used.

Table 1-2: Items supported in the Windows Server 2003 (IPF) system switchover facility when Windows Terminal Service is used

Type		Support	
System switchover facility	Monitor mode	Y	
	Server mode ^{#1}	Console session	Y
		Virtual session	-- ^{#2}

Legend:

Y: Supported.

--: Not supported.

#1

When `pd_ha_acttype = server` is specified in the unit common definition.

#2

Operation of a server PC or console session is required. If you operate such commands as `pdstart`, `pdstop`, `pdprgnew`, or `pdchgconf`, system switchover occurs because the HiRDB status cannot be set in Hitachi HA Toolkit Extension.

1.2 Organization of HiRDB directories and files

(1) Directories and files that the HiRDB administrator creates

Table 1-3 lists the directories and files that the HiRDB administrator creates.

Table 1-3: 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\server-name	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

Table 1-4 lists the directories and files that HiRDB creates.

Table 1-4: 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\servers	Directory for storing HiRDB server's executable files and libraries
%PDDIR%\client\lib	Directory for storing HiRDB client's libraries
%PDDIR%\client\utl	Directory for storing HiRDB client's commands and utilities
%PDDIR%\client\jba	Directory for storing HiRDB Java stored procedure/function distribution wizard (created for use with the Windows or Windows (x64) version of HiRDB)
%PDDIR%\client\include	Directory for storing the header information that is used during UAP creation

1. Overview of HiRDB System Construction

File or directory name	Description
%PDDIR%\include	Directory for storing header information that is used during UAP creation
%PDDIR%\spool\save ¹	Directory for storing saved core files
%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\scdqid1, scdqid2	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	HiRDB's internal work directory
%PDDIR%\spool\olkrsfs	HiRDB's internal work directory

File or directory name	Description
%PDDIR%\spool\cnetusrinf	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\cnetusrdtl	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
%PDDIR%\spool\pdsqldump ¹	Directory for storing access path information files
%PDDIR%\spool\pdrshs1.log, pdrshs2.log	File for storing information about remote commands
%PDDIR%\spool\pdsha1.log, pdsha2.log	File for storing information about the system switchover facility
<i>any name</i> ²	RPC trace file
%PDDIR%\tmp ³	HiRDB's internal work directory
%PDDIR%\tmp\pdommenv	File for storing shared memory information
%PDDIR%\tmp\home\ <i>processor-name</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%\.dbenv	Directory for storing HiRDB database environment information files
%PDCLTPATH\pdsq11.trc, pdsq12.trc ⁴	Files for storing trace information for SQL that is executed by a UAP
%PDCLTPATH\pderr1.trc, pderr2.trc ⁴	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\lib	Directory for storing plug-in libraries
%PDDIR%\plugin\ <i>plug-in-name</i>	Plug-in 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

1. Overview of HiRDB System Construction

File or directory name	Description
%PDDIR%\plugin\plug-in-name\conf	Directory for storing plug-in configuration files
%PDDIR%\jre ⁵	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%\pdistup	Directory containing the simple setup tool
%PDDIR%\sample	Directory containing an execution environment for the sample database
%PDDIR%\HiRDEF	Directory containing an execution environment for the HiRDB definition support
%PDDIR%\UXPLDIR	Directory used by the differing system call specifications absorption library
%PDDIR%\UXPLDIR\spool\uxpllog1.txt, uxpllog2.txt	File for storing the information that is output by the differing system call specifications absorption library
%PDDIR%\Setup.ini	File for managing installation information

¹ Because HiRDB uses this directory to output troubleshooting information, it may keep increasing in size. You should use the `pd_cspool` 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 8 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 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.

⁴ 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 or not X/Open-compliance API (TX_ function) was used. When the TX_ function is used, the files are created with the following names:

- pdsqxxxx-1.trc, pdsqxxxx-2.trc
- pderrxxxx-1.trc, pderrxxxx-2.trc

Legend:

xxxx: Process ID during UAP execution

Because the process ID is used in the file name, note that as many files as there are processes may be output during UAP execution.

⁵ 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.

1.3 Upgrading HiRDB

This section describes the procedure for upgrading HiRDB.

When upgrading a HiRDB/Parallel Server, upgrade all units constituting the HiRDB/Parallel Server so that they have the same version of HiRDB.

Some default values for HiRDB system definition operands have changed from version 07-00 (default values have been changed to recommended values). Also, some operands no longer need to be specified. If you are upgrading from HiRDB Version 6 but do not want to change the default values, set `v6compatible` in the `pd_sysdef_default_option` operand. For details about the default values that have changed, the operands that no longer need to be specified, and the `pd_sysdef_default_option` operand, see the manual *HiRDB Version 8 System Definition*.

Notes

- 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 manual *HiRDB Version 8 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 *1.3.4 Using Java stored procedures and functions*.

1.3.1 Before upgrading

Before upgrading, make sure that the steps described below are taken. For a multi-HiRDB, you must apply the following actions to all directories:

- Check to see whether HiRDB is online.
- Check the HiRDB status.

Before upgrading, use the Windows **Command Prompt** window to execute commands and utilities.

(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 (`pdroorg`).

- 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 8 Command Reference*.

Free space required for upgrading

Check to see if you have enough free space, as shown in Table 1-5, according to the HiRDB version you are upgrading. If there is not enough space, an insufficient space error may occur when you start HiRDB or execute the `pdvrrup` command after upgrading.

Table 1-5: 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 + ↑ 5 ÷ S ↑	5 + ↑ 5 ÷ S ↑	5 + ↑ 15 ÷ S ↑	5 + ↑ 20 ÷ S ↑	5 + 1 ↑ 25 ÷ S ↑
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 + ↑ 5 ÷ S ↑	3 + ↑ 5 ÷ S ↑	3 + ↑ 15 ÷ S ↑	3 + ↑ 20 ÷ S ↑	3 + ↑ 25 ÷ S ↑
SQL_VIEW_DEF table	2 + ↑ 10 ÷ S ↑	3 + ↑ 10 ÷ S ↑	2 + ↑ 40 ÷ S ↑	2 + ↑ 65 ÷ S ↑	2 + ↑ 70 ÷ S ↑
SQL_DIV_COLUMN table ¹	—	—	—	1	—
SQL_ROUTINES table ²	—	—	4 + ↑ 30 ÷ S ↑	4 + ↑ 30 ÷ S ↑	—

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_ROUTINE_PARAM S 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 RDAREA
- Audit trail table RDAREA (applicable when the security audit facility is used)

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 8 System Operation Guide* or *HiRDB Version 8 Command Reference*.

(3) Checking the memory size

You need 135 KB of memory to execute the upgrading command (`pdvrup`). Check your available memory space. Note that this action is not necessary when you are updating to the HiRDB update version.

(4) 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.

(5) Terminating HiRDB normally

Before upgrading, terminate HiRDB normally. In the case of a HiRDB/Parallel Server, 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 event log

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

(6) 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 event log before this message, eliminate the cause of the error, and then start the service. After that, re-start the unit, and then terminate it normally with the `pdstop` command.

(7) Stopping the HiRDB service

Before performing a version upgrade, you must stop the HiRDB service. For details about stopping the service and checking that it has stopped, see the documentation provided with the operating system.

(8) Terminating commands, utilities, applications, HiRDB Datareplicator, and HiRDB Dataextractor

You must first terminate all commands, utilities, applications, HiRDB Datareplicator, and HiRDB Dataextractor. If any such item is running, deletion of executable files and shared libraries fails, resulting in an upgrading error.

(9) Checking the memory requirements

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

(10) 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.

(11) 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.*

(12) 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

$$\text{Total-number-of-records}^* > 5754880/\text{system-log-file-record-length}$$

When upgrading from HiRDB Version 5.0 or earlier

$$\text{Total-number-of-records}^* > 4239360/\text{system-log-file-record-length}$$

When upgrading from HiRDB Version 6 or earlier

$$\text{Total-number-of-records}^* > 3215360/\text{system-log-file-record-length}$$

When upgrading from HiRDB Version 7 or earlier

$$\text{Total-number-of-records}^* > 1413120/\text{system-log-file-record-length}$$

For a HiRDB/Parallel Server, 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.

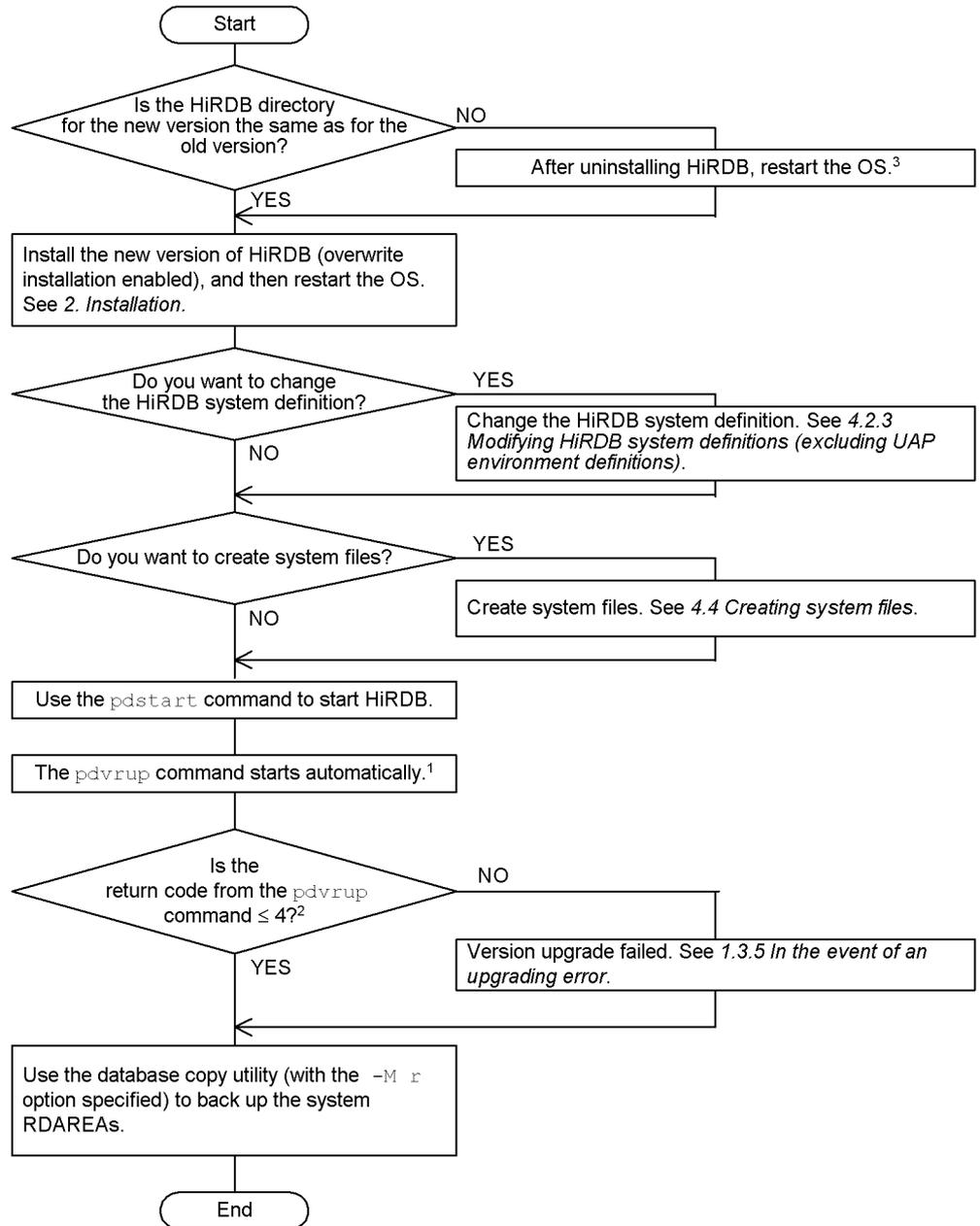
(13) Upgrading option program products

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 *2.4 Notes about option program products installation.*

1.3.2 Replacing an existing version with the new version

Figure 1-2 shows the procedure for replacing an existing version with the new 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 event log.

3

To restore the old version of HiRDB, you need to use the old version's installation directory. Therefore, write down the name of the old version's installation directory.

1.3.3 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*.

1.3.4 Using Java stored procedures and functions

With 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 situation:

- When uninstalling HiRDB

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 manual *HiRDB Version 8 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 8 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.

1.3.5 In the event of an upgrading error

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

- The `pdvrrup` 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 `pdvrrup` 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 `pdvrrup` command entry request message (`KFPS05203-Q`) is issued, so that you can reenter the `pdvrrup` command.

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

To correct the error, you may need to restore the old 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 old version of HiRDB and use the database structure modification utility (`pdmod`). In this case, restore the old version of HiRDB, eliminate the cause of the error, and then upgrade HiRDB again.

For details about how to restore the old version of HiRDB, see *1.3.6 Restoring HiRDB to the previous version when version upgrade fails*.

1.3.6 Restoring HiRDB to the previous version when version upgrade fails

Figure 1-3 shows how to restore HiRDB to the previous version when version upgrade fails.

Figure 1-3: Procedure to restore HiRDB to the previous version (when version upgrade fails)

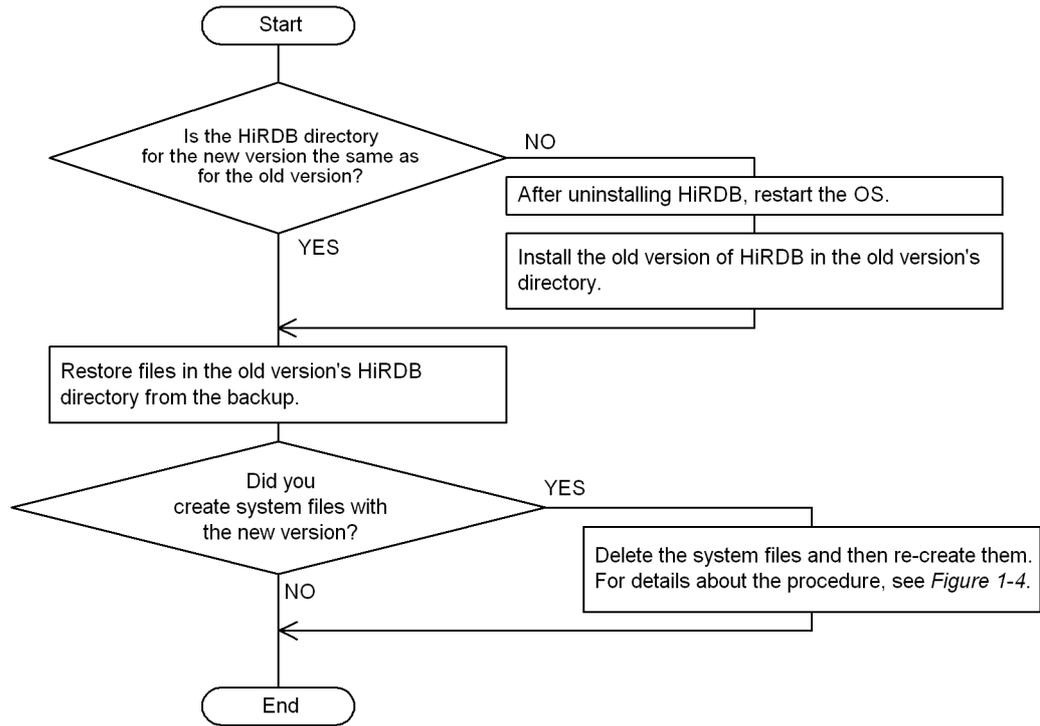
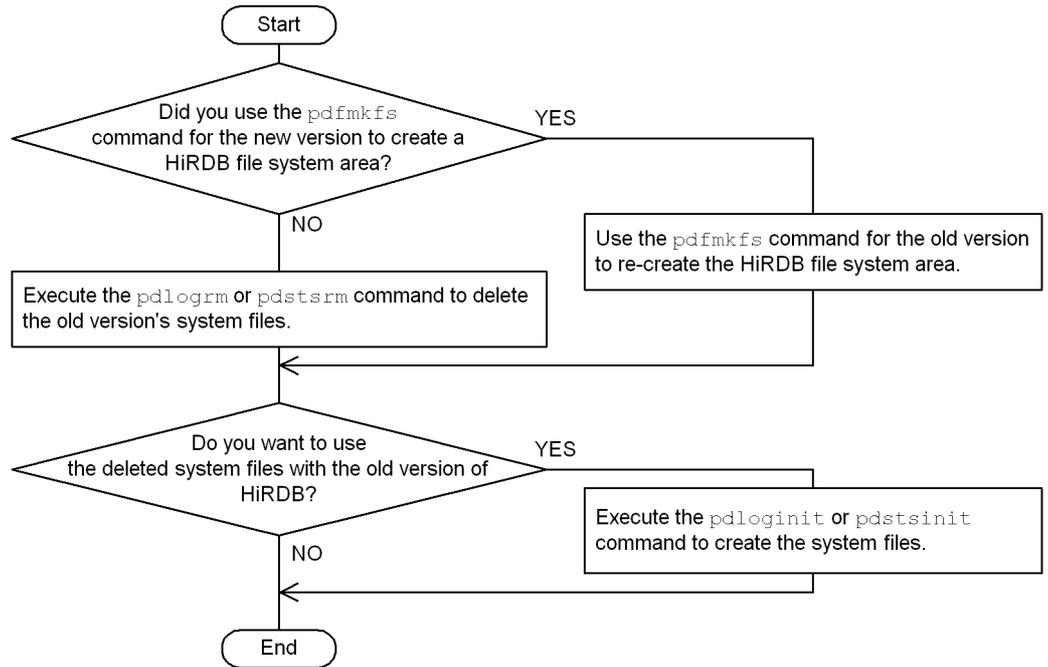


Figure 1-4 shows the procedure for deleting and re-creating the system files that have been created by the new version.

Figure 1-4: Procedure for deleting and re-creating the system files created by the new version



1.4 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-A, where the underlined part is the *-mn* code. For versions earlier than 07-03, *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 07-03 and later, *m* and *n* are both numbers.

You can switch to the HiRDB update version while the existing version of HiRDB is running. However, if you install MSCS and use the system switchover facility, and you have registered the HiRDB service in the MSCS general service, you cannot update to the HiRDB update version while running the existing HiRDB.

1.4.1 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

The procedure for updating after terminating HiRDB is the same as for performing a version upgrade; for details, see *1.3 Upgrading HiRDB*. In this case, the differences in updating to the HiRDB update version are as follows:

- The following steps in *1.3.1 Before upgrading* are not needed:
 - (1) *Checking for available space*
 - (3) *Checking the memory size*
- Execution of the `pdvrrup` command is not required.

(b) Updating with HiRDB operating

See *1.4.2* 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`, which is included with the update patch.

Since there are other HiRDB products in the package, a window appears for you to select the product with which you want to use the patch. Use the update patch of the product you select in the following window.

**(b) Update while running HiRDB**

See 1.4.2 and following sections.

1.4.2 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 or HiRDB/Parallel Server)
- Addressing mode (32-bit mode or 64-bit mode)

- 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. For the space requirements of the HiRDB update version, see the Readme file. In addition,

because the HiRDB update version must be newly installed, sufficient free disk space for new installation is required.

■ HiRDB client

A HiRDB client that is working online must be running on an 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 manual *HiRDB Version 8 UAP Development Guide*.

1.4.3 Actions performed prior to updating

For a HiRDB/Parallel Server, set the OS environment file before updating to the HiRDB update version.

The service name (pdrshsrvRENEW) of the port for connecting to the SERVICES file (%windir%\system32\drivers\etc\SERVICES) and the number of the port for connecting to the remote command shell are set for the server machines on all units. The port numbers must be the same for the server machines on all units. The same number cannot be reused within a machine.

1.4.4 Update procedure

To update to the HiRDB update version:

1. Install the HiRDB update version.

Note the following when installing the HiRDB update version:

- Do not overwrite install the HiRDB update version; instead, install as a new application.
- Versions of HiRDB installed using **Setup for Program Maintenance** can be used only for updating.
- For HiRDB/Parallel Server, install the HiRDB update version in a directory with the same path name throughout all units.

The installation procedure depends on the medium of the HiRDB update version. The procedure for each medium is as follows:

For the installer on the HiRDB installation CD-ROM (for HiRDB 07-01 or earlier)

To install the HiRDB update version:

1. To start the installer, double-click `01_srv\Setup.exe` on the HiRDB update version installation CD-ROM.
2. In the Setup Type window displayed while executing the installer, choose **Setup for Program Maintenance** to install the HiRDB update version in a different directory from that used for the running HiRDB.

For the installer on the integrated CD-ROM (for HiRDB 07-02 and later)

1. To start the Hitachi Integrated Installer, execute `hcd_inst.exe` in the HiRDB update version integrated CD-ROM.
2. To start the HiRDB installer, in the Hitachi Integrated Installer window, choose **HiRDB/Single Server** for the HiRDB/Single Server or choose **HiRDB/Parallel Server** for the HiRDB/Parallel Server.
3. In the HiRDB installer's Choose Program Product window, to start the installer choose **HiRDB/Single Server** for a HiRDB/Single Server or **HiRDB/Parallel Server** for a HiRDB/Parallel Server.
4. In the **Setup Type** window that displays while executing the HiRDB/Single Server or HiRDB/Parallel Server installer, choose **Setup for Program Maintenance** to install the HiRDB update version in a different directory from that used for the running HiRDB.

For an update patch (HiRDB 07-03 and later)

1. To start the Hitachi Integrated Installer, execute `hcd_inst.exe` on the integrated CD-ROM used when installing the HiRDB that is currently running.
 2. For the Installer on the integrated CD-ROM for HiRDB 07-02 and later, perform the same procedure as described in steps 2 to 4 above to install HiRDB.
 3. To use the patch on HiRDB installed in step 2, execute the HiRDB update version update patch. For details about how to use the patch, see `RELEASE.TXT`. The latest update patch includes previous updates, thus you do not need to use patches issued in the past.
2. Start the HiRDB update version service (HiRDB/Parallel Server only).
- To start the HiRDB update version service:

1. On the **Control Panel**, double-click **Services**.
 2. From the **Services** list box, select the HiRDB update version service **HiRDB/ParallelServer-RNW**, and click the **Startup** button.
 3. As **logon** is set to the system account, select that account. Click the button to the right of the account to display the **Add a User** screen.
 4. From **Name List**, select the HiRDB administrator. Click **Add**, and then click **OK**.
 5. On the **Password** screen, enter the HiRDB administrator's password. Re-enter the password on the **Reconfirm Password** screen, and click **OK**.
 6. From the **Service** list box, select the HiRDB update version service, and then click **Startup**.
3. Copy the HiRDB update version to the update directory.
- Choose **Start, Programs, HiRDBSingleServer-RNW** or **HiRDBParallelServer-RNW**, and then **HiRDB Command Prompt** to activate the work console for the HiRDB update version. On the work console, execute the `pdprgcopy` directory command of the operating HiRDB to copy the HiRDB update version to the update directory in the HiRDB directory of the operating HiRDB (%PDDIR%\renew). With a HiRDB/Parallel Server, activate the work console at the unit where the system manager is installed and execute the `pdprgcopy` command.
4. Check that the HiRDB to be updated is online.
- Choose **Start, Programs, HiRDBSingleServer** or **HiRDBParallelServer**, and then **HiRDB Command Prompt** (if installation was performed by setting the setup identifier, use the `HiRDBSingleServer` setup identifier or the `HiRDBParallelServer` setup identifier) to activate the work console of the operating HiRDB, and use the `pdls` command to check that all units display **ACTIVE**. After checking, close the HiRDB command prompt.
5. Close other programs.
- Close all programs that are accessing the HiRDB directory of the currently operating HiRDB (Windows Explorer, command prompt, etc.), programs accessing the registry (`regedit`, etc.), and HiRDB commands.
6. Update to the HiRDB update version.
- On the work console of the HiRDB update version, execute the `pdprgrenew` directory command of the operating HiRDB to update 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 3 to the update directory. For a HiRDB/Parallel Server, execute the `pdprgrenew` command on the work console

of the unit where the system manager is installed.

7. Terminate the HiRDB update version service (HiRDB/Parallel Server only).
Choose **Control Panel**, and then **Services** to terminate the HiRDB update service **HiRDB/ParallelServer-RNW**.
8. Uninstall the HiRDB update version.
Choose **Start, Programs, HiRDBSingleServer-RNW** or **HiRDBParallelServer-RNW**, and then **HiRDB Single Server Uninstall** or **HiRDB Parallel Server Uninstall** to uninstall the HiRDB update version that was installed in step 1. If any files or directories in the HiRDB update version directory are not deleted during uninstallation, you must delete them manually.
9. Set OS environment file (HiRDB/Parallel Server only).
Delete from the `SERVICES` file
(`%windir%\system32\drivers\etc\SERVICES`) the connecting port service name and port number specified in *1.4.3 Actions performed prior to updating*.

1.4.5 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. Perform overwrite installation of the HiRDB update version on the standby system.

5. Re-start the standby system terminated in step 1. On the standby HiRDB system, execute the `pdstart` command (for a HiRDB/Parallel Server, 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 manual *HiRDB Version 8 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 `pdprgnew` 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. On the standby system, perform overwrite installation of the HiRDB update version.

1.4.6 Cautions

■ When you cannot update to the HiRDB update version

In the following cases, an operating HiRDB cannot update to the HiRDB update version:

- In the case of a HiRDB/Parallel Server, when not all servers and units have terminated (including reduced mode units) or when network communication between units is blocked
- When a transaction or utility is still operating more than 15 minutes after the `pdprgnew` command was executed

- When the HiRDB Datareplicator linkage facility is used, and the setting of the `pd_rpl_init_start` operand and the actual data extraction mode differ as follows:
 - Y is the setting for the `pd_rpl_init_start` operand, and the `pdrplstop` command was executed to stop the HiRDB Datareplicator linkage facility
 - N is the setting for the `pd_rpl_init_start` operand, and the `pdrplstart` command was executed to start the HiRDB Datareplicator linkage facility
- 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 manual *HiRDB Version 8 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.

1.4.7 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 with the `pdstend` command in an environment that was activated where the `pd_statistics` operand was specified as `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 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 manual *HiRDB Version 8 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.
- When updating to the HiRDB update version, the service of the updating HiRDB and the Microsoft Cluster Server (MSCS) resource are terminated temporarily or are taken offline.

1.4.8 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.

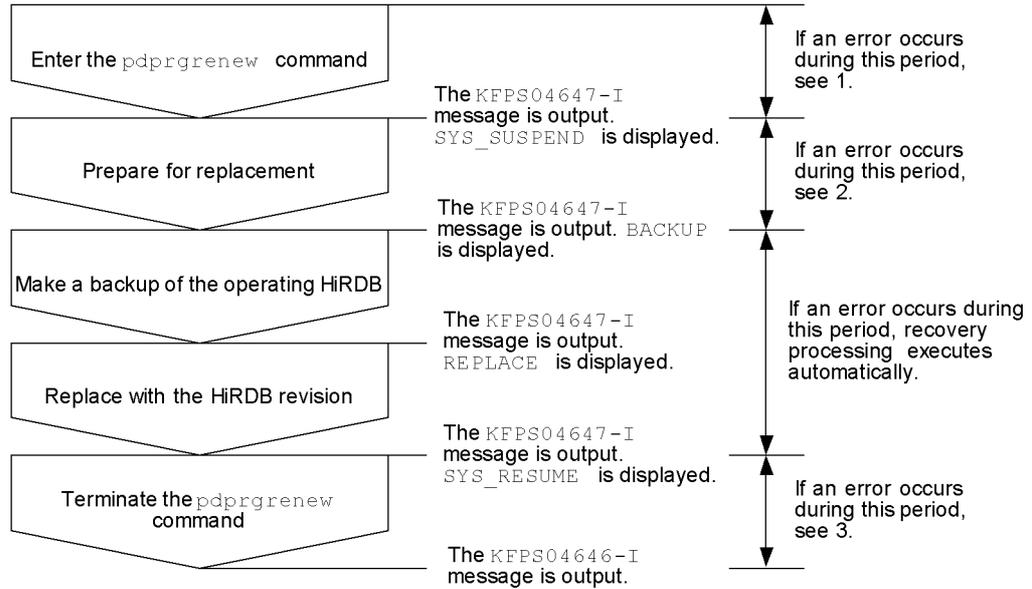
1.4.9 Operation when an error occurs during installation

(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 event log for guidance on what to do next.

Figure 1-5 shows how to handle errors during updating to the HiRDB update version.

Figure 1-5: Handling errors during updating to the HiRDB update version



1. Correct whatever caused the error in the `pdprgnew` command, and then re-execute the `pdprgnew` command.
2. If there are HiRDB processes, use the `pdstop -f` command to perform forced termination of HiRDB, and then execute the `pdprgnew -b` command. If there are no HiRDB processes, execute the `pdprgnew -b` command. When the `pdprgnew -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 are HiRDB processes, use the `pdstop -f` command to perform forced termination of HiRDB, and then execute the `pdprgnew -b` command. If there are no HiRDB processes, execute the `pdprgnew -b` command. When the `pdprgnew -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 or not 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`).

Chapter

2. Installation

This chapter describes the preparations required before installation, the HiRDB installation procedure, notes about the installation of option program products, and uninstallation of HiRDB.

This chapter contains the following sections:

- 2.1 Pre-installation procedure
- 2.2 HiRDB installation procedure
- 2.3 Post-installation procedures
- 2.4 Notes about option program products installation
- 2.5 HiRDB uninstallation procedure

2.1 Pre-installation procedure

This section describes the procedures that must be completed before HiRDB can be installed. The topics include:

- Checking the server machine environment
- Registering the HiRDB administrator
- Setting up the OS environment files

2.1.1 Checking the server machine environment

Before installing HiRDB on Windows, check the environment of the server machine to be used.

(1) *Checking the supported machines*

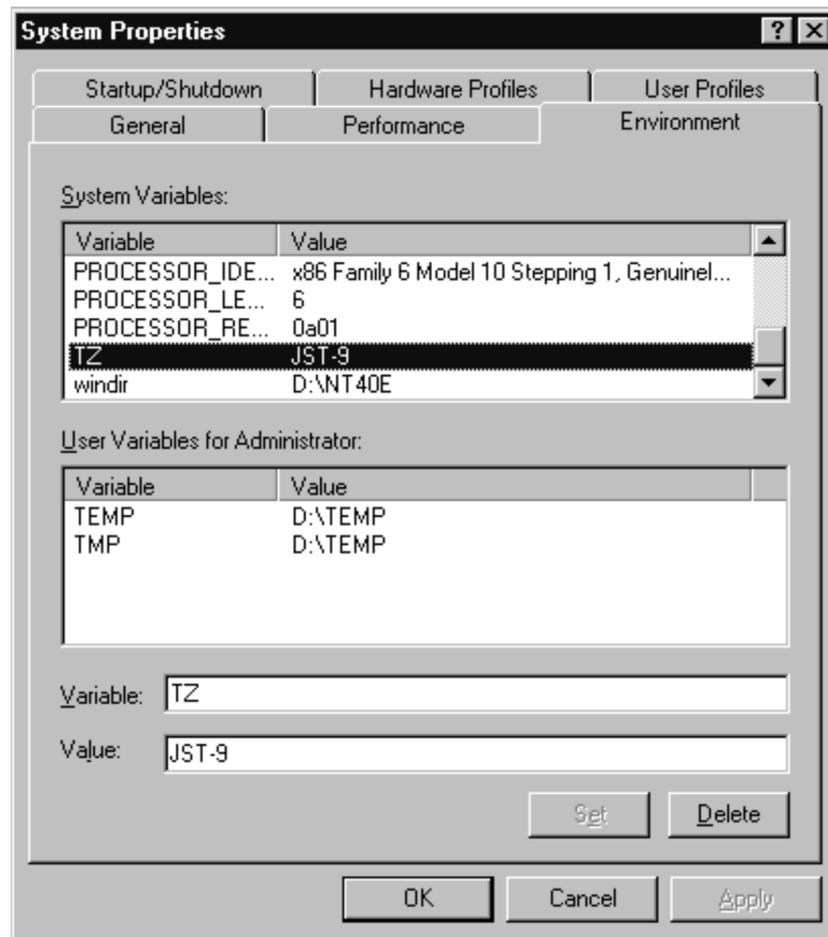
- HiRDB supports PC/AT-compatible machines running Windows. Make sure that your Windows is running on a PC/AT-compatible machine.
- For a HiRDB/Parallel Server, use the same version of Windows on all server machines.
- A single domain format is recommended for the HiRDB/Parallel Server installation.

(2) *Checking the TZ system environment variable*

Check to see if `JST-9` is set in the server machine's TZ system environment variable.

To check the TZ system environment variable:

1. In the **Control Panel** window, double-click **System**.
2. In the **System Properties** dialog box, choose the **Environment** tab.
3. Check to see if the value of system environment variable `TZ` is `JST-9`.
4. If you cannot find `TZ`, add it. If the value of `TZ` is not `JST-9`, change it to `JST-9`.



Note

The value of the TZ system environment variable must match the specified HiRDB system definitions (value of the TZ operand in the system common definitions). If you are specifying a value other than JST-9 for TZ, you need to change the value of the TZ operand in the system common definitions accordingly.

(3) Checking the disk size

The installer also checks the disk space, but make sure that there is enough free disk space before starting the installation. Guidelines for required disk space are as follows:

Space required for installation + space secured automatically when HiRDB operates

Space required for installation:

- For HiRDB/Single Server: 93.4MB
- For HiRDB/Parallel Server: 95.9MB

This value may fluctuate depending on the version. See the memory and disk requirements in the Readme files.

Space secured automatically when HiRDB operates: 1MB + shared memory size

For details about estimating the shared memory size, see *15. Storage Requirements for HiRDB*.

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 not have specified (using `pd_large_file_use`) 2048 MB or more space for the HiRDB file system area.

(4) Checking the virtual memory

To check the virtual memory size of the server machine (the total of the real memory size + paging file size), in the **Control Panel**, double-click **System** to display the System Properties dialog box. In the box, choose the **Performance** tab to check the paging file size. For the paging file, specify the same value (fixed value) for the initial size and maximum size so that continuous areas are created on the same drive. If the paging file cannot be used continuously, HiRDB terminates abnormally due to a lack of memory.

Tables 2-1 and 2-2 list guidelines for required virtual memory. The actual value must include the sizes used by Windows and other programs. If you have changed the virtual memory size, be sure to restart Windows.

Table 2-1: Required virtual memory size (HiRDB/Single Server)

Item	Memory requirement (unit: MB)
HiRDB/Single Server system processes	207.1 or more
Process private area for concurrently executable user processes under HiRDB/Single Server	4.5 or more per user

Virtual memory size (MB) = {207.1 + (4.5 x n)} + m

n: The number of concurrently executable users who operate under HiRDB

m: 0 when shared memory is allocated to files under the HiRDB directory; the shared memory size when shared memory is allocated to the paging file

Table 2-2: Required virtual memory size (HiRDB/Parallel Server)

Item	Memory requirement (unit: MB)
HiRDB/Parallel Server system processes	292.5 or more
Process private area for server processes under HiRDB/Parallel Server	10.8 or more per front-end server 14.0 or more per dictionary server 14.0 or more per back-end server
Process private area for user processes under HiRDB/Parallel Server	5.5 or more per front-end server 5.5 or more per dictionary server 5.5 or more per back-end server

Virtual memory size (MB) =

$$\{292.5 + (10.8 \times x) + (14.0 \times (y + z)) + (5.5 \times (x + y + z)) \times n\} + m$$

x: The number of front-end servers that operate under HiRDB

y: The number of dictionary servers that operate under HiRDB

z: The number of back-end servers that operate under HiRDB

n: The number of concurrently executable users who operate under HiRDB

m: 0 when shared memory is allocated to files under the HiRDB directory; the shared memory size when shared memory is allocated to the paging file

Notes on the virtual memory size

- The value may vary according to versions. See the memory and disk space requirements in the README file.
- Use the `pdntenv` command to specify where the shared memory is to be allocated, either to files under the HiRDB directory or to the paging file. For details about the specification method, see the manual *HiRDB Version 8 Command Reference*.
- For details about the memory requirement for HiRDB Datareplicator, see the manual *HiRDB Datareplicator Version 8*.
- For details about the memory requirement for HiRDB Dataextractor, see the manual *HiRDB Dataextractor Version 8*.

(5) Checking the system cache

The system cache for the files used with Windows may limit the available memory depending on the specification method.

To change the system cache setting:

1. In the **Control Panel** window, double-click **Network**.

2. Installation

2. In the **Network** dialog box, choose the **Services** tab.
3. Select **Server** and choose **Properties**.
4. In the **Server** dialog box, if **Maximize Throughput for File Sharing** is selected, select **Maximize Throughput for Network Applications**, then choose the **OK** button.
5. Restart Windows.

(6) *Checking the system files*

- HiRDB supports NTFS and FAT for file systems. For FAT, pathnames are not case sensitive. Therefore, if there are multiple files with the same name, operation cannot be guaranteed. For details about NTFS and FAT, see the applicable Windows documentation.
- If installing HiRDB on NTFS, do not compress files in the HiRDB directory (specified in the `PDDIR` environment variable). If this directory is compressed, HiRDB cannot operate normally.

(7) *Estimating the environment variables related to the number of resources*

Estimate the number of resources to be used per server machine. For details about the estimating method, see 20. *Determining Environment Variables Related to the Number of Resources*.

2.1.2 Registering the HiRDB administrator

Register a Windows system administrator (user with the Administrators privilege) as the HiRDB administrator. For a HiRDB/Parallel Server, or for a HiRDB/Single Server that is using the system switchover facility, the same system administrator must be used for every server machine.

Note that you should not use reserved words for HiRDB as authorization identifiers. For details about reserved words, see the following manuals:

- *HiRDB Version 8 UAP Development Guide*
- *HiRDB Version 8 SQL Reference*

2.1.3 Setting up the OS environment files

For a HiRDB/Parallel Server, or for a HiRDB/Single Server that is using the system switchover facility, it is necessary to connect to the remote shell. To do this, you need the communication port's service name* and the port number for connecting the client machine's remote shell in the `SERVICES` file

`windir%\system32\drivers\etc\services` for the server machine and the client machine that issues commands. The specified port number must be the same at each machine where HiRDB is installed. The number must be unique within the same machine. The following shows a specification example:

Example

```
pdrshsrv 29999/tcp
```

- Specifies `pdrshsrv` as the communication's service name.
- Specifies any port number. The default value is `29999`.

* When configuring multiple HiRDB systems using multi-HiRDB, you can specify a service name for a communication port for each HiRDB system. For details about the specification method, see *10.1 System design for a multi-HiRDB*.

For a HiRDB/Single Server that is using the system switchover facility, you must use the `-ro` option of the `pdntenv` command to specify that remote system commands are to be used. For details about the `pdntenv` command, see the manual *HiRDB Version 8 Command Reference*.

2.2 HiRDB installation procedure

Executor: Superuser

This section explains the HiRDB installation procedure. The topics include:

- Notes before installation
- HiRDB server installation procedure
- Environment variables set after installation
- Notes after installation
- In the event of error 1073 during installation

2.2.1 Notes before installation

1. The user with the Administrators privilege can install HiRDB.
If a user without the Administrators privilege attempts to install HiRDB, an error dialog is displayed and installation is cancelled.
2. When the power is turned on, the server cannot be started because the network drive is not connected. Therefore, do not install HiRDB on a network drive.
3. If you are uninstalling HiRDB once and then installing it again, be sure to restart Windows after the uninstallation. If you install without restarting Windows, error 1073 is displayed immediately after installation finishes. For details about the action to be taken in this case, see *2.2.5 What to do if error 1073 occurs during installation*.

2.2.2 HiRDB installation procedure

To install HiRDB:

1. Before installing HiRDB, terminate all Windows applications.
2. Execute `hcd_inst.exe` on the integrated CD-ROM to start Hitachi Integrated Installer.
3. In the Hitachi Integrated Installer window, select **HiRDB/Single Server** for a HiRDB/Single Server or **HiRDB/Parallel Server** for a HiRDB/Parallel Server, then click **Install**. The HiRDB installer starts.
4. In the HiRDB installer's Choose Program Product window, select **HiRDB/Single Server** for a HiRDB/Single Server or **HiRDB/Parallel Server** for a HiRDB/Parallel Server, then click **Next**. The installer for the selected program product starts.
5. In the **Setup Type** window, select a setup type. Normally, select **Typical Setup**.

When using multi-HiRDB, select **Setup with Identifier**. For details about multi-HiRDB, see *10.1.1 Installing a multi-HiRDB*.

6. In the **Specify Setup Identifier** window, specify the setup identifier. A setup identifier can be expressed as 1-4 alphanumeric characters. If you have selected **Typical Setup**, the **Specify Setup Identifier** window is not displayed.
7. In the **User Information** window, enter the user's name and company name.
8. In the **Choose Destination Location** window, specify the installation directory. The default value is as follows:

(Hereafter, this chapter assumes C: as the Windows installation drive.)

Windows installation drive for HiRDB/Single Server:

```
\win32app\hitachi\hirdb_s
```

Windows installation drive for HiRDB/Parallel Server:

```
\win32app\hitachi\hirdb_p
```

If installation is performed by specifying the setup identifier, the directory name becomes `hirdb_s` or `hirdb_p` suffixed with the setup identifier.

Example:

```
Windows-target-installation-drive\win32app\hitachi\hirdb_sUNT1
```

If you are using the default value, choose **Next**.

To change the installation directory, note the following:

- Specify a character string that begins with the drive letter and consists of only the following characters:

Alphanumeric characters

_ (underscore)

. (period)

Δ (space)

((left parenthesis)

) (right parenthesis)

\ (backslash for separating paths)

- Do not specify the drive letter only.
- A directory name cannot contain any double-byte character or special character.
- The pathname of the installation directory must be no longer than 200 characters (bytes).

9. For Windows XP Service Pack 2 or later, use the Register to the Windows Firewall window to select whether or not the HiRDB is to be added to the Windows Firewall exception list.
10. In the **Start Copying Files** window, check the specified information. If the information is fine, choose **Next**. Installation begins.
11. When installation is completed, the **Setup Complete** window is displayed on the screen.

To restart the system, select **Yes, I want to restart my computer now.**, and then choose **Finish**.

To install another program product or to not restart the system immediately, select **No, I will restart my computer later.**, and then choose **Finish**. Be sure to restart Windows before using HiRDB.

2.2.3 Environment variables set after installation

When installation is completed, the environment variables listed in Table 2-3 are set. To check the environment variables, in the **Control Panel**, double-click **System**, then in the **System Properties** dialog box, choose **Environment**.

Table 2-3: System environment variables to be set

Environment variable	Information to be specified
PDDIR*	HiRDB directory (installation directory) is set. Example: C:\win32app\hitachi\hirdb_s
PDCONFPATH*	The directory for storing HiRDB system definition files is set. To specify PDCONFPATH in the unit control information definition, specify exactly the same information, which is case sensitive. Example: C:\win32app\hitachi\hirdb_s\CONF
PATH	%PDDIR%\CLIENT\UTL and %PDDIR%\BIN are added. Example: C:\win32app\hitachi\hirdb_s\CLIENT\UTL C:\win32app\hitachi\hirdb_s\BIN
PDUXPLDIR	HiRDB's work directory is set. Example: C:\win32app\hitachi\hirdb_s\UXPLDIR

*

The PDDIR absolute path name can contain a maximum of 200 bytes and the PDCONFPATH absolute path name can contain a maximum of 213 bytes.

Note that the LANG environment variables are not set and must be set by the user. For details, see the manual *HiRDB Version 8 UAP Development Guide*.

Note

If the value of `PATH` is 512 bytes or greater, the environment variables listed in Table 2-3 are not set correctly. In this case, check the values of the environment variables and correct them, if necessary.

2.2.4 Notes after installation

1. For a HiRDB/Parallel Server, or for a HiRDB/Single Server that is using the system switchover facility, you must specify the following:
 - Service startup identifier: Automatic
 - Logon account used for executing the service: See the operating system documentation for details about how the HiRDB administrator specifies this.
2. After installing HiRDB, be sure to restart Windows.
3. When HiRDB is installed, the `tmp` directory is created immediately below the installation drive. This directory is used by HiRDB; do not delete it.
4. *Shift JIS Kanji (SJIS)* has been set as the character codes used by HiRDB.

2.2.5 What to do if error 1073 occurs during installation

If you uninstall HiRDB once and then install it again, be sure to reboot the computer after uninstallation. If you install without rebooting, error 1073 is displayed immediately after installation.

To handle error 1073:

1. Restart Windows so that the registry and service registration information existing during uninstallation take effect.
2. Install HiRDB normally and restart Windows. Specify the same directory that resulted in the error. You cannot start HiRDB yet because the service has not been registered.
3. Uninstall HiRDB*. In this case, DLL and EXE files are retained as is.
4. Delete the remaining files. To restore the files later, save the files under the `CONF` directory that contains the definition information. Then, delete the installation directory (if you have used the default values for installation, this is `C:\win32app\hitachi\hirdb_s` for a HiRDB/Single Server and `C:\win32app\hitachi\hirdb_p` for a HiRDB/Parallel Server.)
5. Restart Windows.
6. Install HiRDB normally, and then restart Windows.
7. Copy the saved HiRDB system definition files to the `%PDDIR%\conf` directory.
8. Prepare HiRDB for startup (such as system file allocation and initialization), and then start HiRDB.

2. Installation

Steps 1 through 5 restore erroneous registry and service registration to normal status (initial status). Steps beginning with 6 are normal installation procedure.

* During uninstallation, the client's definition information file `hirdb.ini` is also deleted; therefore, save this file, if necessary.

2.3 Post-installation procedures

This section describes post-installation procedures.

2.3.1 Selection of the character codes

To use single-byte character codes, Unicodes (UTF-8), or Chinese EUC codes, use the `pdntenv` command to change the setting for the character codes to be used.

The Unicode set (UTF-8) cannot be specified for versions prior to HiRDB Text Search Plug-in Version 2 02-02. Therefore, if you are using HiRDB in conjunction with a version of HiRDB Text Search Plug-in Version 2 that is prior to 02-02, do not specify UTF-8.

Note

To change the character codes once a database has been constructed, it is necessary to reconstruct the database.

2.3.2 Deleting files from the HiRDB directory

When a server process or client has been terminated forcibly, HiRDB outputs troubleshooting information under `%PDDIR%\spool`. If a command or utility is terminated by pressing **Ctrl + C**, temporary work files output by the command or utility remain in `%PDDIR%\tmp` and are not deleted. These remaining troubleshooting information items and temporary work 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`)

Normally, these files are deleted every 24 hours. The interval between deletions can be changed with the `pd_spool_cleanup_interval` operand. The `pd_spool_cleanup_interval_level` operand can be used to specify that only files that were output earlier than a specified number of days previous to the current date are to be deleted.

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 or not to delete the troubleshooting information files. If you omit this operand, the

troubleshooting information files are deleted automatically.

You can also use the `pd_spool_cleanup_level` operand to delete only those troubleshooting information files that have been output a specified number of days ago.

Note

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

2.4 Notes about option program products installation

This section explains the rules for installing option program products and provides notes on the installation of option program products.

If the HiRDB option program products facility is to be used, its version must be the same as the HiRDB version. Table 2-4 shows the option program products facility and the servers on which each option program product is installed.

Table 2-4: Option program products facility and target installation servers

Product name	Facilities available with introduction of the option program product	Target installation server machine
HiRDB Advanced High Availability	<ul style="list-style-type: none"> • Server mode system switchover facility • User server hot standby • Rapid system switchover facility • Standby-less system switchover (1:1) facility • Standby-less system switchover (effects distributed) facility 	All server machines
	Dynamic update of global buffer (pdbufmod command)	
	System reconfiguration command (pdchgconf command)	
HiRDB Advanced Partitioning Option	Table matrix partitioning	
	Changing the partition storage conditions (ALTER TABLE)	
HiRDB Non Recover FES	Recovery-unnecessary front-end server	
HiRDB LDAP Option	Sun Java System Directory Server linkage facility	Server machine containing the dictionary server

(1) Rules and notes

1. The option program products are provided as key modules (the HiRDB LDAP Option is provided as a library).
2. An option program product key module is copied directly below the HiRDB installation directory. The HiRDB LDAP Option library is copied below the %PDDIR%\hirdb_ldap_sods directory.
3. To install an option program product, the HiRDB service must be in terminated status. If the HiRDB service has been started, the installation will terminate.

2. Installation

4. If you have upgraded HiRDB, you need to install the option program product for the new HiRDB version. If you have changed the HiRDB installation directory, you need to reinstall option program product.
5. Only a user in the Administrators group can install option program product.
6. If HiRDB is not installed, option program product cannot be installed.
7. When uninstalling option program product, make sure that the HiRDB service is stopped. If option program product is uninstalled, all the information that was copied and specified during installation is deleted.
8. To install option program product on a HiRDB with a setup identifier, specify the same setup identifier as the corresponding HiRDB.

2.5 HiRDB uninstallation procedure

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

(1) *Checking the status of the unit from which HiRDB is to be uninstalled*

Before uninstalling HiRDB, check the status of the unit from which the HiRDB is to be uninstalled. On the applicable unit, execute the `pdls -d ust` command.

- If the termination status is 0 (unit status is ONLINE):
HiRDB is running. Use the `pdstop` command to terminate HiRDB normally.
- If the termination status is 4 (unit status is STARTING or STOPPING):
HiRDB start or termination processing is underway. After the processing is completed, re-execute the `pdls -d ust` command.
- If the termination status is 8 (unit status is PAUSE):
Restart of the process service has been suspended due to an error. To uninstall HiRDB in this status, proceed to (3) *Uninstalling HiRDB*.

If you do not wish to uninstall HiRDB, see the KFPS00715-E message and the message issued to the event log before this message, eliminate the cause of the error, start the service, then restart the unit.

- If the termination status is 12 (unit status is STOP):
HiRDB has stopped. Stop the service.
- If the termination status is 16 (unit status is UNSETUP):
You can uninstall HiRDB.

(2) *Stopping the HiRDB service*

Choose **Control Panel**, then **Services** to determine whether or not the HiRDB service to be uninstalled has started. If the HiRDB service has started, stop it, then uninstall the HiRDB.

(3) *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.

(a) **Uninstalling HiRDB**

To uninstall a HiRDB/Single Server:

1. From **Start**, choose **Programs**, **HiRDBSingleServer**, then **Uninstall**

HiRDBSingleServer.

If **Uninstall HiRDBSingleServer** is not registered for the version, open **Control Panel**, choose **Add/Remove Programs**, then **Install/Uninstall**. From the list, select **HiRDBSingleServer**, and choose **Add/Remove**, then the **OK** button.

2. Uninstallation begins. After a while, the message **Do you want to delete shared files?** may be displayed. In this case, choose **No**.
3. When uninstallation is completed, the message **Reinstall HiRDB/Single Server after restarting the computer** is displayed. Choose the **OK** button.
4. In the **Remove Program from Computer** dialog box, choose the **OK** button. Uninstallation is now completed.

The following table describes the possible causes of uninstallation errors and the actions to be taken:

Possible cause	Action
A command, utility, HiRDB Datareplicator, or HiRDB Dataextractor is running.	Terminate the command, utility, HiRDB Datareplicator, or HiRDB Dataextractor that is running, then re-execute uninstallation.
Other (such as an error)	See the message output to the event log, eliminate the cause of the error, then re-execute uninstallation.

5. Even after HiRDB is uninstalled, the directories and files created after installation remain in the installation directory. If you were using the default installation directory, use a program such as Explorer to delete the `hirdb_s` subdirectory from the `C:\win32app\hitachi` directory. If you have not uninstalled HiRDB Control Manager - Agent, uninstall it and then delete the `hirdb_s` subdirectory.

(4) Uninstalling a HiRDB/Parallel Server

To uninstall a HiRDB/Parallel Server, execute the uninstallation procedure at all server machines that constitute the HiRDB/Parallel Server.

To uninstall a HiRDB/Parallel Server:

1. From **Start**, choose **Programs, HiRDBParallelServer**, then **Uninstall HiRDBParallelServer**.

If **Uninstall HiRDBParallelServer** is not registered for the version, open **Control Panel**, choose **Add/Remove Programs**, then **Install/Uninstall**. From the list, select **HiRDBParallelServer**, and choose **Add/Remove**, then the **OK** button.

2. Uninstallation begins. After a while, the message **Do you want to delete shared files?** may be displayed. In this case, choose **No**.
3. When uninstallation is completed, the message **Reinstall HiRDB/Parallel**

Server after restarting the computer is displayed. Choose the **OK** button.

4. In the **Remove Program from Computer** dialog box, choose the **OK** button. Uninstallation is now completed.

The following table describes the possible causes of uninstallation errors and the actions to be taken:

Possible cause	Action
A command, utility, HiRDB Datareplicator, or HiRDB Dataextractor is running.	Terminate the command, utility, HiRDB Datareplicator, or HiRDB Dataextractor that is running, then re-execute uninstallation.
Other (such as an error)	See the message output to the event log, eliminate the cause of the error, then re-execute uninstallation.

5. Even after HiRDB is uninstalled, the directories and files created after installation remain in the installation directory. If you were using the default installation directory, use a program such as Explorer to delete the `hirdb_p` subdirectory from the `C:\win32app\hitachi` directory. If you have not uninstalled HiRDB Control Manager - Agent, uninstall it and then delete the `hirdb_p` subdirectory.

Chapter

3. Setting Up an Environment Using the Simple Setup Tool

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

3.1 Overview of the simple setup tool

3.1 Overview of the simple setup tool

The simple setup tool is not supported.

Chapter

4. Setting Up an Environment Using Commands

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

This chapter contains the following sections:

- 4.1 Overview of environment setup using commands
- 4.2 Creating the HiRDB system definitions
- 4.3 Creating HiRDB file system areas
- 4.4 Creating system files
- 4.5 Creating system RDAREAs
- 4.6 Starting HiRDB for the first time
- 4.7 Creating user RDAREAs
- 4.8 Creating user LOB RDAREAs
- 4.9 Creating data dictionary LOB RDAREAs
- 4.10 Creating list RDAREAs

4.1 Overview of environment setup using commands

(1) Items to be defined before environment setup

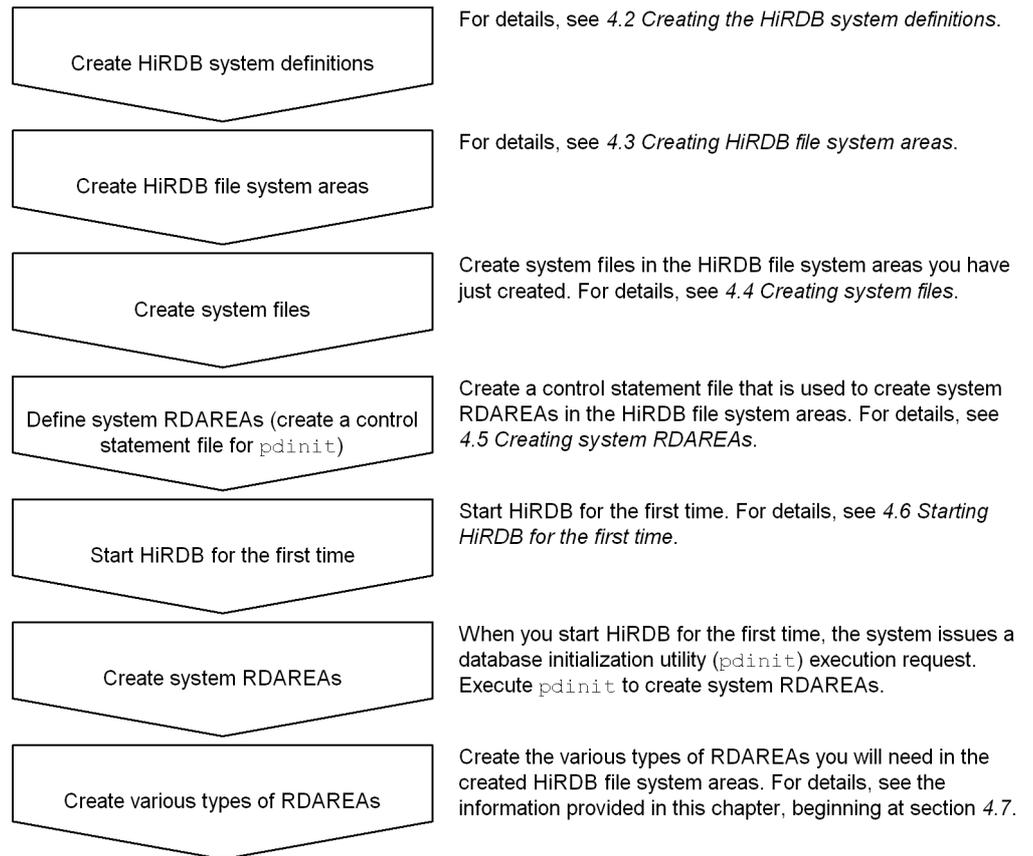
Before starting the HiRDB environment setup, design the system configuration for the following items:

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

Determine the configuration of these items by referring to 8. *Designing a HiRDB/Single Server*, or 9. *Designing a HiRDB/Parallel Server*. 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

Figure 4-1 shows 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 RDAREAs), data directory RDAREAs, and data dictionary RDAREAs, 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 RDAREAs for lists).

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
- Modifying HiRDB system definitions
- Modifying UAP environment definitions

For details about the HiRDB system definition operands, see the manual *HiRDB Version 8 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 8 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)

(1) Creating system common definitions (HiRDB/Single Server)

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.

(2) Creating unit control information definitions (HiRDB/Single Server)

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.

(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

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

* 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-name1`

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.

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.

¹ The file name must begin with an alphabetic character and must not exceed 8 characters in length. File names are not case sensitive. For example, the characters A and a are treated as being identical. The file names ABC and abc are the same file name.

(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**

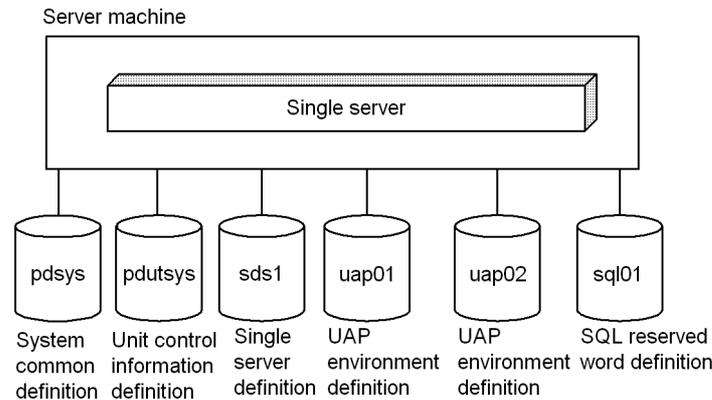
The HiRDB administrator must grant to a user who uses the SQL reserved word definition the read (r) and execution (x) privileges for the %PDDIR%\conf\pdrsvwd directory and the read privilege (r) for the SQL reserved word deletion file.

* The file name must be expressed as no more than eight alphanumeric characters beginning with an alphabetic character. The file name is not case-sensitive. For example, A is not distinguished from a, so the file names ABC and abc are regarded as the same.

(6) Configuring the HiRDB system definition files

Figure 4-2 shows an example configuration of HiRDB system definition files created by HiRDB.

Figure 4-2: Configuration of HiRDB system definition files: HiRDB/Single Server



4.2.2 Creating HiRDB system definitions (HiRDB/Parallel Server)

(1) Creating system common definitions (HiRDB/Parallel Server)

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.

(2) Creating unit control information definitions (HiRDB/Single Server)

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.

(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.

(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

(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

(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

(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`¹

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.

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.

¹ The file name must begin with an alphabetic character and must not exceed 8 characters in length. File names are not case sensitive. For example, the characters A and a are treated as being identical. The file names ABC and abc are the same file name.

(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.

* The file name must be expressed as no more than eight alphanumeric characters beginning with an alphabetic character. The file name is not case-sensitive. For example, A is not distinguished from a, so the file names ABC and abc are regarded as the same.

(9) Foreign server information definition creation

When the HiRDB External Data Access facility is used, foreign server information definition is required. This definition defines the linkage environment for a foreign server. You create and store a foreign server information definition in the following file:

- %PDDIR%\conf*foreign-server-name*

A foreign server information definition must be created on the server machine where the back-end server that will link to the foreign server is defined.

Which operands you specify in a foreign server information definition depends on the type of foreign server to be linked; for details, see the manual *HiRDB Version 8 System Definition*.

(10) Hub optimization information definition creation

When the HiRDB External Data Access facility is used, Hub optimization information definition is required. This definition defines optimization information for the HiRDB External Data Access facility. You create and store a Hub optimization information definition in the following file:

- %PDDIR%\conf*any-name*

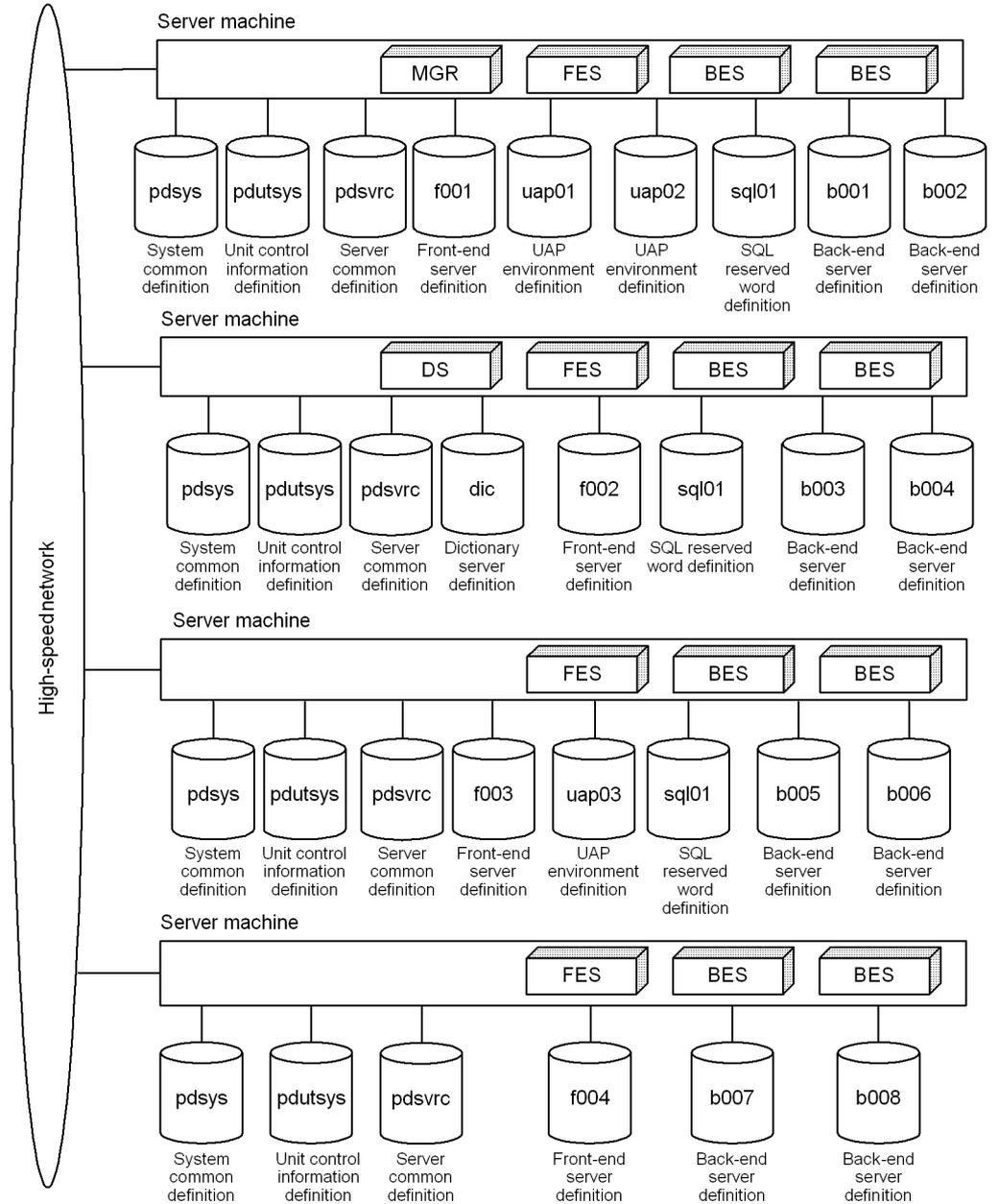
A Hub optimization information definition must be created on the server machine where the front-end server is defined. If there are multiple front-end servers, the same Hub optimization information definition is required for each unit.

(11) Configuring the HiRDB system definition files

Figure 4-3 shows an example configuration of HiRDB system definition files.

Figure 4-4 shows an example configuration of HiRDB system definition files when the HiRDB External Data Access facility is used.

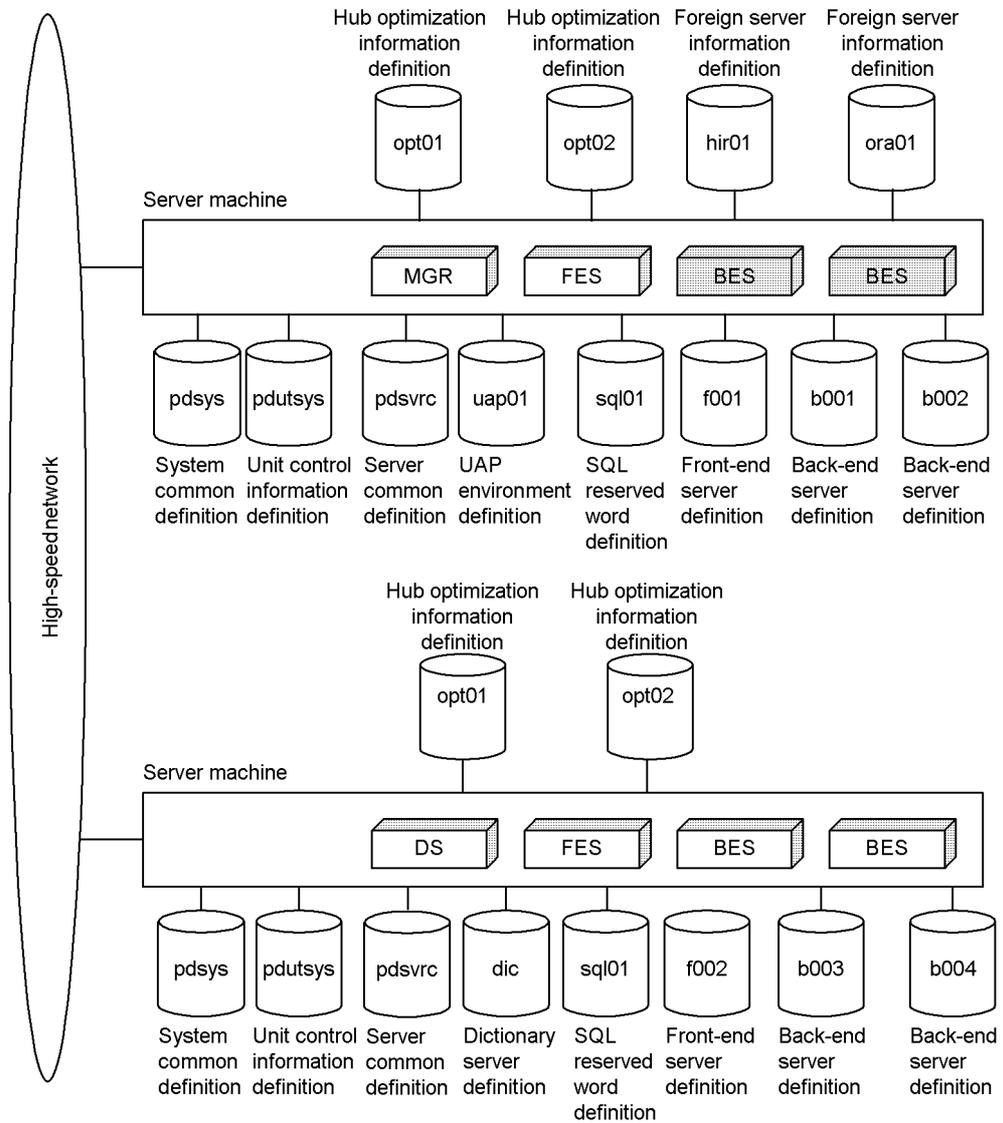
Figure 4-3: Configuration of HiRDB system definition files: HiRDB/Parallel Server



Notes

- The system common definitions must be identical at all server machines.

Figure 4-4: Configuration of HiRDB system definition files (when the HiRDB External Data Access facility is used)



Notes

- The system common definitions must be identical at all server machines.

- The networked back-end server is the back-end server to be linked with the foreign server. A foreign server information definition file must be created at the unit where the back-end server to be linked to the foreign server is installed.
- Because this is an environment with multiple front-end servers, a Hub optimization information definition file must be created on each unit.

4.2.3 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, 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.

4. Setting Up an Environment Using Commands

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\chgconf`. 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 manual *HiRDB Version 8 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 HiRDB/Parallel Server).
- 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 8 System Definition*.

4.2.4 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*).

4.3.1 Types of HiRDB file system areas

Create the HiRDB file system areas according to their purposes as shown in Table 4-1. 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	Utilities (that do not use Windows cache)	NUTL
7	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, and 9.2 *Designing HiRDB file system areas* for a HiRDB/Parallel Server.

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.

Maximum size of a HiRDB file system area

The maximum size of a HiRDB file system area is as follows:

Condition	Maximum size of HiRDB file system area
<code>pd_large_file_use=N</code> specified (default)	2,047 MB

Condition	Maximum size of HiRDB file system area
pd_large_file_use=Y specified	1,048,575 MB

4.3.2 Creating a HiRDB file system area using the raw I/O facility

You can create a HiRDB file system area that uses Windows's direct disk access (raw I/O) capabilities, which access partitions or logical drives in the same manner as files. This is called the raw I/O facility. To use the raw I/O facility, you need the preparations discussed here before using the `pdfmkfs` command.

(1) Support range of the raw I/O facility

The following table shows the range of HiRDB file system areas supported by the raw I/O facility:

Purpose ¹	Support
DB	S ²
SDB	S ²
SYS	S
WORK	B
UTL	B
SVR	B
NUTL ³	S

S: The raw I/O facility is supported.

B: The raw I/O facility is not supported.

¹ This is the purpose of the HiRDB file system area that is specified with the `-k` option of the `pdfmkfs` command.

² The following table shows whether or not the raw I/O facility is supported:

Type of RDAREA	Support
Master directory RDAREA	S
Data directory RDAREA	S
Data dictionary RDAREA	S
User RDAREA	S

Type of RDAREA	Support
Registry RDAREA	S
Data dictionary LOB RDAREA	S
User LOB RDAREA	S
Registry LOB RDAREA	S
List RDAREA	B

S: The raw I/O facility is supported.

B: The raw I/O facility is not supported.

3

The following utility files support the raw I/O facility:

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

(2) Environment setup

To use the raw I/O facility, you need an unformatted partition. To create a partition in Windows, choose **Computer Management**, and then **Disk management**. For details of how to create a partition, see *Disk Operation* in the Windows help file.

For the size of a partition, specify the size of HiRDB file system area plus 1 MB.

Notes

- Do not use a partition made up of a group of partitions (such as a volume set or stripe set).
- You need to assign a drive letter to the created partition.
- The raw I/O facility is applicable only to a fixed disk with a sector length of 512 bytes.

4.3.3 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 C:\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.

`C:\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 8 Command Reference*.

4.3.4 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 C:\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.

C:\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 8 Command Reference*.

4.3.5 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 C:\workarea01
```

Explanation:

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

For details about how to estimate the area size, see *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.

C:\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 8 Command Reference*.

4.3.6 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

Example

Create a HiRDB file system area for utilities:

```
pdfmkfs -n 50 -l 10 -k UTL -i C:\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.

`C:\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 8 Command Reference*.

4.3.7 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 C:\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.

C:\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 8 Command Reference*.

4.3.8 Example 6 (creating a HiRDB file system area that uses the raw I/O facility)

To create a HiRDB file system area that uses the raw I/O facility, specify the name of the HiRDB file system area in the following format:

\\.\drive-name:

For *drive-name*, specify the letter of the drive that has already been prepared.

Example

Create a HiRDB file system area for RDAREAs:

```
pdfmkfs -n 100 -l 50 -e 1 -k DB \\.\J:
```

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.

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

\\.\J:

Specifies that logical disk `J` is to be created as a HiRDB file system area.

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 and *9.3 Designing system files* for a HiRDB/Parallel Server.

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 (`C:\sysarea01`):

```
pdloginit -d sys -s b001 -f C:\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.

`-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 usually has a record length of 4096 bytes, but this size can be changed with the `pd_log_rec_leng` operand.

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 8 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 (`C:\sysarea01`):

```
pdloginit -d spd -s b001 -f C:\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.

`-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 4096 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 8 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 (`C:\sysarea01`):

```
pdstsinit -s b001 -f C:\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 8 Command Reference*.

Relationship with HiRDB system definitions

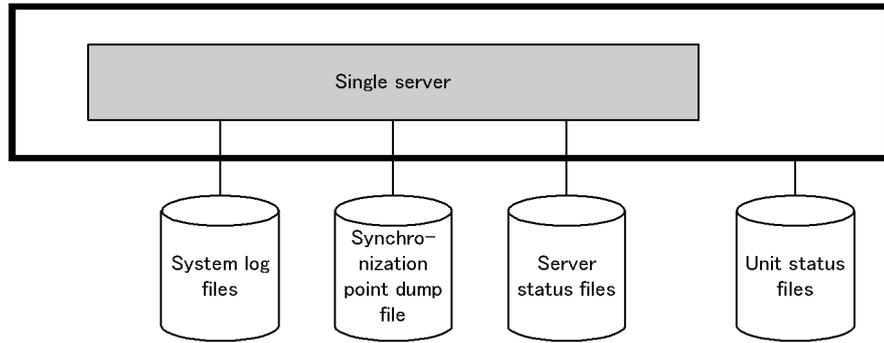
This `pdstsinit` command is associated with the following operands in the HiRDB system definitions:

- `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.

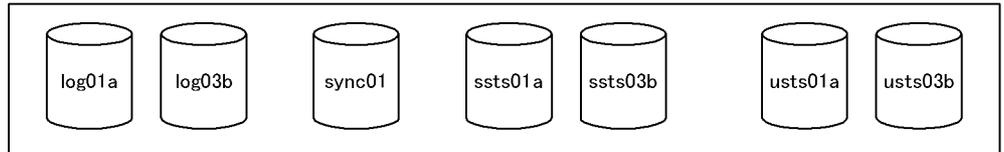
4.4.4 Example of system file creation (HiRDB/Single Server)

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

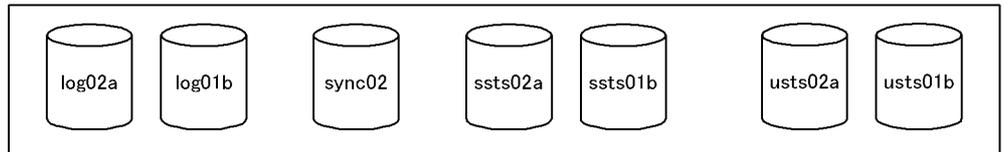


Configuration of the HiRDB file system area

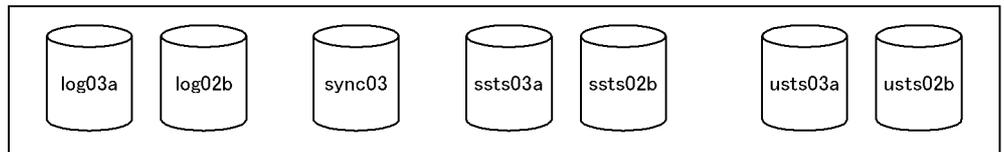
HiRDB file system area (C:\sysarea01)



HiRDB file system area (C:\sysarea02)



HiRDB file system area (C:\sysarea03)



System log files Synchronization point dump file Server status files Unit status files

(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", "C:\sysarea01\usts01a"\
, "C:\sysarea02\usts01b"
set pd_syssts_file_name_2="usts2", "C:\sysarea02\usts02a"\
, "C:\sysarea03\usts02b"
set pd_syssts_file_name_3="usts3", "C:\sysarea03\usts03a"\
, "C:\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

```
set pd_log_rec_leng=4096
pdlogadfg -d sys -g log1 ONL
pdlogadfg -d sys -g log2 ONL
pdlogadfg -d sys -g log3 ONL
pdlogadpf -d sys -g log1 -a "C:\sysarea01\log01a"\
-b "C:\sysarea02\log01b"
pdlogadpf -d sys -g log2 -a "C:\sysarea02\log02a"\
-b "C:\sysarea03\log02b"
pdlogadpf -d sys -g log3 -a "C:\sysarea03\log03a"\
-b "C:\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 "C:\sysarea01\sync01"
pdlogadpf -d spd -g sync2 -a "C:\sysarea02\sync02"
pdlogadpf -d spd -g sync3 -a "C:\sysarea03\sync03"
```

Definition example of server status files

```

set pd_sts_file_name_1="sst1", "C:\sysarea01\ssts01a"\  

, "C:\sysarea02\ssts01b"\  

set pd_sts_file_name_2="sst2", "C:\sysarea02\ssts02a"\  

, "C:\sysarea03\ssts02b"\  

set pd_sts_file_name_3="sst3", "C:\sysarea03\ssts03a"\  

, "C:\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 C:\sysarea01
pdfmkfs -n 50 -l 20 -i -k SYS C:\sysarea02
pdfmkfs -n 50 -l 20 -i -k SYS C:\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 C:\sysarea01\log01a -n 1024
pdloginit -d sys -f C:\sysarea01\log03b -n 1024
pdloginit -d sys -f C:\sysarea02\log02a -n 1024
pdloginit -d sys -f C:\sysarea02\log01b -n 1024
pdloginit -d sys -f C:\sysarea03\log03a -n 1024
pdloginit -d sys -f C:\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 C:\sysarea01\sync01 -n 64
pdloginit -d spd -f C:\sysarea02\sync02 -n 64
pdloginit -d spd -f C:\sysarea03\sync03 -n 64

```

(c) Creating the server status files

Use the `pdstsininit` command to create the server status files.

Example of command entry

```
pdstsininit -s sds1 -f C:\sysarea01\ssts01a -l 4096 -c 256
pdstsininit -s sds1 -f C:\sysarea01\ssts03b -l 4096 -c 256
pdstsininit -s sds1 -f C:\sysarea02\ssts02a -l 4096 -c 256
pdstsininit -s sds1 -f C:\sysarea02\ssts01b -l 4096 -c 256
pdstsininit -s sds1 -f C:\sysarea03\ssts03a -l 4096 -c 256
pdstsininit -s sds1 -f C:\sysarea03\ssts02b -l 4096 -c 256
```

(d) Creating the unit status files

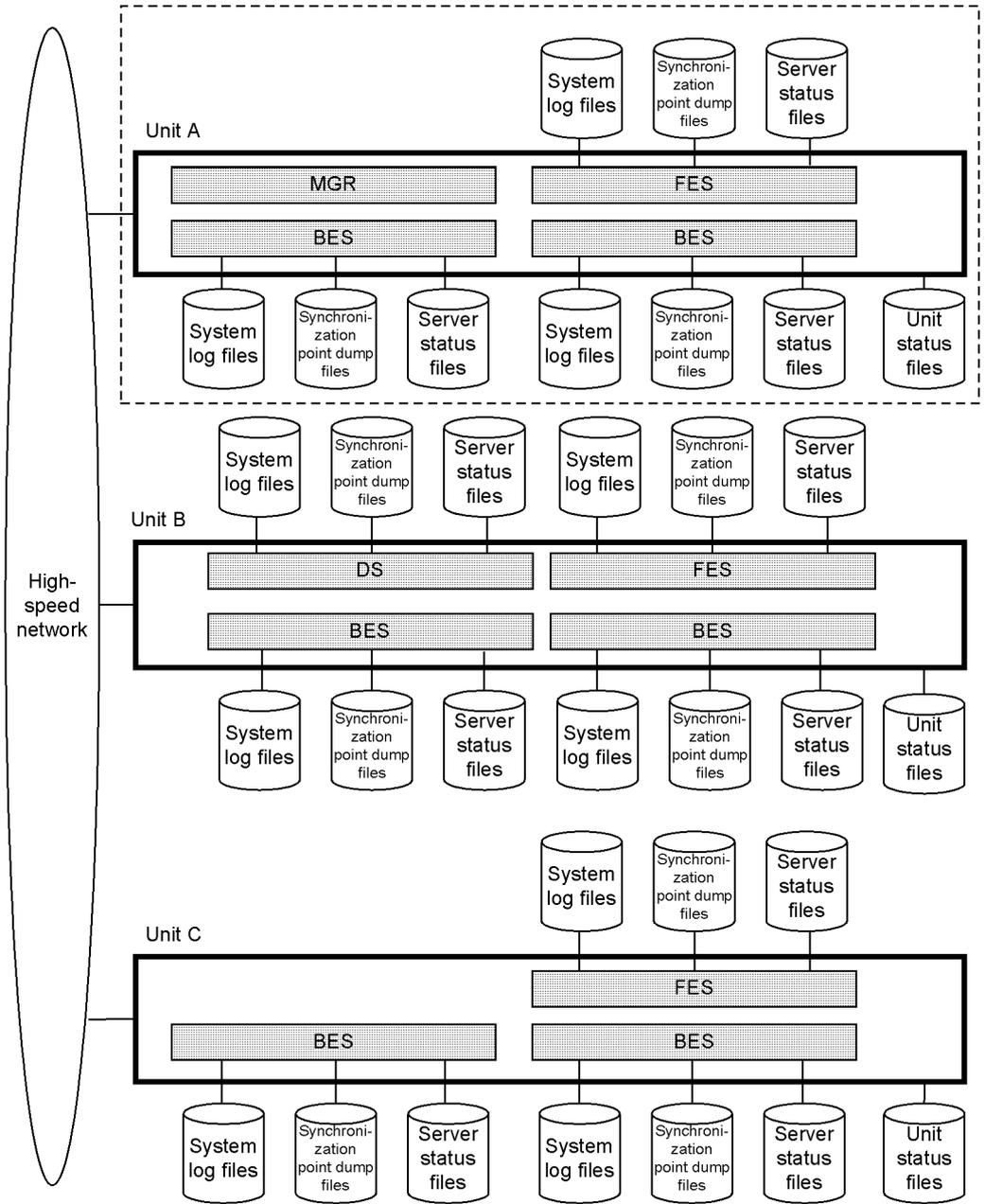
Use the `pdstsininit` command to create the unit status files.

Example of command entry

```
pdstsininit -u unt1 -f C:\sysarea01\usts01a -l 4096 -c 256
pdstsininit -u unt1 -f C:\sysarea01\usts03b -l 4096 -c 256
pdstsininit -u unt1 -f C:\sysarea02\usts02a -l 4096 -c 256
pdstsininit -u unt1 -f C:\sysarea02\usts01b -l 4096 -c 256
pdstsininit -u unt1 -f C:\sysarea03\usts03a -l 4096 -c 256
pdstsininit -u unt1 -f C:\sysarea03\usts02b -l 4096 -c 256
```

4.4.5 Example of system file creation (HiRDB/Parallel Server)

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



MGR: System manager

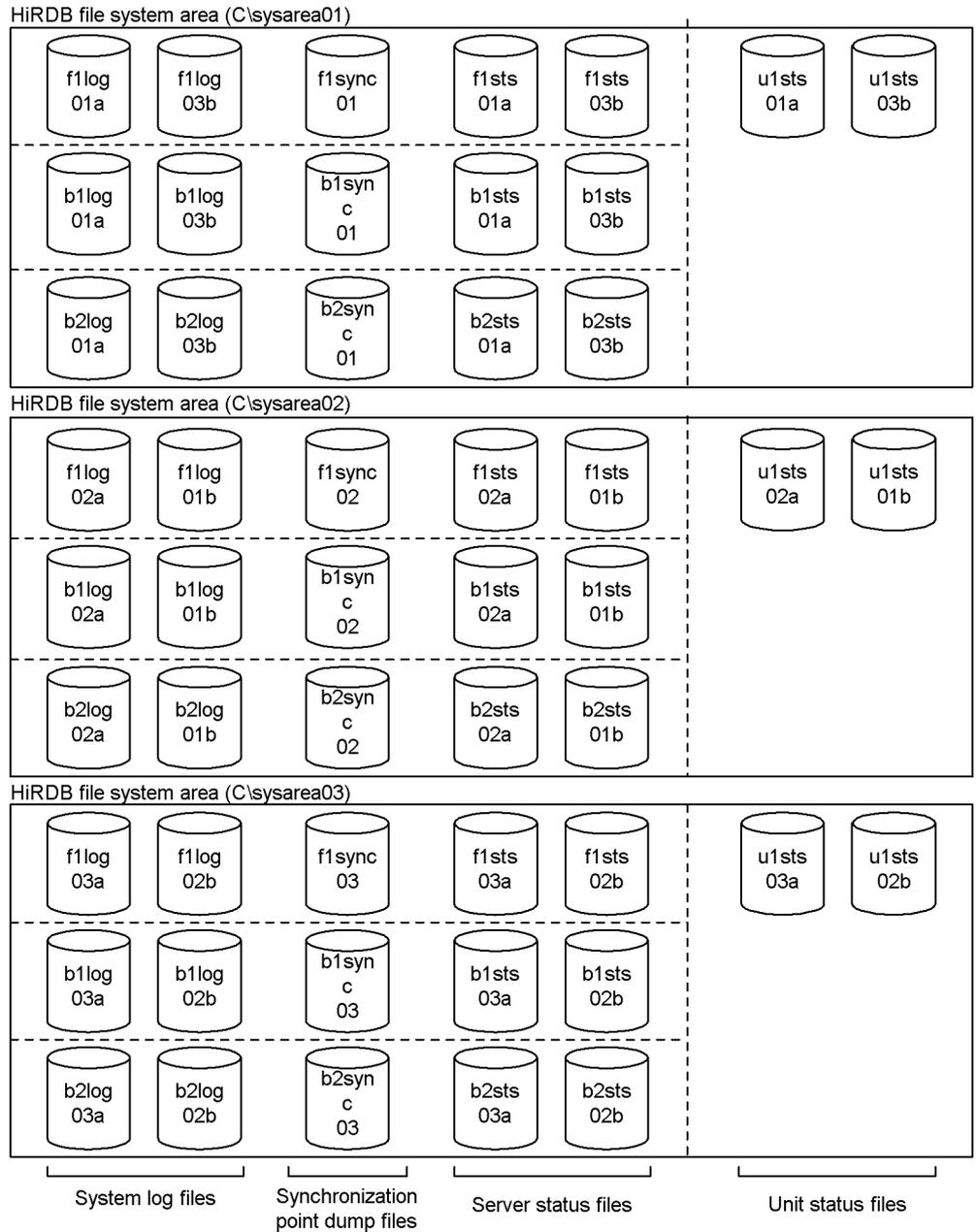
FES: Front-end server

4. Setting Up an Environment Using Commands

DS: Dictionary server

BES: Back-end server

Configuration of the HiRDB file system area



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", "C:\sysarea01\ulsts01a"\  
    , "C:\sysarea02\ulsts01b"\  
set pd_syssts_file_name_2="ulsts2", "C:\sysarea02\ulsts02a"\  
    , "C:\sysarea03\ulsts02b"\  
set pd_syssts_file_name_3="ulsts3", "C:\sysarea03\ulsts03a"\  
    , "C:\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

```
set pd_log_rec_leng=4096  
pdlogadfg -d sys -g fllog1 ONL  
pdlogadfg -d sys -g fllog2 ONL  
pdlogadfg -d sys -g fllog3 ONL  
pdlogadpf -d sys -g fllog1 -a "C:\sysarea01\fllog01a"\  
    -b "C:\sysarea02\fllog01b"\  
pdlogadpf -d sys -g fllog2 -a "C:\sysarea02\fllog02a"\  
    -b "C:\sysarea03\fllog02b"\  
pdlogadpf -d sys -g fllog3 -a "C:\sysarea03\fllog03a"\  
    -b "C:\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 "C:\sysarea01\flsync01"\  
pdlogadpf -d spd -g flsync2 -a "C:\sysarea02\flsync02"\  
pdlogadpf -d spd -g flsync3 -a "C:\sysarea03\flsync03"
```

Definition example of server status files

```

set pd_sts_file_name_1="f1sts1", "C:\sysarea01\f1sts01a"\
                                , "C:\sysarea02\f1sts01b"
set pd_sts_file_name_2="f1sts2", "C:\sysarea02\f1sts02a"\
                                , "C:\sysarea03\f1sts02b"
set pd_sts_file_name_3="f1sts3", "C:\sysarea03\f1sts03a"\
                                , "C:\sysarea01\f1sts03b"

```

(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

```

set pd_log_rec_leng=4096
pdlogadfg -d sys -g b1log1 ONL
pdlogadfg -d sys -g b1log2 ONL
pdlogadfg -d sys -g b1log3 ONL
pdlogadpf -d sys -g b1log1 -a "C:\sysarea01\b1log01a"\
                             -b "C:\sysarea02\b1log01b"
pdlogadpf -d sys -g b1log2 -a "C:\sysarea02\b1log02a"\
                             -b "C:\sysarea03\b1log02b"
pdlogadpf -d sys -g b1log3 -a "C:\sysarea03\b1log03a"\
                             -b "C:\sysarea01\b1log03b"

```

Definition example of synchronization point dump files

```

pdlogadfg -d spd -g b1sync1 ONL
pdlogadfg -d spd -g b1sync2 ONL
pdlogadfg -d spd -g b1sync3 ONL
pdlogadpf -d spd -g b1sync1 -a "C:\sysarea01\b1sync01"
pdlogadpf -d spd -g b1sync2 -a "C:\sysarea02\b1sync02"
pdlogadpf -d spd -g b1sync3 -a "C:\sysarea03\b1sync03"

```

Definition example of server status files

```

set pd_sts_file_name_1="b1sts1", "C:\sysarea01\b1sts01a"\
                                , "C:\sysarea02\b1sts01b"
set pd_sts_file_name_2="b1sts2", "C:\sysarea02\b1sts02a"\
                                , "C:\sysarea03\b1sts02b"
set pd_sts_file_name_3="b1sts3", "C:\sysarea03\b1sts03a"\
                                , "C:\sysarea01\b1sts03b"

```

(d) BES2 back-end server definitions

Define system log files, synchronization point dump files, and server status files in the

4. Setting Up an Environment Using Commands

BES2 back-end server definitions.

Definition example of system log files

```
set pd_log_rec_leng=4096
pdlogadfg -d sys -g b2log1 ONL
pdlogadfg -d sys -g b2log2 ONL
pdlogadfg -d sys -g b2log3 ONL
pdlogadpf -d sys -g b2log1 -a "C:\sysarea01\b2log01a"\
-b "C:\sysarea02\b2log01b"
pdlogadpf -d sys -g b2log2 -a "C:\sysarea02\b2log02a"\
-b "C:\sysarea03\b2log02b"
pdlogadpf -d sys -g b2log3 -a "C:\sysarea03\b2log03a"\
-b "C:\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 "C:\sysarea01\b2sync01"
pdlogadpf -d spd -g b2sync2 -a "C:\sysarea02\b2sync02"
pdlogadpf -d spd -g b2sync3 -a "C:\sysarea03\b2sync03"
```

Definition example of server status files

```
set pd_sts_file_name_1="b2sts1","C:\sysarea01\b2sts01a"\
,"C:\sysarea02\b2sts01b"
set pd_sts_file_name_2="b2sts2","C:\sysarea02\b2sts02a"\
,"C:\sysarea03\b2sts02b"
set pd_sts_file_name_3="b2sts3","C:\sysarea03\b2sts03a"\
,"C:\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 C:\sysarea01
pdfmkfs -n 50 -l 20 -i -k SYS C:\sysarea02
pdfmkfs -n 50 -l 20 -i -k SYS C:\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 C:\sysarea01\f1log01a -n 1024
pdloginit -d sys -s f001 -f C:\sysarea01\f1log03b -n 1024
pdloginit -d sys -s f001 -f C:\sysarea02\f1log02a -n 1024
pdloginit -d sys -s f001 -f C:\sysarea02\f1log01b -n 1024
pdloginit -d sys -s f001 -f C:\sysarea03\f1log03a -n 1024
pdloginit -d sys -s f001 -f C:\sysarea03\f1log02b -n 1024
```

Example of command entry (BES1)

```
pdloginit -d sys -s b001 -f C:\sysarea01\b1log01a -n 1024
pdloginit -d sys -s b001 -f C:\sysarea01\b1log03b -n 1024
pdloginit -d sys -s b001 -f C:\sysarea02\b1log02a -n 1024
pdloginit -d sys -s b001 -f C:\sysarea02\b1log01b -n 1024
pdloginit -d sys -s b001 -f C:\sysarea03\b1log03a -n 1024
pdloginit -d sys -s b001 -f C:\sysarea03\b1log02b -n 1024
```

Example of command entry (BES2)

```
pdloginit -d sys -s b002 -f C:\sysarea01\b2log01a -n 1024
pdloginit -d sys -s b002 -f C:\sysarea01\b2log03b -n 1024
pdloginit -d sys -s b002 -f C:\sysarea02\b2log02a -n 1024
pdloginit -d sys -s b002 -f C:\sysarea02\b2log01b -n 1024
pdloginit -d sys -s b002 -f C:\sysarea03\b2log03a -n 1024
pdloginit -d sys -s b002 -f C:\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 C:\sysarea01\f1sync01 -n 64
pdloginit -d spd -s f001 -f C:\sysarea02\f1sync02 -n 64
pdloginit -d spd -s f001 -f C:\sysarea03\f1sync03 -n 64
```

Example of command entry (BES1)

4. Setting Up an Environment Using Commands

```
pdloginit -d spd -s b001 -f C:\sysarea01\b1sync01 -n 64
pdloginit -d spd -s b001 -f C:\sysarea02\b1sync02 -n 64
pdloginit -d spd -s b001 -f C:\sysarea03\b1sync03 -n 64
```

Example of command entry (BES2)

```
pdloginit -d spd -s b002 -f C:\sysarea01\b2sync01 -n 64
pdloginit -d spd -s b002 -f C:\sysarea02\b2sync02 -n 64
pdloginit -d spd -s b002 -f C:\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 C:\sysarea01\f1sts01a -l 4096 -c 256
pdstsinit -s f001 -f C:\sysarea01\f1sts03b -l 4096 -c 256
pdstsinit -s f001 -f C:\sysarea02\f1sts02a -l 4096 -c 256
pdstsinit -s f001 -f C:\sysarea02\f1sts01b -l 4096 -c 256
pdstsinit -s f001 -f C:\sysarea03\f1sts03a -l 4096 -c 256
pdstsinit -s f001 -f C:\sysarea03\f1sts02b -l 4096 -c 256
```

Example of command entry (BES1)

```
pdstsinit -s b001 -f C:\sysarea01\b1sts01a -l 4096 -c 256
pdstsinit -s b001 -f C:\sysarea01\b1sts03b -l 4096 -c 256
pdstsinit -s b001 -f C:\sysarea02\b1sts02a -l 4096 -c 256
pdstsinit -s b001 -f C:\sysarea02\b1sts01b -l 4096 -c 256
pdstsinit -s b001 -f C:\sysarea03\b1sts03a -l 4096 -c 256
pdstsinit -s b001 -f C:\sysarea03\b1sts02b -l 4096 -c 256
```

Example of command entry (BES2)

```
pdstsinit -s b002 -f C:\sysarea01\b2sts01a -l 4096 -c 256
pdstsinit -s b002 -f C:\sysarea01\b2sts03b -l 4096 -c 256
pdstsinit -s b002 -f C:\sysarea02\b2sts02a -l 4096 -c 256
pdstsinit -s b002 -f C:\sysarea02\b2sts01b -l 4096 -c 256
pdstsinit -s b002 -f C:\sysarea03\b2sts03a -l 4096 -c 256
pdstsinit -s b002 -f C:\sysarea03\b2sts02b -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 C:\sysarea01\ulsts01a -l 4096 -c 256
pdstsinit -u unt1 -f C:\sysarea01\ulsts03b -l 4096 -c 256
pdstsinit -u unt1 -f C:\sysarea02\ulsts02a -l 4096 -c 256
pdstsinit -u unt1 -f C:\sysarea02\ulsts01b -l 4096 -c 256
pdstsinit -u unt1 -f C:\sysarea03\ulsts03a -l 4096 -c 256
pdstsinit -u unt1 -f C:\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 RDAREAs
- Data directory RDAREAs
- Data dictionary RDAREAs

4.5.1 Basics

To create system RDAREAs, use the following general procedures:

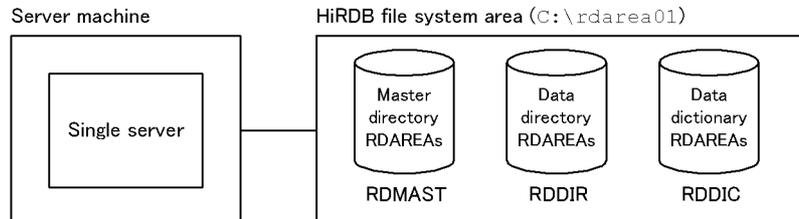
1. Create a system RDAREA in the HiRDB file system area for RDAREAs that you created in *4.3 Creating HiRDB file system areas*.
2. For a HiRDB/Parallel Server, create a system RDAREA in a HiRDB file system area for the server machine on which the dictionary server is defined.
3. For details about how to design the system RDAREA, see *8.4 Placing RDAREAs* for a HiRDB/Single Server, and *9.4 Placing RDAREAs* for a HiRDB/Parallel Server.
4. For a HiRDB/Parallel Server, 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)

This example creates system RDAREAs in the following HiRDB file system area for RDAREAs:

- C:\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:

- C:\hirdb\pdinit01

Contents of the control statement file

```
create rdarea RDMAST for masterdirectory           1
  file name "C:\rdarea01\rdmast01"
  initial 10 segments;
create rdarea RDDIR for datadirectory             2
  file name "C:\rdarea01\rddir01"
  initial 5 segments;
create rdarea RDDIC for datadictionary           3
  extension use 50 segments
  file name "C:\rdarea01\rddic01"
  initial 20 segments;
```

Explanation:

1. Definition of the master directory RDAREAs

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 RDAREAs

4. Setting Up an Environment Using Commands

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 RDAREAs

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 C:\hirdb\pdinit01
```

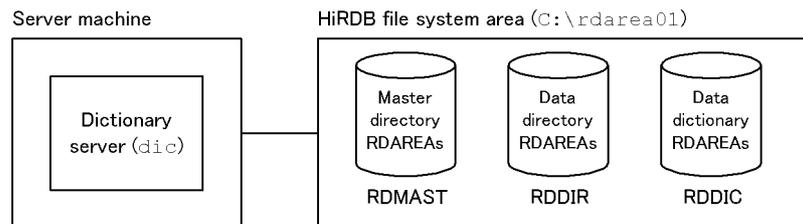
Explanation:

-d: Specifies the name of the control statement file that was previously created in (1).

4.5.3 Example 2 (HiRDB/Parallel Server)

This example creates system RDAREAs in the following HiRDB file system area for RDAREAs on the server machine where the dictionary server is defined:

- C:\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:

- C:\hirdb\pdinit01

Contents of the control statement file

```

create rdarea RDMAST for masterdirectory           1
  server name dic
  file name "C:\rdarea01\rdmast01"
  initial 10 segments;
create rdarea RDDIR for datadirectory             2
  server name dic
  file name "C:\rdarea01\rddir01"
  initial 5 segments;
create rdarea RDDIC for datadictionary           3
  server name dic
  extension use 50 segments
  file name "C:\rdarea01\rddic01"
  initial 20 segments;

```

Explanation:

1. Definition of the master directory RDAREAs

This example specifies the name of the dictionary server (`dic`) that manages the master directory RDAREAs. It creates a HiRDB file named `rdmast01` in the HiRDB file system area. This HiRDB file has 10 segments.

2. Definition of data directory RDAREAs

This example specifies the name of the dictionary server (`dic`) that manages the data directory RDAREAs. 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 RDAREAs

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 C:\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, execute the `pdstart` command from the server machine where the single server is defined.
- To start a HiRDB/Parallel Server, 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

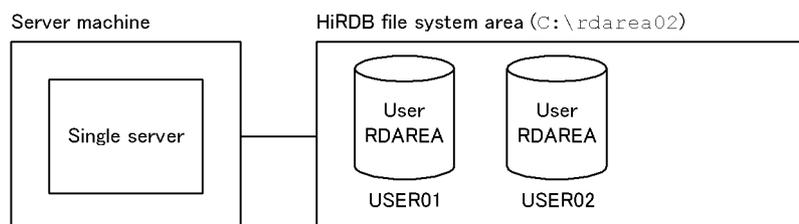
To create user RDAREAs, use the following general procedures:

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, 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, and *9.4 Placing RDAREAs* for a HiRDB/Parallel Server.
4. For a HiRDB/Parallel Server, 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, 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, enter the `pdstart` command from the server machine where the system manager is defined.

4.7.2 Example 1 (HiRDB/Single Server)

This example creates user RDAREAs in the following HiRDB file system area for RDAREAs:

- `C:\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:

- `C:\hirdb\pdmod01`

Contents of the control statement file

```
create rdarea USER01 for user used by PUBLIC           1
  extension use 50 segments
  file name "C:\rdarea02\user01"
  initial 500 segments;
create rdarea USER02 for user used by PUBLIC           2
  extension use 50 segments
  file name "C:\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 C:\hirdb\pdmod01
```

Explanation:

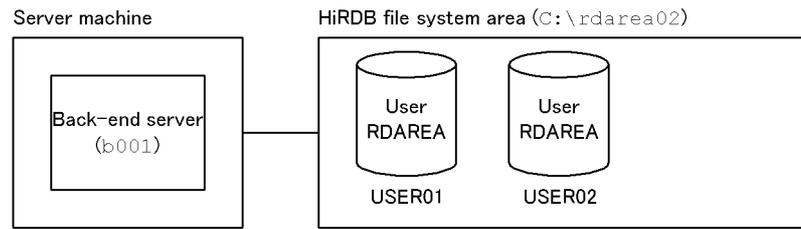
-a: Specifies the name of the control statement file that was previously created in (1).

4.7.3 Example 2 (HiRDB/Parallel Server)

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:

- C:\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:

- C:\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 "C:\rdarea02\user01"
    initial 500 segments;
create rdarea USER02 for user used by PUBLIC           2
  server name b001
  extension use 50 segments
  file name "C:\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

4. Setting Up an Environment Using Commands

extension size is 50 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a C:\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, 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, and *9.4 Placing RDAREAs* for a HiRDB/Parallel Server.
4. For a HiRDB/Parallel Server, 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, 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, enter the `pdstart` command from the server machine where the system manager is defined.

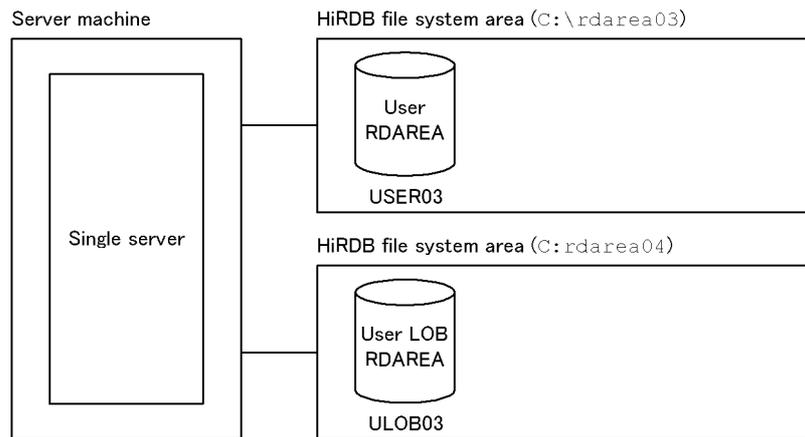
4.8.2 Example 1 (HiRDB/Single Server)

This example creates a user RDAREA to store a LOB column structure base table in the following HiRDB file system area for RDAREA:

- C:\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:

- C:\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:

- C:\hirdb\pdmod02

Contents of the control statement file

```
create rdarea USER03 for user used by PUBLIC           1
  extension use 50 segments
  file name "C:\rdarea03\user03"
  initial 500 segments;
create rdarea ULOB03 for LOB used by PUBLIC           2
  extension use 50 segments
  file name "C:\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 20000

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 C:\hirdb\pdmod02
```

Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

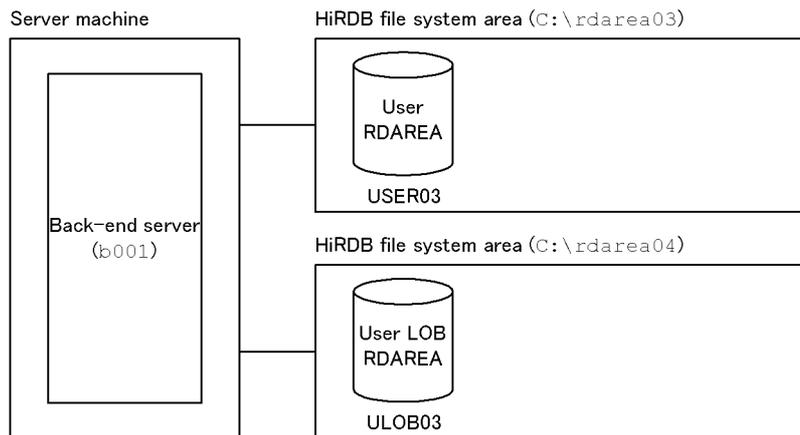
4.8.3 Example 2 (HiRDB/Parallel Server)

This example creates a user RDAREA to store a LOB column structure base table in the following HiRDB file system area for RDAREA:

- C:\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:

- C:\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:

4. Setting Up an Environment Using Commands

- C:\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 "C:\rdarea03\user03"
    initial 500 segments;
create rdarea ULOB03 for LOB used by PUBLIC          2
  server name b001
  extension use 50 segments
  file name "C:\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 20000 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 C:\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

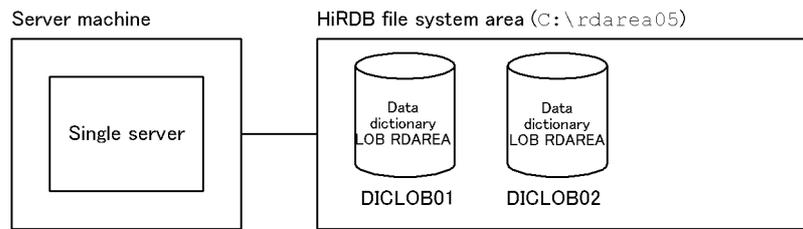
To create data dictionary LOB RDAREAs, use the following general procedures:

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, 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, and *9.4 Placing RDAREAs* for a HiRDB/Parallel Server.
4. For a HiRDB/Parallel Server, 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, 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, enter the `pdstart` command from the server machine where the system manager is defined.

4.9.2 Example 1 (HiRDB/Single Server)

This example creates data dictionary LOB RDAREAs in the following HiRDB file system area for RDAREA:

- `C:\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:

- C:\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 "C:\rdarea05\diclob01"
  initial 10000 segments;
create rdarea DICLOB02 for LOB used by HiRDB(SQL_ROUTINES)    2
  extension use 1000 segments
  file name "C:\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 10000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 1000 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 10000 segments.

The example uses the RDAREA automatic extension feature. The extension

size is 1000 segments.

(2) Executing the database structure modification utility (pdmod)

Example of command entry

```
pdmod -a C:\hirdb\pdmod03
```

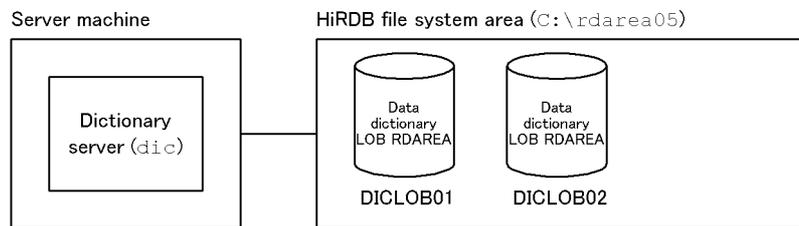
Explanation:

-a: Specifies the name of the control statement file that was previously created in (1).

4.9.3 Example 2 (HiRDB/Parallel Server)

This example creates data dictionary LOB RDAREAs in the following HiRDB file system area for RDAREA:

- C:\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:

- C:\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 "C:\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 "C:\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 10000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 1000 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 10000 segments.

The example uses the RDAREA automatic extension feature. The extension size is 1000 segments.

(2) Executing the database structure modification utility (`pdmod`)

Example of command entry

```
pdmod -a C:\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

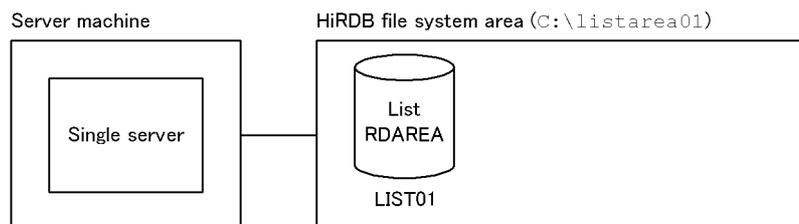
To create RDAREAs for lists, use the following general procedures:

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, 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, and *9.4 Placing RDAREAs* for a HiRDB/Parallel Server.
4. For a HiRDB/Parallel Server, 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, 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, enter the `pdstart` command from the server machine where the system manager is defined.

4.10.2 Example 1 (HiRDB/Single Server)

This example creates an list RDAREA in the following HiRDB file system area for list RDAREAs:

- `C:\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:

- `C:\hirdb\pdmod04`

Contents of the control statement file

```
create rdarea LIST01 for list                               /
  page 4096 characters storage control segment 2 pages
  file name "C:\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 1000 segments.

(2) **Executing the database structure modification utility (`pdmod`)**

Example of command entry

```
pdmod -a C:\hirdb\pdmod04
```

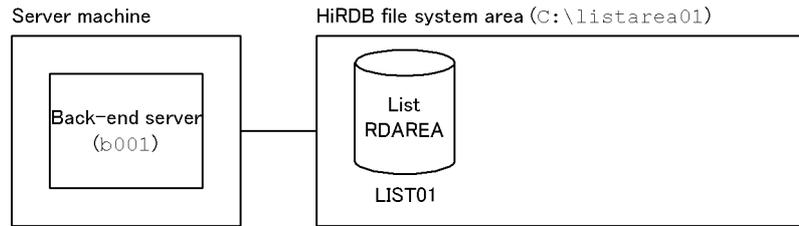
Explanation:

- a: Specifies the name of the control statement file that was previously created in (1).

4.10.3 Example 2 (HiRDB/Parallel Server)

This example creates a list RDAREA in the following HiRDB file system area for list RDAREAs:

- `C:\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:

- `C:\hirdb\pdmod04`

Contents of the control statement file

```
create rdarea LIST01 for list           /
server name b001
page 4096 characters storage control segment 2 pages
file name "C:\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 1000 segments.

(2) Executing the database structure modification utility (`pdmod`)

Example of command entry

```
pdmod -a C:\hirdb\pdmod04
```

Explanation:

- a: Specifies the name of the control statement file that was previously created in (1).

Chapter

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.

This chapter contains the following sections:

- 5.1 Overview of plug-in environment setup
- 5.2 Upgrading plug-ins
- 5.3 Deleting 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. Start HiRDB.
5. Add data dictionary LOB RDAREAs, user RDAREAs, and user LOB RDAREAs.¹
6. Register plug-ins.
7. Initialize the registry facility.^{1, 2}
8. Terminate HiRDB.
9. Add the `pdplugin` operand.
10. Start HiRDB.
11. 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) Starting HiRDB

The HiRDB administrator uses the `pdstart` command to start HiRDB.

(5) 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 RDAREA¹
- User LOB RDAREA¹
- Data dictionary LOB RDAREA² (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.

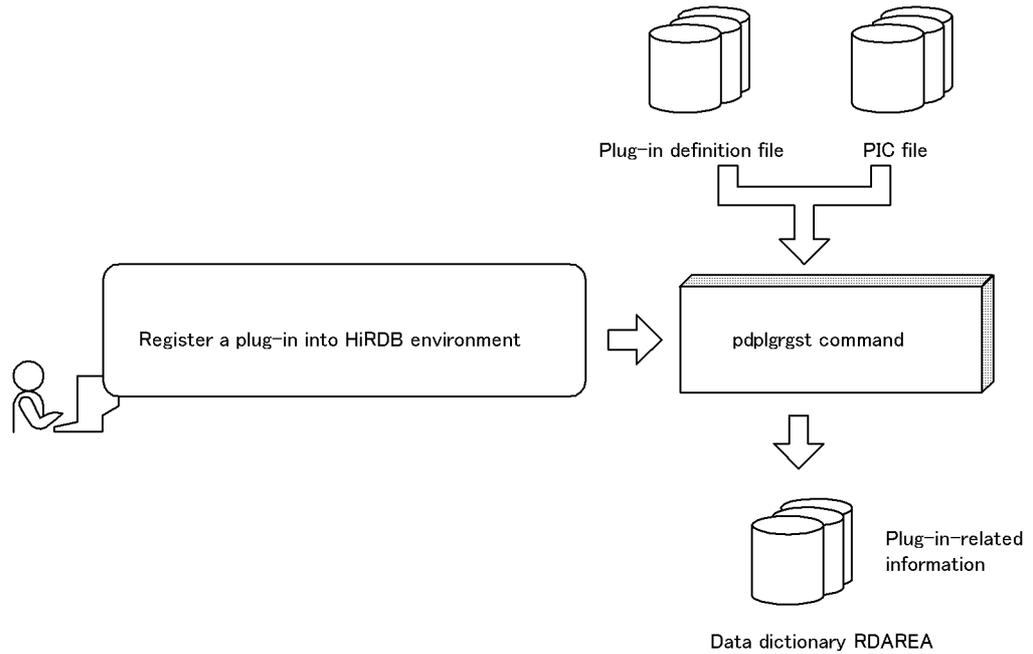
² You need to add this RDAREA to make the stored-procedure facility available to HiRDB before the registry facility initialization utility (`pdreginit`) is executed.

(6) Registering plug-ins

Use the `pdplgrgst` command to register your plug-ins in HiRDB. You can enter the `pdplgrgst` command from any server machine.

Figure 5-1 shows the procedure for registering plug-ins.

Figure 5-1: 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: C:\TSPlugin_phsgml\etc)
- Index type plug-in
`pdplgrgst _phngram.idx _phngram.pic`
 (Current directory: C:\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.

(7) 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.

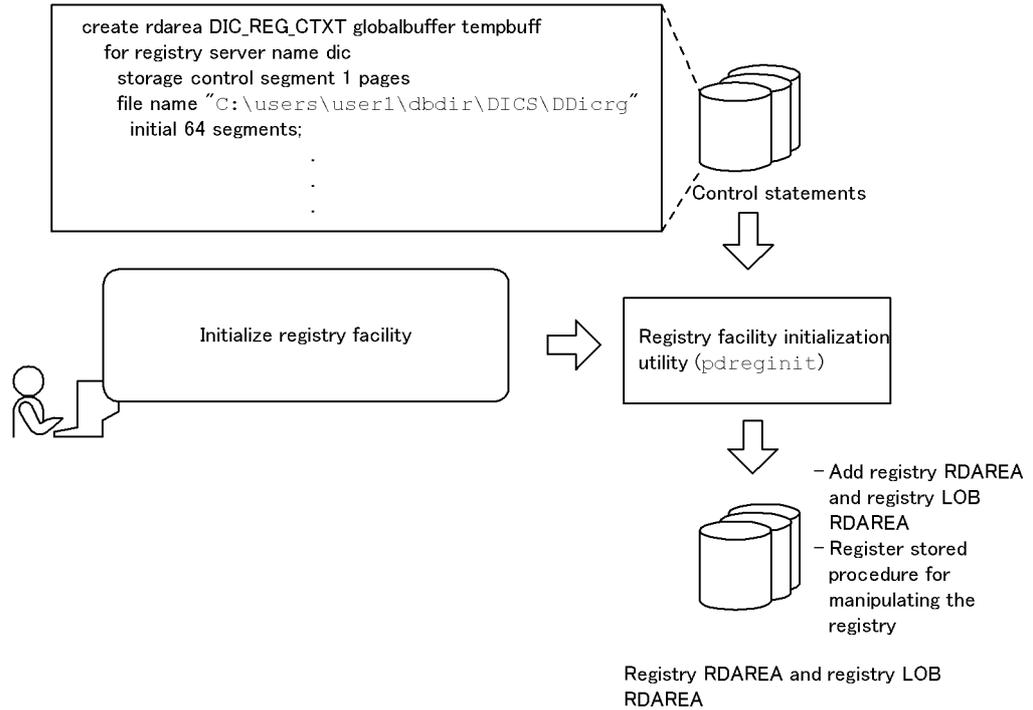
- Registry RDAREA
- Registry LOB RDAREA

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.

Figure 5-2 shows the procedure for creating a registry RDAREA and registry LOB RDAREA.

Figure 5-2: Procedure for creating a registry RDAREA and registry LOB RDAREA



(8) 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.

(9) 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, 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.

(10) Starting HiRDB

Use the `pdstart` command to start HiRDB.

(11) 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**

Table 5-1 shows the HiRDB (unit) setup/startup conditions and availability of plug-ins.

Table 5-1: HiRDB (unit) startup conditions and availability of plug-ins

Plug-in utilization declaration (pdplugin operand)	Registration of plug-in (pdplgrgst)	Plug-in initialization error	Whether or not 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:

5. Setting Up the Plug-in Environment

- 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 using the data type provided by the plug-in.
- Indexes using the index type provided by the plug-in.
- Functions provided by the plug-in.

5.2.1 Notes about upgrading

When upgrading your plug-ins, note the following:

- The versions of the data type plug-in and the index type plug-in must match.
- When upgrading both data type and index type plug-ins, be sure to upgrade the data type plug-in first.
- Once you register a new version of a plug-in, you can no longer restore the old version.
- The `SPECIFIC_NAME` column is updated in the `SQL_PLUGIN_ROUTINES` and `SQL_PLUGIN_ROUTINE_PARAMS` data dictionary tables.
- If you change the owner of a plug-in during upgrading, the functions may become invalid. In such cases, re-create the function's SQL object using `ALTER ROUTINE`.

5.2.2 Upgrading procedure

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 RDAREAs
- Data dictionary RDAREAs
- Data directory RDAREAs
- Data dictionary LOB RDAREAs

(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*plug-in-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) 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.

(5) 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.

(6) Restoring files that were saved

Restore the files that were saved in step (3) at all server machines where HiRDB is set up.

(7) Restarting HiRDB

Enter the `pdstart` command to start HiRDB.

(8) Registering the new version of the plug-in

Execute the `pdplgrgst` command, specifying the `-a` option to re-register the plug-in.

(9) Terminating HiRDB

Enter the `pdstop` command to terminate HiRDB normally.

(10) 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, add this operand to the system common definition at all server machines; otherwise, you will not be able to start HiRDB.

(11) 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. Table 5-2 lists the SQL statements that can be used to delete these resources.

- Abstract data type, procedure, function, and table using the abstract data type that is provided by the plug-in that is to be deleted (applicable when the abstract data type provided by the plug-in is specified as an attribute among all the user-defined abstract data types)
- 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
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.
 - 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) 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.

Chapter

6. Creating Databases

This chapter describes the procedures from schema, table, and index creation through data storage.

This chapter contains the following sections:

- 6.1 Overview of database creation
- 6.2 Creating a row partitioned table
- 6.3 Creating a table with a LOB column
- 6.4 Creating a table containing a plug-in-provided abstract data type
- 6.5 Creating a table containing a user-defined abstract data type
- 6.6 Handling errors during batch index creation
- 6.7 Handling utility abnormal termination errors during data loading with the synchronization point specification

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)
- 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 manual *HiRDB Version 8 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 manual *HiRDB Version 8 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;
```

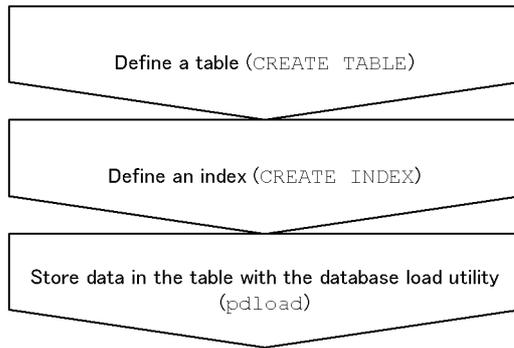
(5) Specifying the data conversion facility

When storing data in a database, you can use a facility to convert the data. Evaluate whether or not to use the following facilities. For details about these facilities, see the manual *HiRDB Version 8 System Operation Guide*.

- Space conversion facility
- Facility for conversion to a DECIMAL signed normalized number

6.1.2 Database creation procedure

Figure 6-1 shows the database creation procedure.

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 manual *HiRDB Version 8 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 Table 6-1.

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:

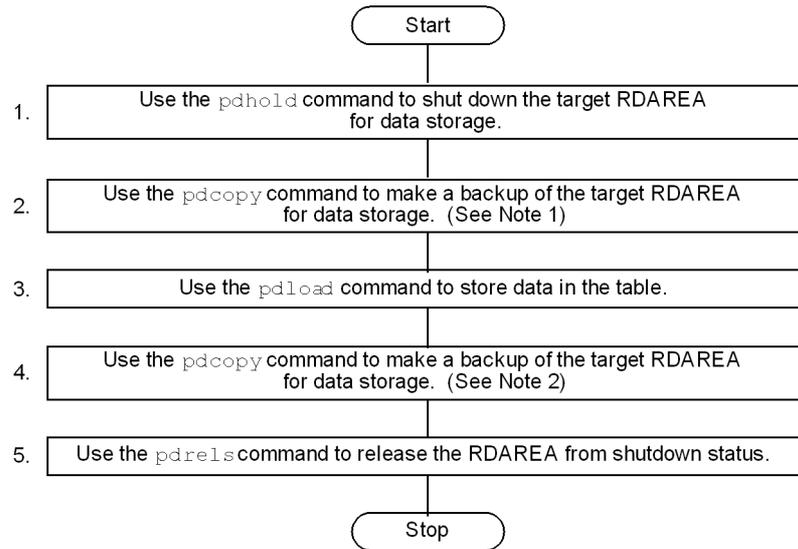
Condition	Mode to be selected
For initial storage, only the target table (or index) for data storage is stored in the RDAREA.	No-log mode
There is large volume of input data, and data storage will take a long time.	
There is only a small amount of input data.	Log acquisition mode

(4) Operational differences

Depending on the mode that is selected, there are differences in the operating

procedure during data storage, as shown in Figure 6-2.

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		
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 (Creating divided-input data files)

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 8 Command Reference*.

6.1.7 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.

6.1.8 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 RDAREa and save disk space.

For details about how to retrieve data dictionary tables and for details about the `SQL_RDAREAs` table, see the manual *HiRDB Version 8 UAP Development Guide*. For details about how to delete RDAREAs, see the manual *HiRDB Version 8 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 `RDAREA RDAREA05`. For a `HiRDB/Parallel Server`, 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*.

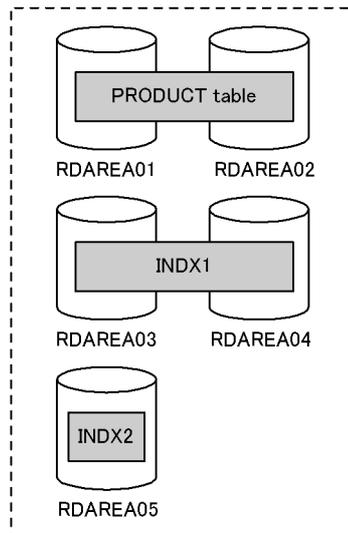
Definition example of partitioning key index (INDX1)

PNO	PNAME	LPRICE	QTY
01010	BLOUSE	3500	96
01011	BLOUSE	3500	35
02021	SHIRT	3640	63
⋮	⋮	⋮	⋮

Definition example of non-partitioning key index (INDX2)

•HiRDB/Single Server

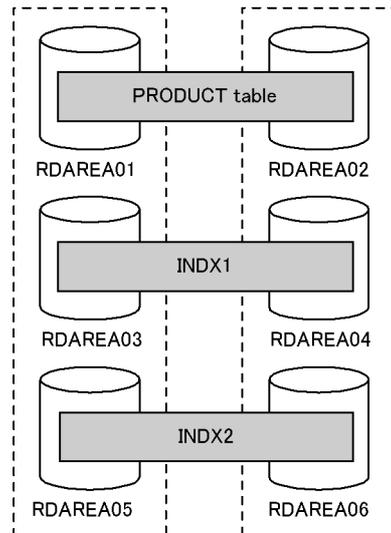
SDS



•HiRDB/Parallel Server

BES1

BES2



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),
 PNAME NCHAR(15),
```

6. Creating Databases

```
LPRICE INTEGER,  
QTY INTEGER  
) IN ((RDAREA01) PNO<='10000', (RDAREA02));
```

Specification of boundary value:

```
CREATE TABLE PRODUCT  
(PNO CHAR(5),  
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),  
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

```
CREATE INDEX INDX1 ON TABLE (PNO)  
IN ((RDAREA03), (RDAREA04));  
CREATE INDEX INDX2 ON PRODUCT (QTY)  
IN (RDAREA05);
```

(b) HiRDB/Parallel Server

```
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, 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 8 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 manual *HiRDB Version 8 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 8 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

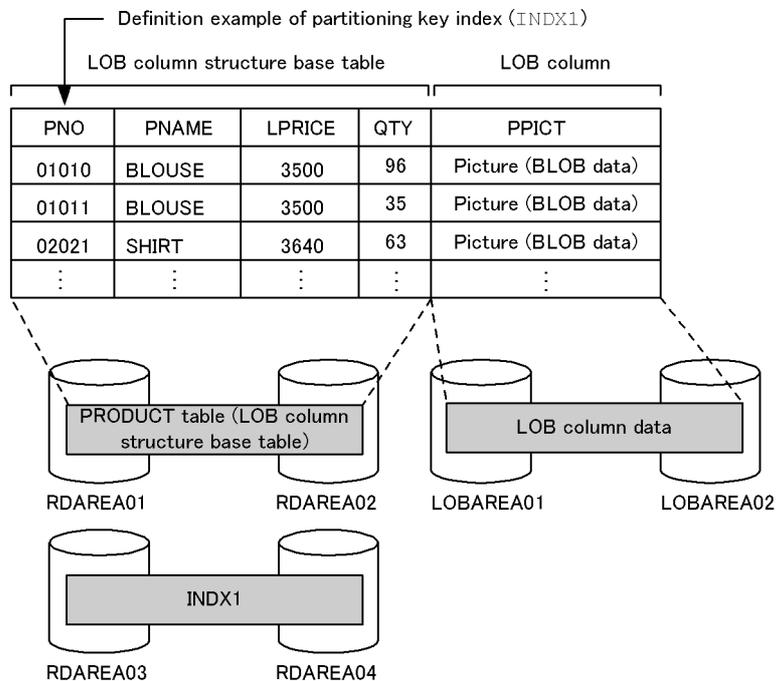
If you have executed data loading, you should execute the database condition analysis utility (`pddbst`) next to check the data storage status. This utility enables you to check whether the database has been created exactly as designed. The database condition analysis utility can obtain the following 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.



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 8 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 manual *HiRDB Version 8 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 8 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
- 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 8 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 was 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 manual *HiRDB Version 8 System Operation Guide*.
5. Use the `pdrels` command to release the target data storage RDAREAs from shutdown status.

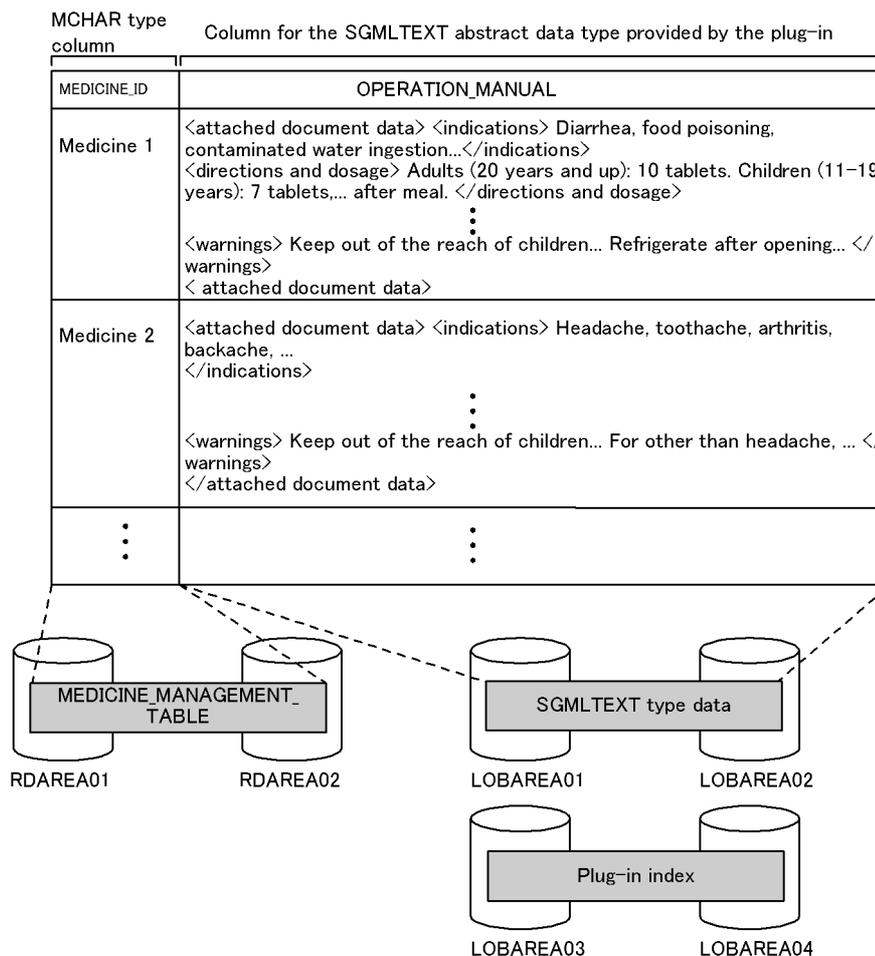
For details about these commands and utilities, and about how to verify the command and utility execution results, see the manual *HiRDB Version 8 Command Reference*.

6.4 Creating a table containing a plug-in-provided abstract data 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                                1.
  ALLOCATE(SGMLTEXT IN((LOBAREA01), (LOBAREA02)))        2.
  PLUGIN'<DTD>medicine.dtd</DTD>'                        3.
) IN((RDAREA01)MEDICINE_ID<='MEDICINE 10', (RDAREA02)); 4.

```

Specification of boundary value:

```

CREATE TABLE MEDICINE_MANAGEMENT_TABLE (
MEDICINE_ID MCHAR(15),
OPERATION_MANUAL SGMLTEXT                                1.
  ALLOCATE(SGMLTEXT IN((LOBAREA01), (LOBAREA02)))        2.
  PLUGIN'<DTD>medicine.dtd</DTD>'                        3.
) PARTITIONED BY MEDICINE_ID
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                                1.
  ALLOCATE(SGMLTEXT IN((LOBAREA01), (LOBAREA02)))        2.
  PLUGIN'<DTD>medicine.dtd</DTD>'                        3.
) [FIX] * HASH HASH6 BY MEDICINE_ID
IN(RDAREA01, 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.

* This specification is applicable to FIX hash partitioning.

(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 8 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 manual *HiRDB Version 8 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 8 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 manual *HiRDB Version 8 UAP Development Guide*.

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;                                7.
        SET d_EMPLOYEE..POSITION=p_POSITION;                    7.
        SET d_EMPLOYEE..EMPLOYMENT_DATE
          =p_EMPLOYMENT_DATE;                                    7.
        SET d_EMPLOYEE..ID_PHOTO =p_ID_PHOTO;                   7.
        SET d_EMPLOYEE..SALARY=p_SALARY;                         7.
        RETURN d_EMPLOYEE;                                       8.
    END,

    PUBLIC FUNCTION SERVICE_YEARS (p t_EMPLOYEE)
    RETURNS INTEGER                                             9.
    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                                              10.
    BEGIN
        DECLARE rate FLOAT;
        SET rate=SERVICE_YEARS (p)*0.2/30;
        RETURN rate;
    END,

    PUBLIC FUNCTION BONUS (p t_EMPLOYEE)
    RETURNS INTEGER                                           11.
    BEGIN
        DECLARE bonus INTEGER;
        SET bonus=p..SALARY*BONUS_FACTOR (p);
        RETURN bonus;
    END
)

```

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 *12.16 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 *12.16 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 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.

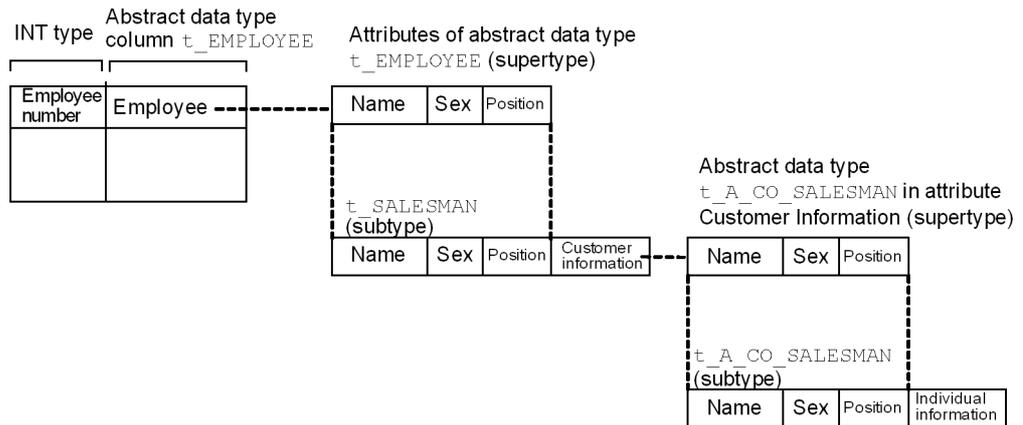
(4) Procedure for deleting the subtype of an abstract data type

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 Figure 6-3.

Figure 6-3: 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

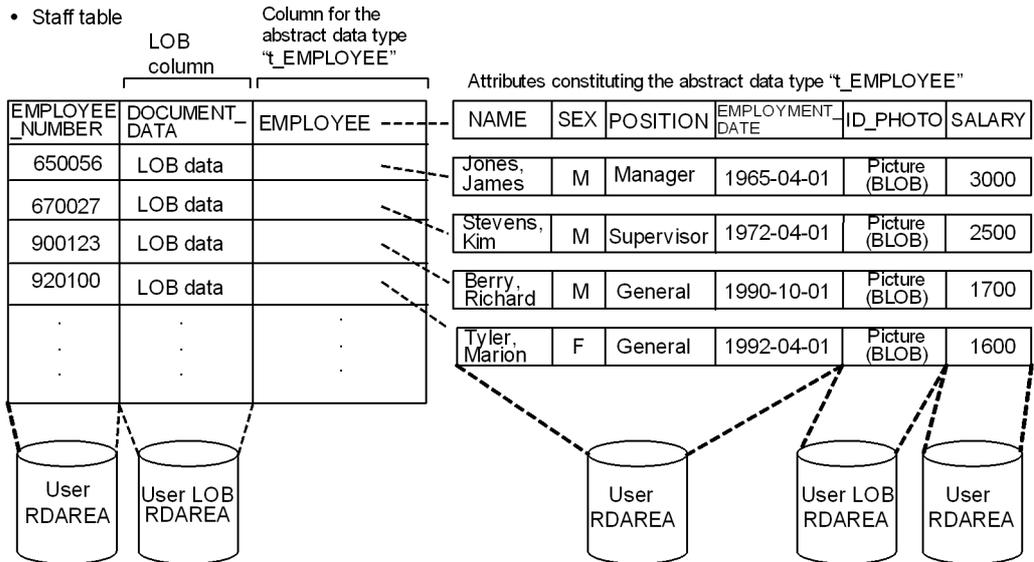
1. If a constructor function is used to generate values, the abstract data type as a whole is not null, even if the value of each attribute constituting the abstract data type is null.
2. If an abstract data type and its supertype are defined in a table, the abstract data type's subtypes cannot be deleted.
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));
```

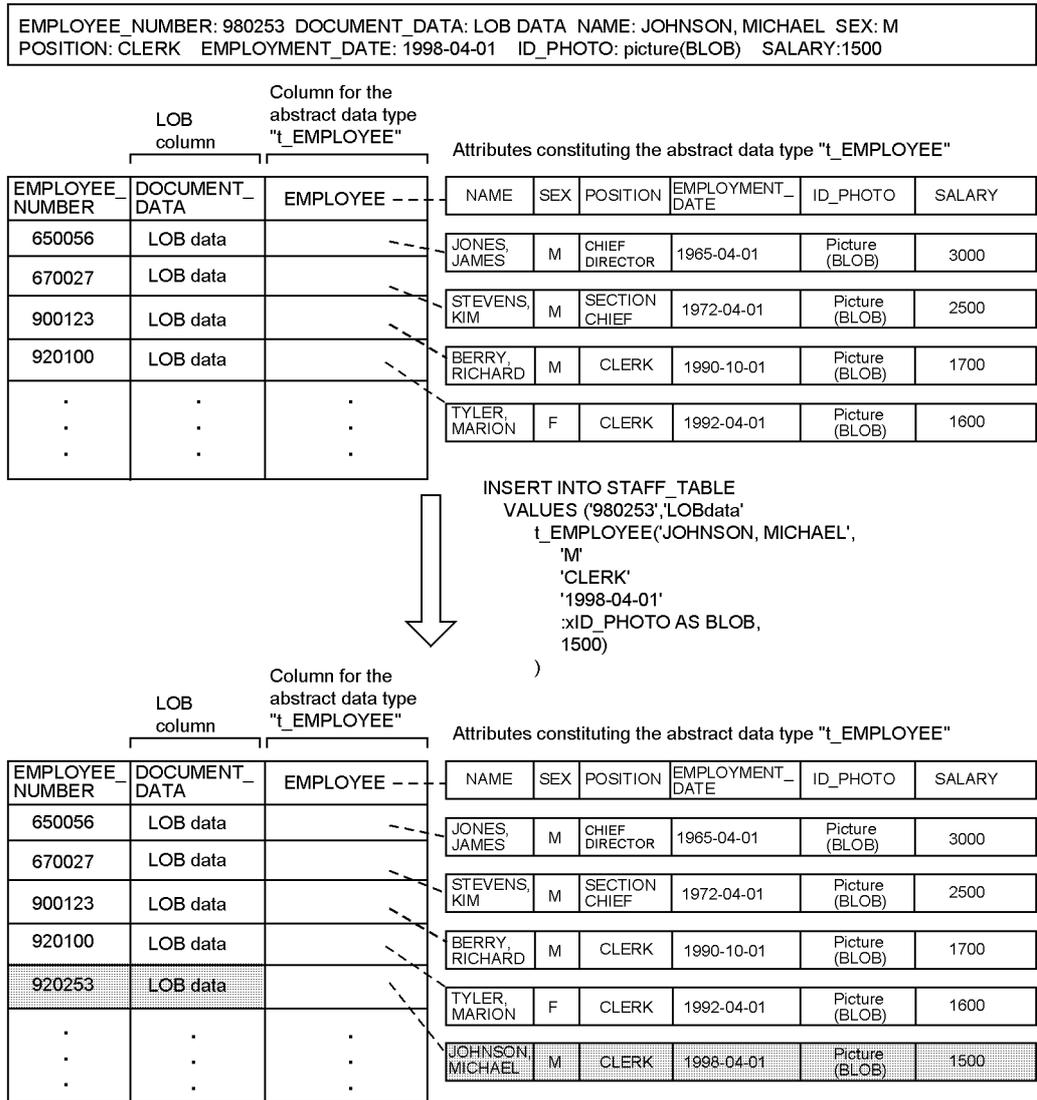
Explanation:

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. Figure 6-4 shows the procedure for inserting data in a table that contains an abstract data type column.

Figure 6-4: Procedure for inserting data in a table that contains an abstract data type column



Note: xPHOTO_ID 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 definition¹
- 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 Table 6-2.

Table 6-2: Database update log acquisition method for user LOB RDAREAs depending on a specification in the client environment definition

Client environment definition <code>PDDBLOG</code>	<code>RECOVERY</code> operand value in <code>CREATE TABLE</code>		
	<code>ALL</code>	<code>PARTIAL</code>	<code>NO</code>
<code>ALL</code>	<code>ALL</code>	<code>PARTIAL</code>	<code>NO</code>
<code>NO</code>	<code>NO</code>	<code>NO</code>	<code>NO</code>

`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

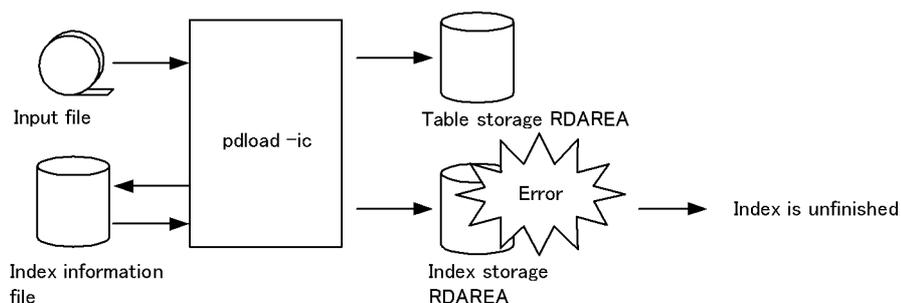
When data is inserted into a table containing an abstract data type column, the database condition analysis utility (`pddbst`) should be executed first to check the data storage status. This utility can check whether the database has been created exactly as designed.

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. Figure 6-5 shows the status of the index in the event of an error during batch index creation by the database load utility (`pdload`).

Figure 6-5: 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

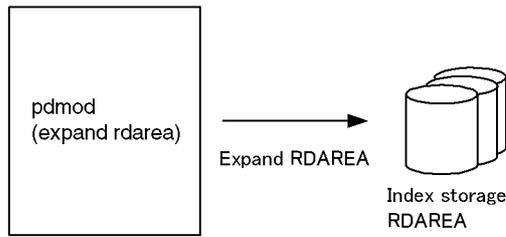
This section explains the procedure for handling errors that may occur during data loading in the log acquisition mode or pre-log acquisition mode.

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*.

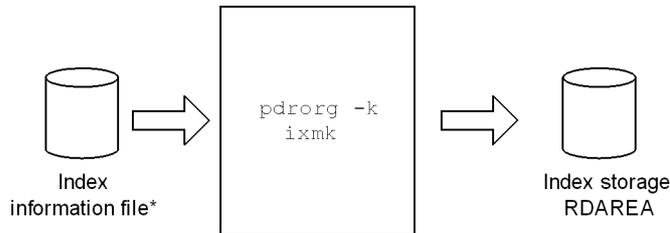
Restore the index storage RDAREAs according to the cause of error. The following shows the procedure:

(1) Error due to shortage of space in index storage RDAREA

1. Expand the index storage RDAREA.

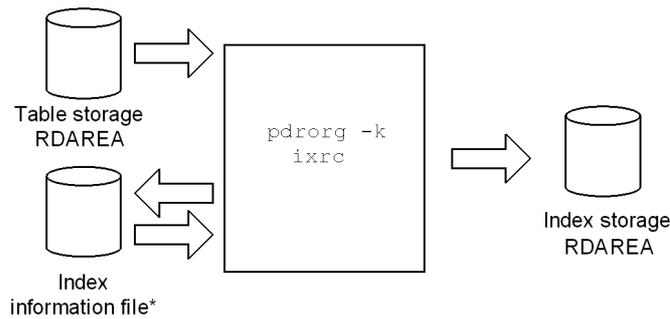


2. Create the index. Use the index information file created by the database load utility (`pdload`) to create the index in the batch mode.



* The key information for the addition mode storage and from the existing data is output to this file. In the case of a plug-in index, only the key information from the addition mode data 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 (`pdroorg`).

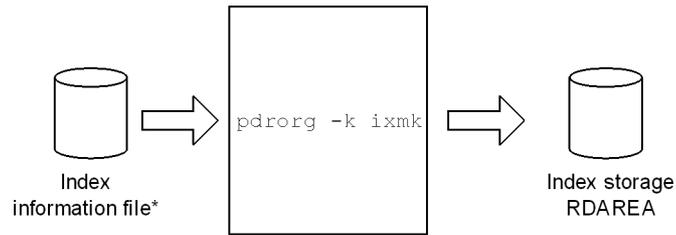


* This file was newly created by the database reorganization utility (`pdroorg`). The key information for the addition mode storage and from the existing data is output.

(2) Sorting error (message *KFPL15062-E* is output) or an error due to forced termination of the utility by the *pdcancel* command

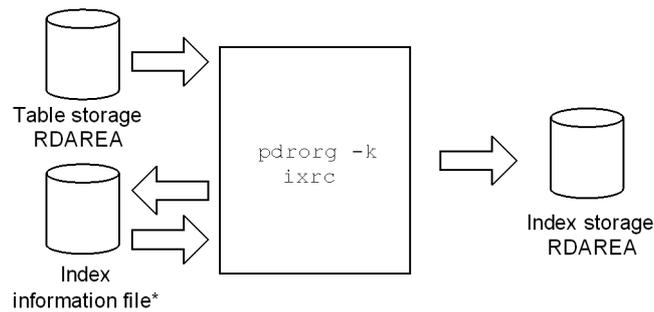
1. Create the index. Use the index information file created by the database load

utility (`pdload`) to create the index in the batch mode.



* The key information for the addition mode storage and from the existing data is output to this file. In the case of a plug-in index, only the key information from the addition mode data is output.

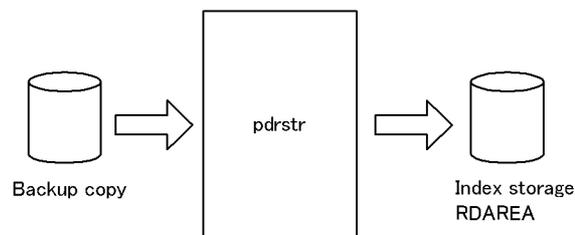
If the index information file created by the database load utility (`pdload`) is not available, re-create the index with the database reorganization utility (`pdrorg`).



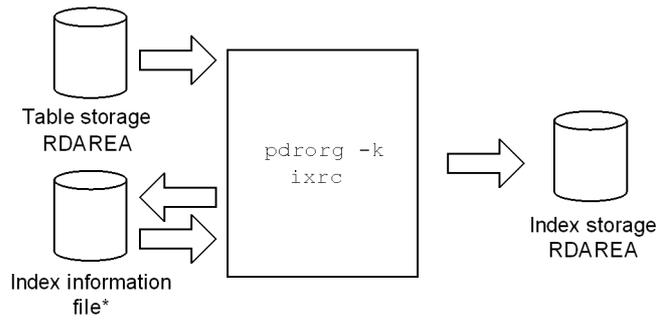
* This file is newly created by the database reorganization utility (`pdrorg`). The key information for the addition mode storage and from the existing data is output.

(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 newly created by the database reorganization utility (pdrorg). The key information for the addition mode storage and from the existing data is output.

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.

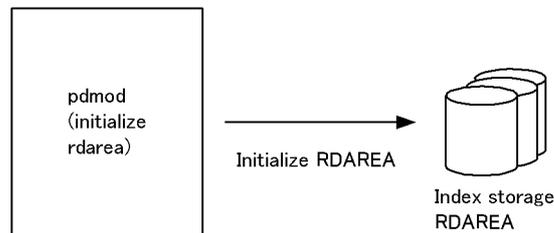
Restore the index storage RDAREAs according to the cause of error. The following shows the procedure:

(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:

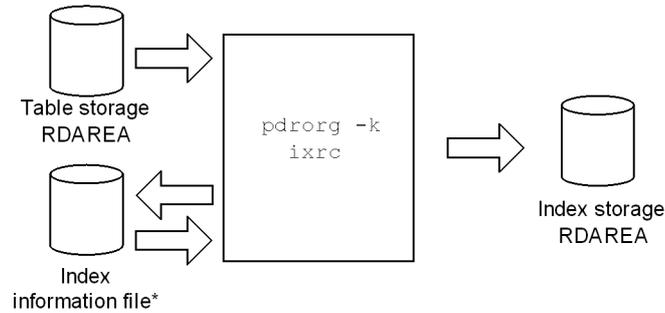
- The relevant index is stored in an RDAREA with different tables, indexes, or falsification prevented tables.



2. Expand the index storage RDAREA.



3. Re-create the index with the database reorganization utility (`pdrorg`).



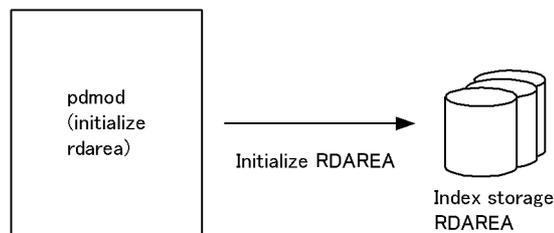
* This file is newly created by the database reorganization utility (`pdrorg`). The key information for the addition mode storage and from the existing data is output.

(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.

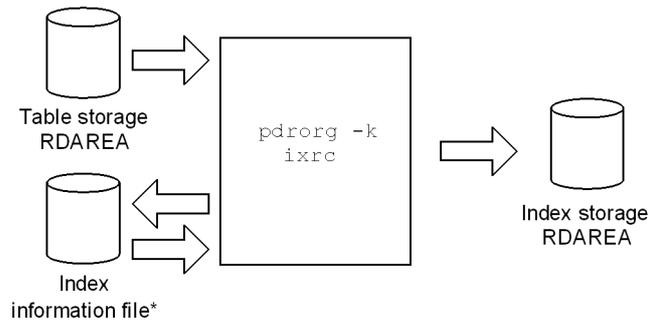
You can also recover from a backup. In such a case, until index storage has been completed, place the RDAREA in shutdown status in order to prevent accesses. In the following case, recovery must be performed using a backup:

- 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 (`pdrorg`).

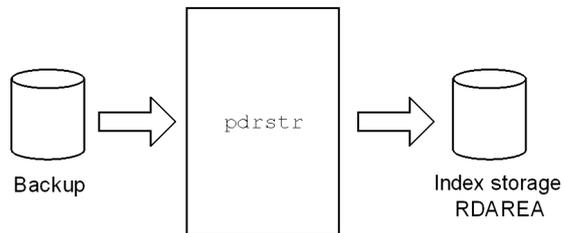
6. Creating Databases



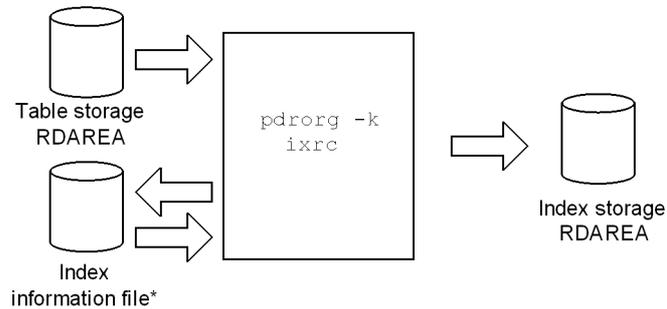
* This file is newly created by the database reorganization utility (`pdrorg`). The key information for the addition mode storage and from the existing data is output.

(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 newly created by the database reorganization utility (`pdrorg`). The key information for the addition mode storage and from the existing data is output.

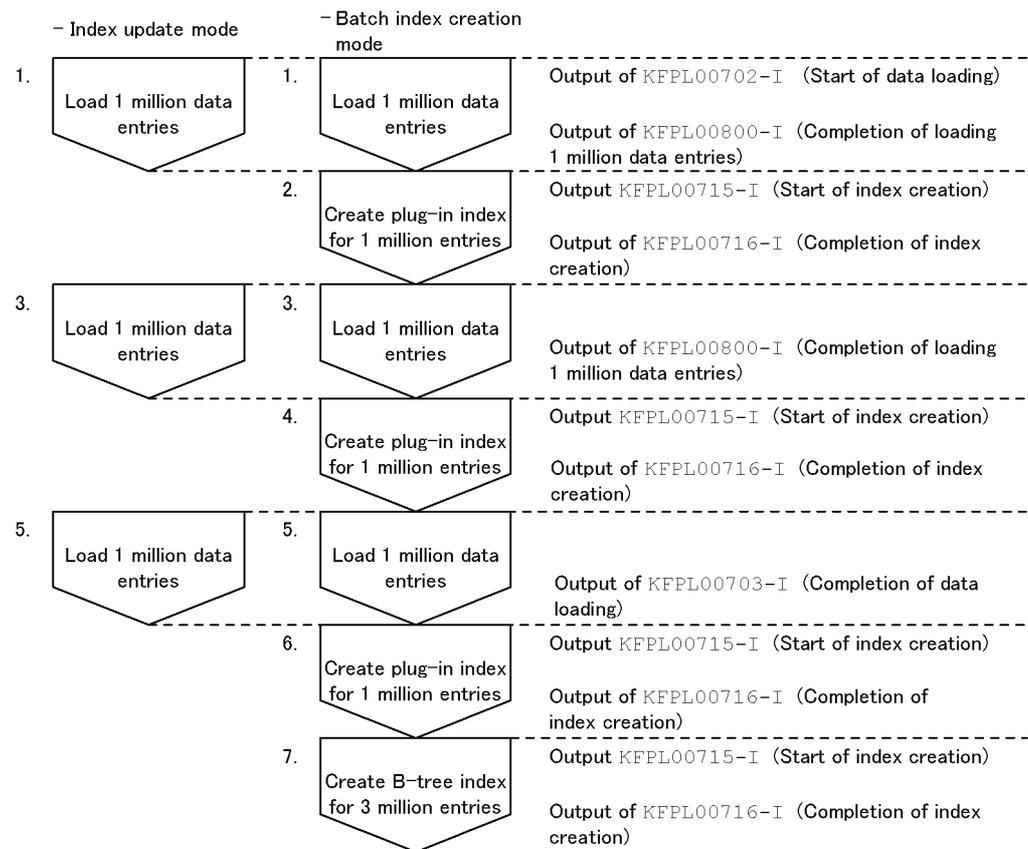
6.7 Handling utility abnormal termination errors during data loading with the synchronization point specification

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. Figure 6-6 shows the error handling procedure in the event of abnormal termination of a utility during data loading with the synchronization point specification.

Figure 6-6: 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 C:\pdrorg\rorg01
```

Explanation:

-k: Specifies `ixmk` to create the plug-in index in batch mode.

-t: Specifies the name of the table.

C:\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 C:\pdrorg\INDX01_2
```

(3) Reexecuting data loading

```
pdload TABLE1 C:\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 manual *HiRDB Version 8 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
```

6. Creating Databases

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 8 Command Reference*.

Chapter

7. Linking to Other Products

This chapter describes how to link HiRDB to other products.

This chapter contains the following sections:

- 7.1 Linking to the replication facility
- 7.2 Linking with an OLTP system
- 7.3 Linking to the HiRDB External Data Access facility
- 7.4 Linking to JP1

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 or not 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 manual *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide*.

Notes on system construction

- 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_sqlc` 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 manual *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. Notes

7.2.1 OLTP products supported for linking

HiRDB supports the following OLTP products for linking:

- OpenTP1
- TPBroker
- TUXEDO
- WebLogic Server

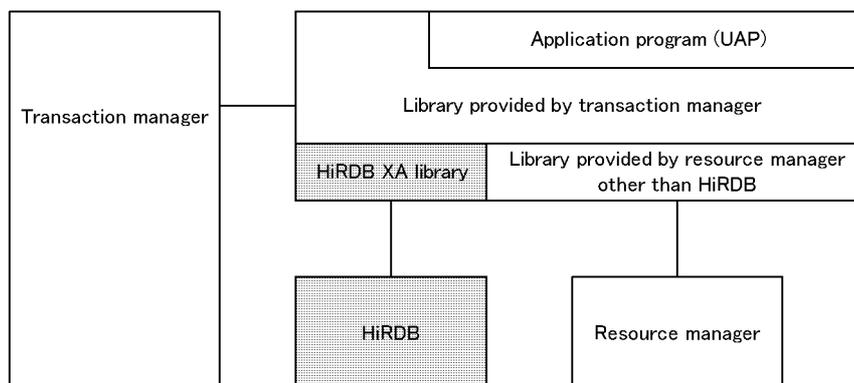
However, when the operating system is Windows (x64), HiRDB cannot link with OLTP products because no client library is provided for OLTP products that run in 64-bit mode.

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. Figure 7-1 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

Table 7-1 lists the functions supported by the HiRDB XA library.

Table 7-1: 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 or not to use the transaction transfer function depends on the <code>PDXAMODE</code> client environment definition operand. For details about the transaction transfer function, see 7.2.4 <i>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 7.2.10(3) <i>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.

Function	Description
Dynamic transaction registration	Enables HiRDB to register dynamically a transaction immediately before executing a UAP.
Multi-connection facility	Executes multiple CONNECTs for HiRDB servers separately from one process. For details about the multi-connection facility in the X/Open XA interface environment, see the manual <i>HiRDB Version 8 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

An XA interface supporting multi-thread enables you to link Object Transaction Service (OTS) with TPBroker 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.

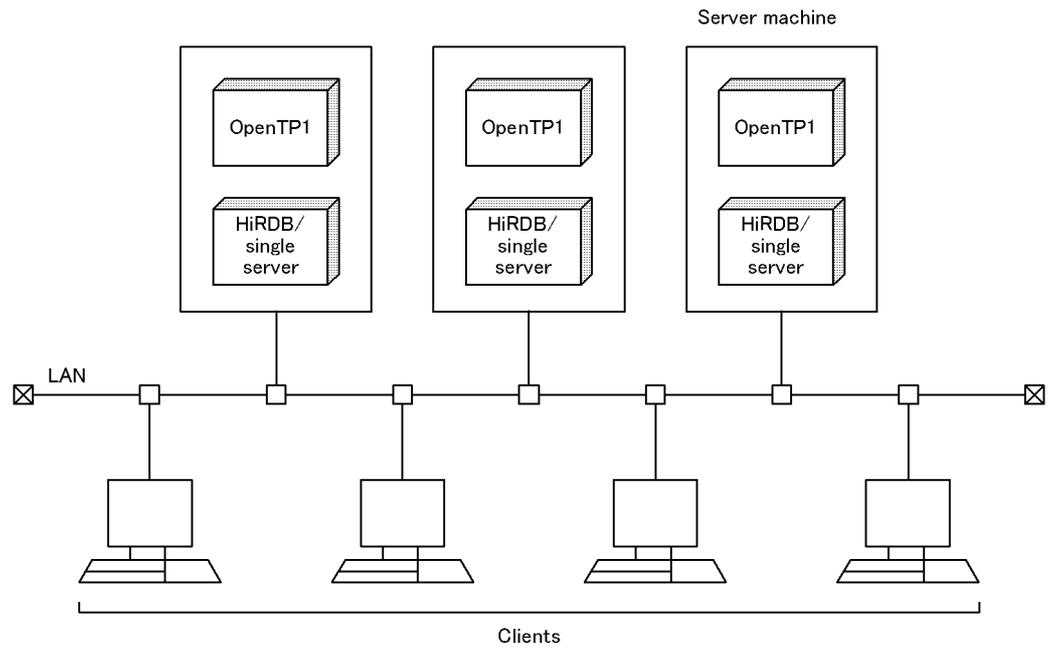
7.2.3 Example of HiRDB system configuration with OLTP linkage

This section describes a HiRDB system linked with OLTP by way of an example using OpenTP1.

(1) Linking with HiRDB/Single Servers

You can execute multiple HiRDB/Single Servers' update processing as a single transaction by linking OLTP (OpenTP1) to HiRDB/Single Servers. 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 Servers. This enables transaction processing to be performed at high speed. When integrating multiple HiRDB/Single Servers, consider linking your system to an OLTP. Figure 7-2 shows multiple HiRDB/Single Servers linked with an OLTP system (OpenTP1).

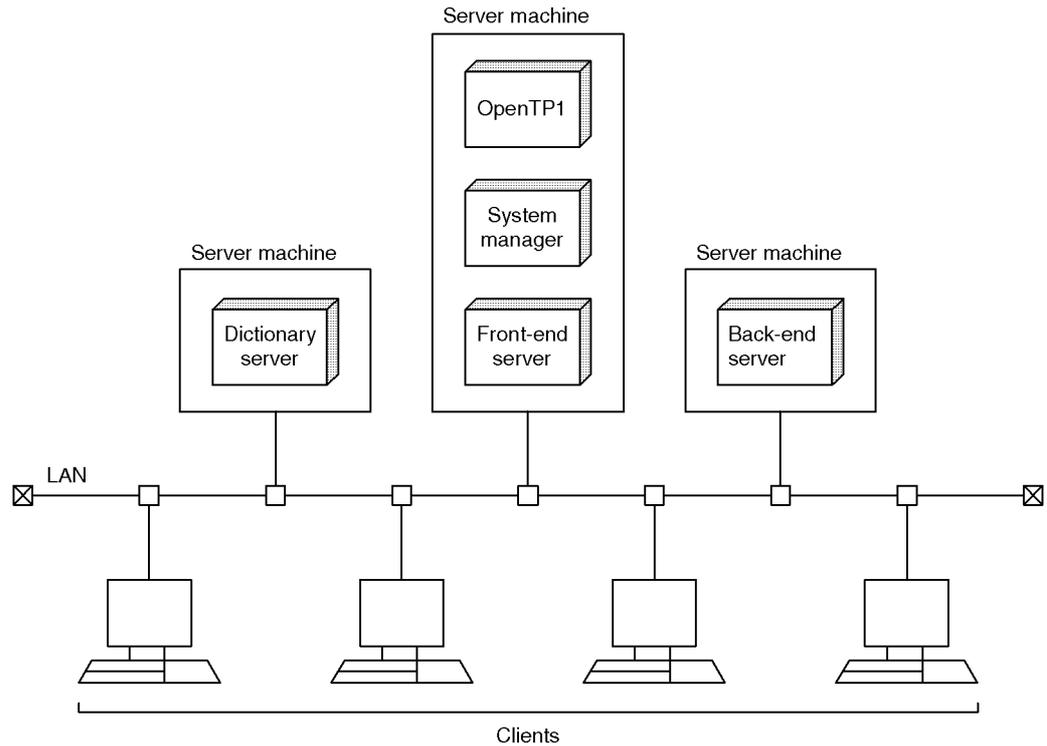
Figure 7-2: Linking HiRDB/Single Server with an OLTP system (OpenTP1)



(2) Linking with a HiRDB/Parallel Server

When a HiRDB/Parallel Server is linked with an OLTP system (OpenTP1), transaction processing can be achieved at high performance and high workload. Figure 7-3 shows a HiRDB/Parallel Server linked with an OLTP system (OpenTP1).

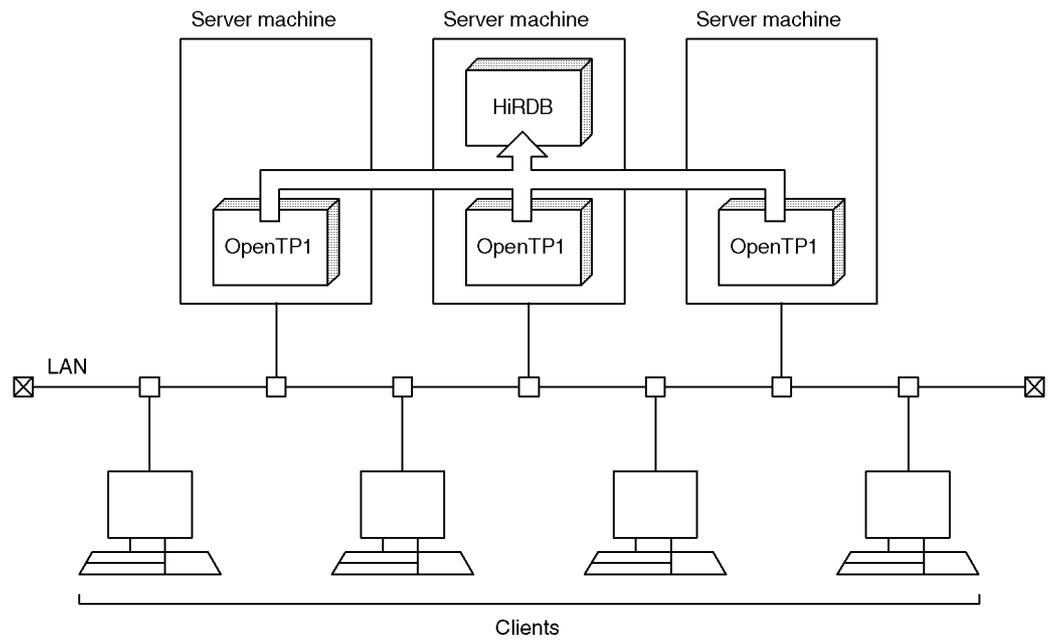
Figure 7-3: Linking HiRDB/Parallel Server with an OLTP system (OpenTP1)



(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). Figure 7-4 shows linking between multiple OLTPs (OpenTP1) and a single HiRDB.

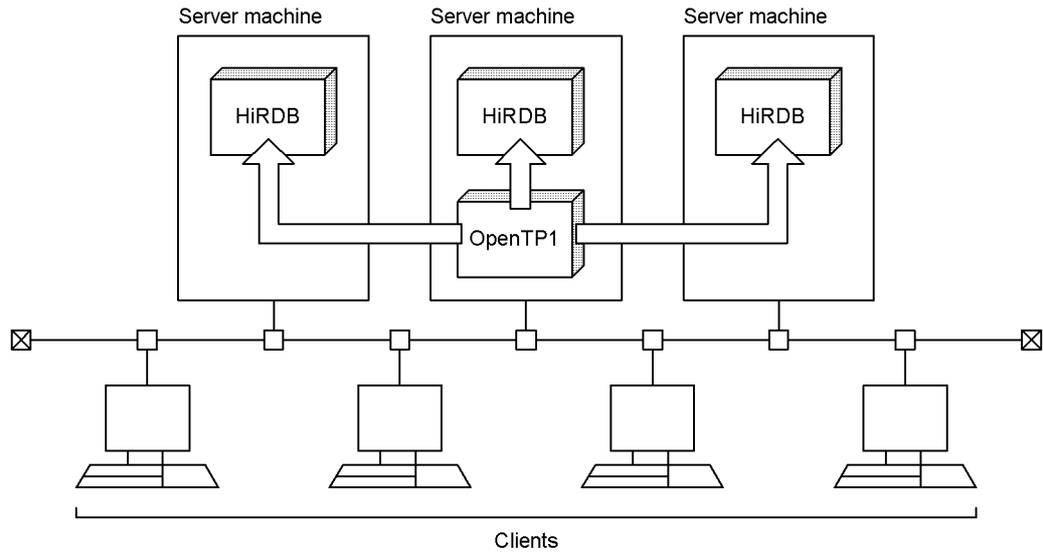
Figure 7-4: Linking between multiple OLTPs (OpenTP1) and a single HiRDB



(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. Figure 7-5 shows linking between one OLTP (OpenTP1) and multiple HiRDBs.

Figure 7-5: Linking between one OLTP (OpenTP1) and multiple HiRDBs

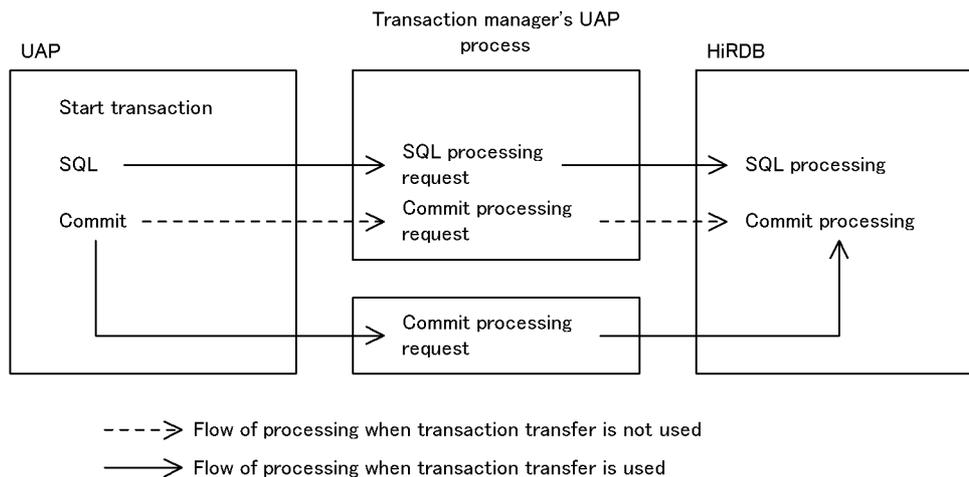


For details about the multi-connection facility, see the manual *HiRDB Version 8 UAP Development Guide*.

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*. Figure 7-6 provides 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 or not 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 manual *HiRDB Version 8 UAP Development Guide*.

Notes

1. If the information indicating whether or not 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 manual *HiRDB Version 8 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.

However, depending on conditions, as shown in Table 7-2, HiRDB can be set to not use the transaction transfer function even when OpenTP1 is set to use the transaction transfer function (to not suppress commit optimization or prepare optimization). Table 7-2 shows the relationship between the trnstring operand in the transaction service definition of the OpenTP1 system definition and the PDXAMODE operand for HiRDB.

Table 7-2: Relationship between OpenTP1's trnstring operand and HiRDB's PDXAMODE operand

Condition		PDXAMODE operand value	
trnstring operand specification	Status of OpenTP1 system		
-d option omitted	In the same OpenTP1 system, global transaction consists of only one AP.	0 or 1	
	In the same OpenTP1 system, global transaction consists of multiple APs	Only one AP is linked to the HiRDB XA library	0 or 1
		Multiple APs are linked to the HiRDB XA library	1
-d option specified	—	0	

Legend:

— : Not applicable

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).

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*.

The following criteria should be used to determine whether or not the transaction transfer function should be used in systems connected to OpenTP1 and HiRDB:

Criteria

The transaction transfer function should be used when the client UAPs and server UAPs satisfy the following conditions:

1. Client UAPs and server UAPs are located in the same OpenTP1 system environment.
2. Client UAPs and server UAPs are linked to the HiRDB XA library.
3. Client UAPs and server UAPs access HiRDB.*

* This means that one of the following conditions is satisfied:

1. SQL for accessing HiRDB is issued.
2. Either `process` or nothing is specified in the `trn_rm_open_close_scope` operand in the transaction service definition/user service definition of the OpenTP1 system definition.

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 `trnlnkrm` command to register HiRDB.
- TPBroker: Use the `tslnkrm` 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's JDBC connection pool.

(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 or not 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, etc.), 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

Table 7-3 lists the differences between dynamic and static registration.

Table 7-3: 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.#
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.# 	<ul style="list-style-type: none"> • Starts HiRDB transaction. • Processes SQL statements.

Timing	Dynamic registration	Static registration
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)#	Number of SQL statements + number of commit processing communications + 1 (for transaction startup processing) + 1 (communication for establishing connection)#
Reconnection method if connection between transaction manager and HiRDB is broken during processing	Reestablishes connection automatically the next time transaction is started.	Reestablishes connection automatically the next time transaction is started.

#: Applicable when the XA interface supporting multi-thread is used.

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's 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. 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, or the environment variable group name specified when the environment variable settings that take effect at the destination

are registered (for details about how to register environment variables in the registry, see the manual *HiRDB Version 8 UAP Development Guide*)

- Environment variable group ID

Use one of the following formats:

- "*environment-variable-group-identifier* + *environment-variable-setup-file-name* or *environment-variable-group-name*"
- "*environment-variable-group-identifier* * *environment-variable-setup-file-name* or *environment-variable-group-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 or TPBroker 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 or Environment variable group name
HiRDB1	HDB1	hirdb11 hirdb12
HiRDB2	HDB2	hirdb21 hirdb22

The following shows an example of registering the open character string:

```
trnstring -n HiRDB_DB_SERVER -i H1 -o "HDB1*hirdb11" -O "HDB1+hirdb12"
trnstring -n HiRDB_DB_SERVER -i H2 -o "HDB2*hirdb21" -O "HDB2+hirdb22"
```

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 * variable-setup-file-name* or *environment-variable-group-name*.

-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-file-name* or *environment-variable-group-name*.

- Specify the same environment variable group ID for -o and -o.
- Specify the same environment variables in the file or registry 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

Register the open character string using the xa_open_string_info operand in TPBroker's resource manager definition. This example registers two HiRDBs in TPBroker. The registration conditions are as follows:

Resource manager	Environment variable group ID	Environment variable setup file name or Environment variable group name
HiRDB1	HDB1	hirdb11 hirdb12
HiRDB2	HDB2	hirdb21 hirdb22

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*hirdb11"
tsdefvalue /OTS/RM/HIRDB_DB_SERVER_1/xa_open_string_info 2
-s "HDB1+hirdb12"

tsdefvalue /OTS/RM/HIRDB_DB_SERVER_2/DMN/xa_open_string_info 1
-s "HDB2*hirdb21"
tsdefvalue /OTS/RM/HIRDB_DB_SERVER_2/xa_open_string_info 2
-s "HDB2+hirdb22"

```

Explanation:

1. For `/OTS/RM/RM-name/DMN/xa_open_string_info`, specify the open character string that is used by TPBroker's recovery process. Insert an asterisk (*) between the environment variable group identifier and environment variable setup file or environment variable group name.
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. Insert a plus sign (+) between the environment variable group identifier and environment variable setup file or environment variable group name.
 - 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 or environment variable group.
 - 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='
```

(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 Table 7-4.

Table 7-4: Library names for RM-related object names

Library type	Library name
Single thread	pdcltx32.lib
Single thread (Multi-connection facility supported)	pdcltxs.lib*
Multi-thread (Multi-connection facility supported)	pdcltxm.lib

* Specify this library name when linking to TUXEDO.

(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 manual *HiRDB Version 8 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

Specify the client environment definition in TPBroker'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.

(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 `trnlnkrm` command. The following shows examples of `trnlnkrm` command specification:

(a) Dynamic registration

```
trnlnkrm -a HiRDB_DB_SERVER -s pdtxa_switch
-o C:\win32app\hitachi\hirdb_s\client\lib\pdcltx32.lib
```

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

```
trnlnkrm -a HiRDB_DB_SERVER -s pdtxa_switch_y
-o C:\win32app\hitachi\hirdb_s\client\lib\pdcltx32.lib
```

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

To register HiRDB in TPBroker, use TPBroker's `tslnkrm` command. The following shows examples of `tslnkrm` command specification:

(a) Dynamic registration

```
tslnkrm -a HiRDB_DB_SERVER_1 -s pdtxa_switch
-o 'C:\win32app\hitachi\hirdb_s\client\lib\pdcltxm.lib' -r -m
tslnkrm -a HiRDB_DB_SERVER_2 -s pdtxa_switch
-o 'C:\win32app\hitachi\hirdb_s\client\lib\pdcltxm.lib' -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 'C:\win32app\hitachi\hirdb_s\client\lib\pdcltxm.lib' -r -m
tslnkrm -a HiRDB_DB_SERVER_2 -s pdtxa_switch_y
-o 'C:\win32app\hitachi\hirdb_s\client\lib\pdcltxm.lib' -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;C:\HiRDB\client\lib\pdcltxs.lib
```

(b) Static registration

```
HiRDB_DB_SERVER;pdtxa_switch_y;C:\HiRDB\client\lib\pdcltxs.lib
```

(4) WebLogic Server

Register HiRDB using the driver class name and the provider for the WebLogic Server's 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

7.2.8 Modifying the registration information in the transaction manager

To modify registration information in the transaction manager, 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 `trnlnkrm` 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

To modify registration information in the transaction manager when using TPBroker:

1. Use TPBroker's `tslnkrm` command to reregister HiRDB in the transaction manager.
2. Use TPBroker'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 `buildtms` 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*.

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. 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 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 trn_rm_open_close_scope operand in OpenTP1's transaction service definition or user service definition, so that OpenTP1 establishes connection with HiRDB when a transaction is started and disconnects from HiRDB 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.

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.

(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.

7.2.10 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. Even in a multi-thread environment, a transaction can establish connection with only one HiRDB server process. 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 Sever 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) 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 HiRDB External Data Access facility

When the HiRDB External Data Access facility is used, the HiRDB interface can be used to access the tables of multiple databases constructed from other database management systems, including products of other companies. Also, when multiple databases of other types exist, their information can be viewed and updated as one table using foreign table or view table. To use the HiRDB External Data Access facility, HiRDB External Data Access is required.

The HiRDB External Data Access facility can be used only with the HiRDB/Parallel Server.

(1) *HiRDB system definition specifications*

The following operands and definition statements are specified in the HiRDB system definition:

- `pd_max_foreign_server` operand
Specifies the maximum number of foreign servers that can be linked.
- `pdhubopt` operand
Specifies the foreign servers to which Hub optimization information definition is applicable.
- `pd_foreign_server_libpath` operand
Specifies the path names of the client libraries of the foreign servers.
- Environment variables required for foreign servers
The environment variables required for foreign servers are specified in the back-end server definition.
- Foreign server information definition
- Hub optimization information definition

In addition to the operands above, you should also consider the specification of the following operand:

- `pdstbegin` operand
Specifies whether or not the following statistical information related to the HiRDB External Data Access facility is to be collected:
 - Statistical information related to the operation of foreign servers
 - Statistical information related to the usage status of foreign servers

(2) Environment definition specification

For details about the system environment definition in conjunction with using the HiRDB External Data Access facility, see the manual *HiRDB External Data Access Version 7 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/Integrated Management - Manager (JP1/Integrated Manager for JP1 Version 7 or earlier)
- JP1/Automatic Job Management System 2

In this manual, JP1/Integrated Management - Manager is called *JP1/IM* and JP1/Automatic Job Management System 2 is called *JP1/AJS2*.

Windows (x64) 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 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/AJS2 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/AJS2 linkage, see *7.4.3 Automatic job execution using JP1/AJS2 linkage*.

For details about JP1/Base, see the manual *Job Management Partner 1/Base User's Guide*.

(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.

(2) HiRDB events that can be sent

Table 7-5 lists and describes the HiRDB events that can be sent to JP1/Base.

Table 7-5: 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	
					1	2
Start	0x00010C00	KFPS05210 -I or KFPS05219 -I ³	"start"	After the system start completion message (KFPS05210-I or KFPS05219-I) is output	Y	Y
	0x00010C80					
Normal termination	0x00010C01	KFPS01850 -I ³	"end_normal"	After termination of system service (immediately before completion of HiRDB termination or immediately after output of the KFPS01850-I message)	Y	Y
	0x00010C81					
Planned termination	0x00010C01	KFPS01850 -I ³	"end_plan"		Y	Y
	0x00010C81					
Forced termination	0x00010C01	KFPS01850 -I ³	"end_force"		Y	Y
	0x00010C81					
Message log output	0x00010C03	HiRDB message	NULL	After message log is output	Y ⁴	Y ⁴
	0x00010C83					
Abnormal termination	0x00010C02	KFPS01821 -E	NULL	<ul style="list-style-type: none"> Upon HiRDB error termination After error termination is reported to the cluster software when system switchover is used in server mode 	N	Y
	0x00010C82					
Definition change	0x00010C04	KFPS04666 -I	NULL	At completion of HiRDB system normal start	N	Y
	0x00010C84					
RDAREA full	0x00010C05	KFPH00213 -W	NULL	At output of RDAREA-full error message (KFPH00213-W)	N	Y
	0x00010C85					
RDAREA expansion error	0x00010C06	KFPH14229 -E	NULL	Upon pdmod (with expand specified) execution error	N	Y
	0x00010C86					

Event	Event ID ¹	Message	Detail information ²	Registration timing	Value of pd_jp1_event_level	
					1	2
Log file free space warning	0x00010C07	KFPS01160 -E or	NULL	At output of a system log file free-space warning message	N	Y
	0x00010C87	KFPS01162 -W				

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; the bottom code is for a HiRDB/Parallel Server.

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

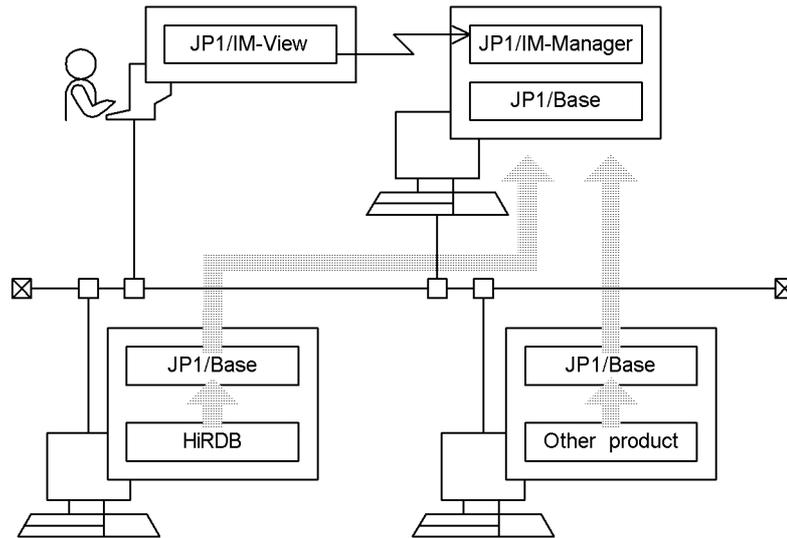
³ When pd_jp1_event_level=1 is specified, the message is null.

⁴ When pd_jp1_event_msg_out=N is specified, the event is not sent.

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. Figure 7-7 provides an overview of event monitoring by JP1/IM.

Figure 7-7: Overview of event monitoring by JP1/IM



 : Flow of JP events

JP1/IM-View: Job Management Partner 1/Integrated Management - View
 JP1/IM-Manager: Job Management Partner 1/Integrated Management - Manager

For an overview of event monitoring by JP1/Integrated Management, see the applicable manual for the JP1 version being used.

- 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:

- JP1/Integrated Management - Manager (for 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 name of the event attribute definition file is

HITACHI_HIRDB_NT_attr_ja.conf.

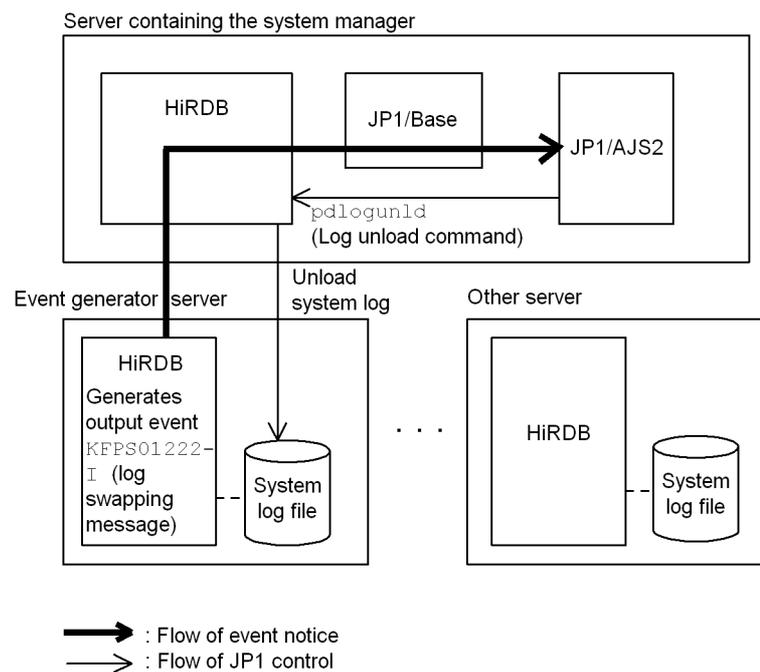
In the case of a multi-HiRDB, copy the most recent version of the event attribute definition file in the running HiRDBs.

7.4.3 Automatic job execution using JP1/AJS2 linkage

In the case of a HiRDB/Parallel Server, operations such as unloading system log files at each server may become complex. If HiRDB events are sent to JP1/Base, you can use JP1/AJS2 to execute jobs automatically on the basis of those events, thereby achieving automation of HiRDB operation.

Figure 7-8 shows 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/AJS2, 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 7i or later
JP1/Automatic Job Management System 2
- For JP1 Version 6

Chapter

8. Designing a HiRDB/Single Server

This chapter describes the design considerations for a HiRDB/Single Server, its HiRDB file system areas, and its system files, and provides notes on the placement of RDAREAs.

This chapter contains the following sections:

- 8.1 System design for a HiRDB/Single Server
- 8.2 Designing HiRDB file system areas
- 8.3 Designing system files
- 8.4 Placing RDAREAs

8.1 System design for a HiRDB/Single Server

This section describes the system design considerations and the system configuration for a HiRDB/Single Server.

8.1.1 System design

(1) Storage requirements

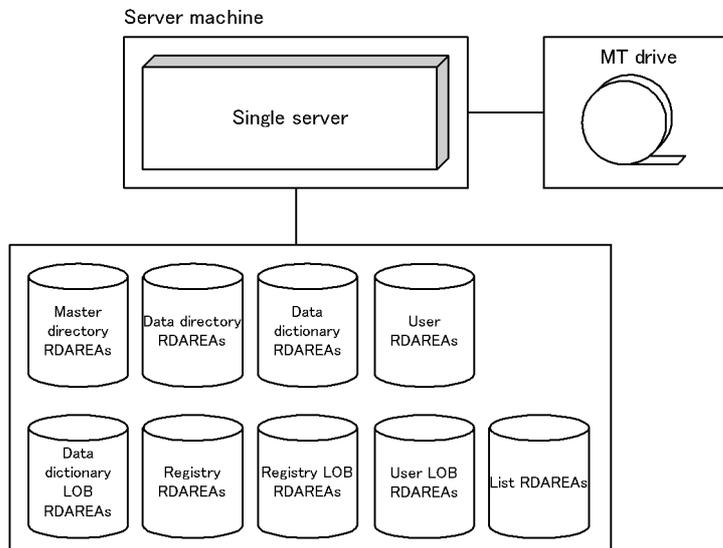
The storage space required by the HiRDB/Single Server must be estimated. For details about the storage requirements for a HiRDB/Single Server, see *15.1 Estimating the memory size required for a HiRDB/Single Server*.

8.1.2 System configuration

Figure 8-1 shows system configurations for a HiRDB/Single Server.

The HiRDB/Single Server system configuration is defined in the HiRDB system definition. For details about definition examples of the system configurations, see the manual *HiRDB Version 8 System Definition*.

Figure 8-1: System configuration for a HiRDB/Single Server



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. 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*

None.

(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.

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. 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.
2. 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.

(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 for improved reliability

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. For details about the sizes of work table files, see *18. Determining Work Table File Size*.

(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. 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 *16. Determining RDAREA Size*.
2. If you use the system switchover facility, create unload log files on a shared disk.
3. 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:

$$\text{Total number of records in system log files to be unloaded} \times \text{record length of system log files} \times \text{number of unload log files to be created} \times 1.2 \div 1048576$$
 - `-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 10.

(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

A list is used to store temporary intermediate results of search processing; therefore, it does not need to be as reliable as other RDAREAs.

(2) Design for improved performance

You should create HiRDB file system areas for list RDAREAs on a 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

8.2.6 Maximum sizes of HiRDB file system areas

Table 8-1 shows the maximum sizes of HiRDB file system areas.

Table 8-1: Maximum sizes of HiRDB file system areas

Condition	Maximum size of HiRDB file system area
<code>pd_large_file_use=N</code> specified (default)	2,047 MB
<code>pd_large_file_use=Y</code> specified	1,048,575 MB

Note

If the size of a HiRDB file system area exceeds 100 GB, specify `pd_ntfs_cache_disable=Y`. If you specify `pd_ntfs_cache_disable=Y`, do not store both a LOB RDAREA and another type of RDAREA in the same HiRDB file system area.

8.2.7 Notes on the use of the system switchover facility

The following points should be considered when the system switchover facility is used

to create a HiRDB file system area:

- Use the `pdfmkfs` command with the `-i` option specified. If the `-i` option is not specified, files might become corrupted if there is a power abnormality at the server machine while the HiRDB file system is being expanded.
- Do not specify `SVR` (the default) in the `-k` option of the `pdfmkfs` command. If `SVR` is specified in the `-k` option, then when the HiRDB file system area is created, HiRDB will continue processing at the point where the Windows file cache was written. Because of this, if an abnormality occurs before data is written to the disk, file integrity may be lost.

8.3 Designing system files

This section describes the design considerations for system files.

8.3.1 Designing system log files

When system log files need to be swapped, HiRDB will terminate abnormally if there are no swappable target system log files. To prevent this, HiRDB has a facility for monitoring the free area for system log files (*monitoring free area for system log files facility*). 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:

Output the `KFPS01162-W` warning message when the percentage of free area for system log files reaches the warning level.

Level 2:

When the percentage of free area for the system log files reaches the warning level, suppress all further scheduling of new transactions, terminate forcibly all transactions in the server, and output the `KFPS01160-E` message. This controls the output volume of the system logs.

If level 2 is selected, all transactions in the server are terminated forcibly when there becomes insufficient free space in the system log files. Because of the severity of this action, the system log files should be designed quite carefully.

The following explains some of the design considerations for system log files.

(1) Design considerations

1. Specify the same record length and number of records for all system log files.
2. The number of system log files that can be created is 6 to 200 groups.
If HiRDB is restarting after an abnormal termination due to insufficient space for the system log files, the number of system log files to be added must be the same as the number that were already created. For example, if 50 groups of 50 system log files had been created, each of the maximum size (2 gigabyte), then 50 groups of 50 system log files of the maximum size should now be added. Therefore, it is recommended that system log files always be created in units of 100 groups.
3. The maximum total size of the system log files is 400 gigabytes.
4. To reduce the number of unload operations, it is advisable to create many large system log files.
5. 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.

6. The amount of space required for one system log file must satisfy the condition shown following.

$$\text{Size of one system log file (bytes)} \geq \lceil (a + 368) \div c \rceil \times c \times b \times d$$

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

7. The total amount of space required for system log files (if dual system log files are used, the total amount of space for one version) must satisfy the condition shown in Formula 1 and 2 below.

Formula 1

$$\text{Total size of all system log files (bytes)} \geq \lceil a \div c \rceil \times 3 \times b$$

a: Output system log size

For details about determining this value, see *17.1.1(2) Determining the size of system log information*.

b: Record length of system log files specified in the `pd_log_rec_leng` operand

c: Substitute the following values:

If `pd_log_rec_leng` = 1024: 1000

If `pd_log_rec_leng` = 2048: 2000

If `pd_log_rec_leng` = 4096: 4000

Formula 2

$$\text{Total size of all system log files (bytes)} \geq 3 \times a \times (b + 1)$$

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. Table 8-2 shows the effects of the number of generations of system log files on operations. The total size of the system log files is unchanged.

Table 8-2: 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) Record length of a system log file

Use the `pdloginit` command's `-l` option to specify the record length of a system log file. You can select 1024, 2048, or 4096 as the record length. By changing the record length, you may be able to reduce the size of system log file.

(a) When constructing a new HiRDB

When constructing a new HiRDB, you should select 1024 as the record length. To do this, specify a value of 1024 in the `pd_log_rec_leng` operand in the server definition.

(b) When already running a HiRDB

Change the record length of system log files on the basis of the average block length that is output to the system log file (`OUTPUT BLOCK LENGTH`), which is obtained from the statistical information about system activity by the statistics analysis utility (`pdstedit`). You should change the record length as follows:

- If the average block length is 1024 bytes or less, change the record length to 1024.
- If the average block length is 1025 to 2048 bytes, change the record length to 2048.
- If the average block length is 2049 bytes or greater, change the record length to 4096.

Notes about changing the record length:

- If records are short, the number of input/output operations in the file increases for large amounts of data; however, the amount of free space is reduced due to rounding up to the next HiRDB record unit, resulting in higher file utilization efficiency.
- If records are long, the number of input/output operations in the file decreases for large amounts of data; however, the amount of free space is increased due to rounding up to the next HiRDB record unit, resulting in lower file utilization efficiency.

How to change the record length:

For details about how to change the record length of system log files, see the manual *HiRDB Version 8 System Operation Guide*.

(4) 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.

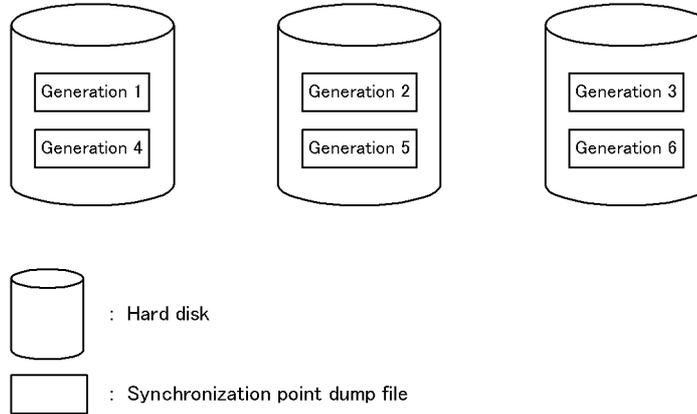
(1) Design considerations

1. Between 2 and 60 groups (if `ONL` is specified, 2 and 30 groups) of synchronization point dump files can be created per server.
2. HiRDB uses synchronization point dump files in the order specified in the `pdlogadfg -d spd` operand.
3. You should create at least four synchronization point dump files.
4. 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 Figure 8-2.

Figure 8-2: Example of creating adjacent generations on separate hard disks (HiRDB/Single Server)

**(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:

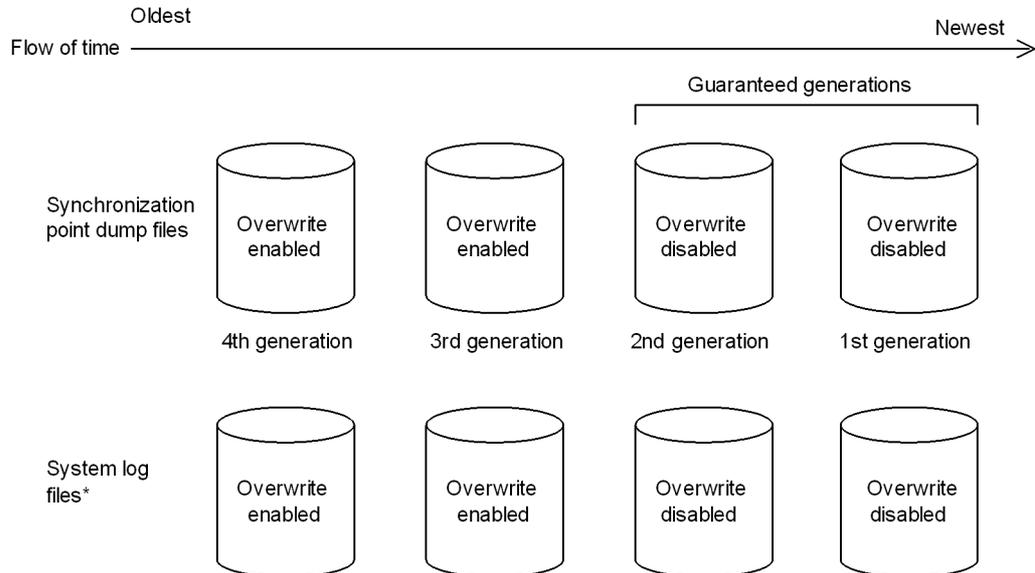
- `pd_spd_dual = Y`
- `-b` option in the `pdlogadpf` operand (specifies the system log file name on system B)

(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 Figure 8-3.

Figure 8-3: Number of guaranteed-valid generations for synchronization point dump files



* These are the system log files that are relevant 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. Make sure that the primary and secondary files have the same record length and the same number of records.
4. Specify the same record length and number of records for both versions (A and

B) of the status files.

5. File versions A and B must be created on separate hard disks.
6. You can create 1-7 sets of unit status files.
7. You can create 1-7 sets of server status files.

(2) Design for improved reliability

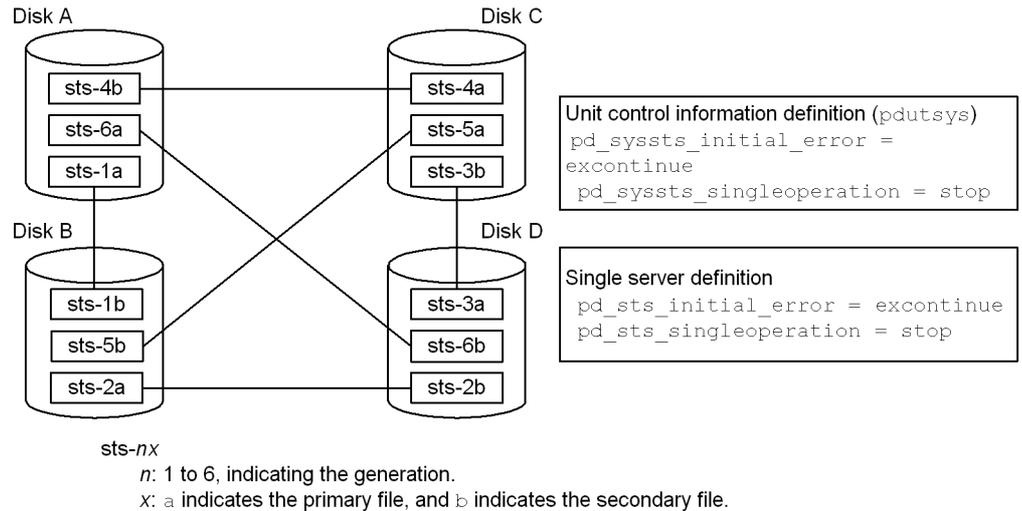
1. Provide at least three sets of status files (dual files \times 3 = 6 files) and place them in such a manner that corruption of all status files by disk errors is unlikely.
2. To prevent abnormal termination of HiRDB resulting from a space shortage, we recommend that you set the size of a status file 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 Figure 8-4. 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`).

Figure 8-4 shows an example of placing six sets of status files on four sets of disks.

Figure 8-4: 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 `pdstssinit` 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 8 System Definition*.

(c) Notes about using single operation

Table 8-3 describes the HiRDB operation and HiRDB administrator's action that depend on whether or not single operation is used. For details about how to handle status file errors, see the manual *HiRDB Version 8 System Operation Guide*.

Table 8-3: HiRDB operation and HiRDB administrator's action that depend on whether or not 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 manual <i>HiRDB Version 8 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 administrator's action: Create spare files immediately and return HiRDB to the double operation mode.	HiRDB operation: Terminates abnormally. 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 manual <i>HiRDB Version 8 System Operation Guide</i> .	

Condition		Status-file single operation (pd_syssts_singleoperation or pd_sts_singleoperation operand value)	
		Used (continue specified)	Not used (omitted or stop specified)
	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 manual <i>HiRDB Version 8 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

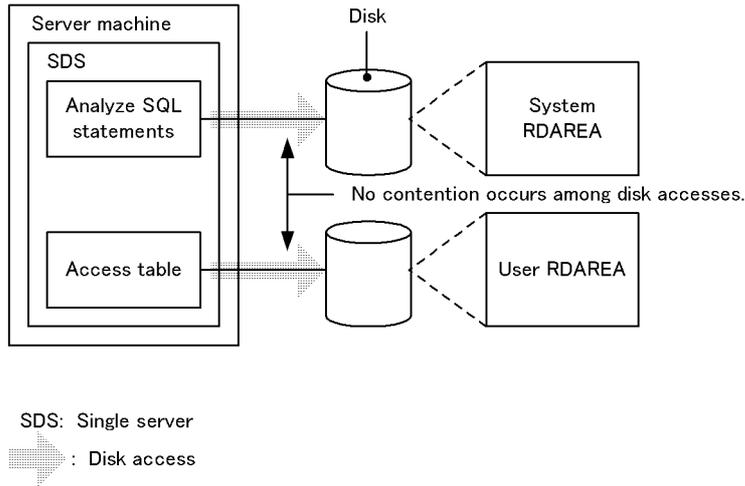
System RDAREAs should be placed taking into account the placement of user RDAREAs. Points to be considered when a system RDAREA is placed are discussed as follows.

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, etc. 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.

Figure 8-5 shows an example of system RDAREA placement that can avoid contention among disk accesses.

Figure 8-5: Example of system RDAREA placement (HiRDB/Single Server)

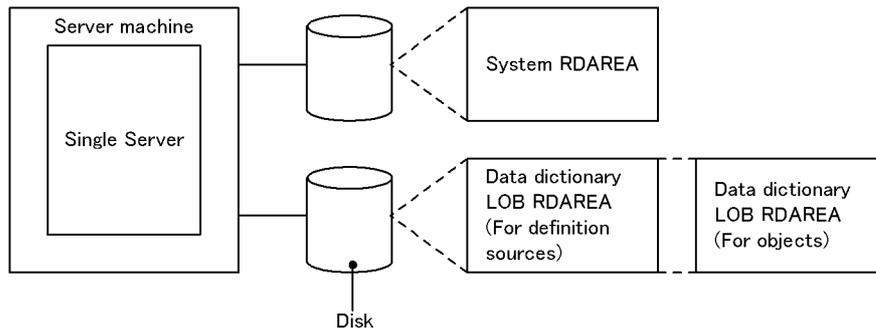


8.4.2 Placing data dictionary LOB RDAREAs

To avoid contention among disk accesses, a data dictionary LOB RDAREA should not be placed on the same disk as any other RDAREA.

Figure 8-6 shows an example of data dictionary LOB RDAREA placement.

Figure 8-6: Example of data dictionary LOB RDAREA placement (HiRDB/Single Server)



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.

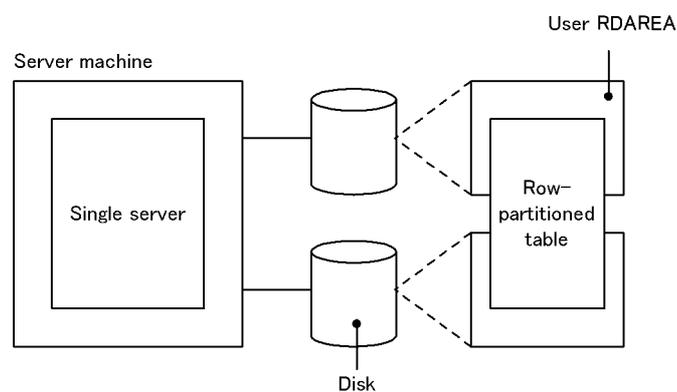
(2) Relationship with system RDAREAs

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. Figure 8-7 shows an example of user RDAREA placement.

Figure 8-7: Example of user RDAREA placement (HiRDB/Single Server)

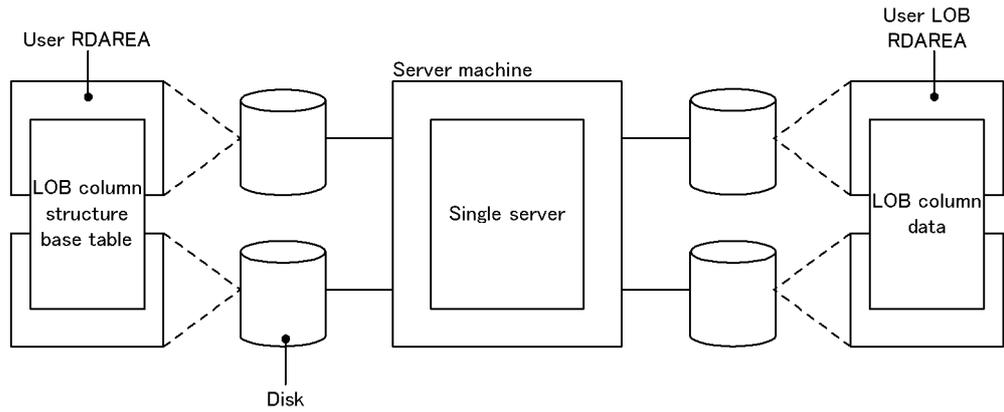


8.4.4 Placing user LOB RDAREAs

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. Figure 8-8 shows an example of user LOB RDAREA placement.

Figure 8-8: Example of user LOB RDAREA placement (HiRDB/Single Server)



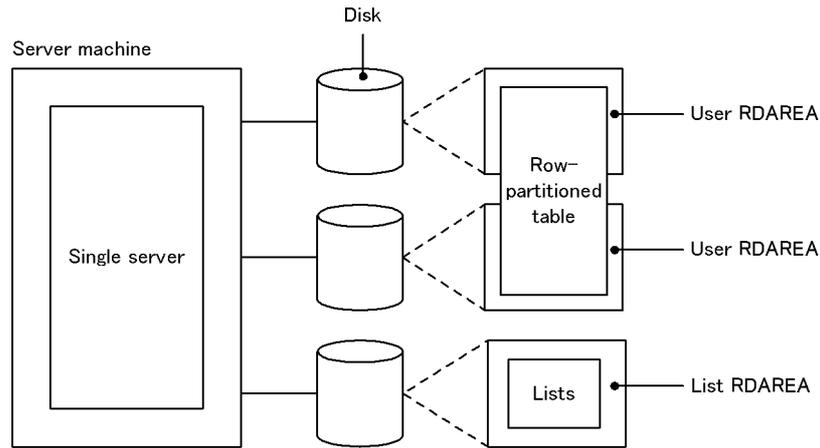
8.4.5 Placing list RDAREAs

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.

Figure 8-9 shows an example of list RDAREA placement.

Figure 8-9: Example of list RDAREA placement (HiRDB/Single Server)



Chapter

9. Designing a HiRDB/Parallel Server

This chapter describes the design considerations for a HiRDB/Parallel Server, its HiRDB file system areas, and its system files, and provides notes on the placement of RDAREAs.

This chapter contains the following sections:

- 9.1 System design for a HiRDB/Parallel Server
- 9.2 Designing HiRDB file system areas
- 9.3 Designing system files
- 9.4 Placing RDAREAs
- 9.5 Heterogeneous system configuration for HiRDB/Parallel Server

9.1 System design for a HiRDB/Parallel Server

This section describes the system design considerations and the system configuration for a HiRDB/Parallel Server.

If the HiRDB External Data Access facility is being used, also see the manual *HiRDB External Data Access Version 7*.

9.1.1 System design

(1) Server configuration

The basic configuration of a HiRDB/Parallel Server 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.

Table 9-1 shows the permitted number of servers you can install.

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.

(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*.

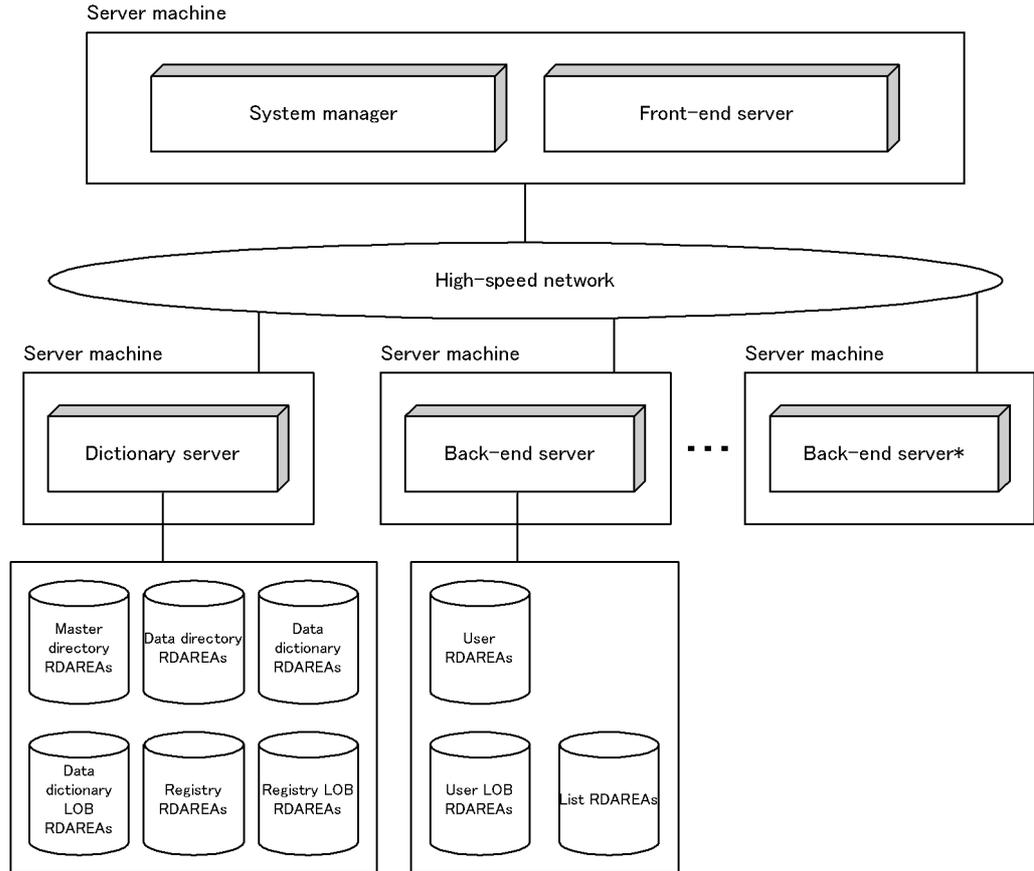
(5) Storage requirements

The storage space required by the HiRDB/Parallel Server 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*.

9.1.2 System configuration of HiRDB/Parallel Server

Figure 9-1 shows an example of a HiRDB/Parallel Server system configuration. The system configuration of a HiRDB/Parallel Server is defined in the HiRDB system definition. For a definition example of the HiRDB system configuration shown in Figure 9-1, see the manual *HiRDB Version 8 System Definition*.

Figure 9-1: System configuration for a HiRDB/Parallel Server



* Used as a floatable server without using RDAREAs

9.1.3 Setting up multiple front-end servers

A HiRDB/Parallel Server uses multiple back-end servers to process multiple SQL processing in parallel. The front-end server is responsible for executing SQL analysis and SQL optimization, sending instructions to the back-end servers, editing retrieval results, etc. 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.

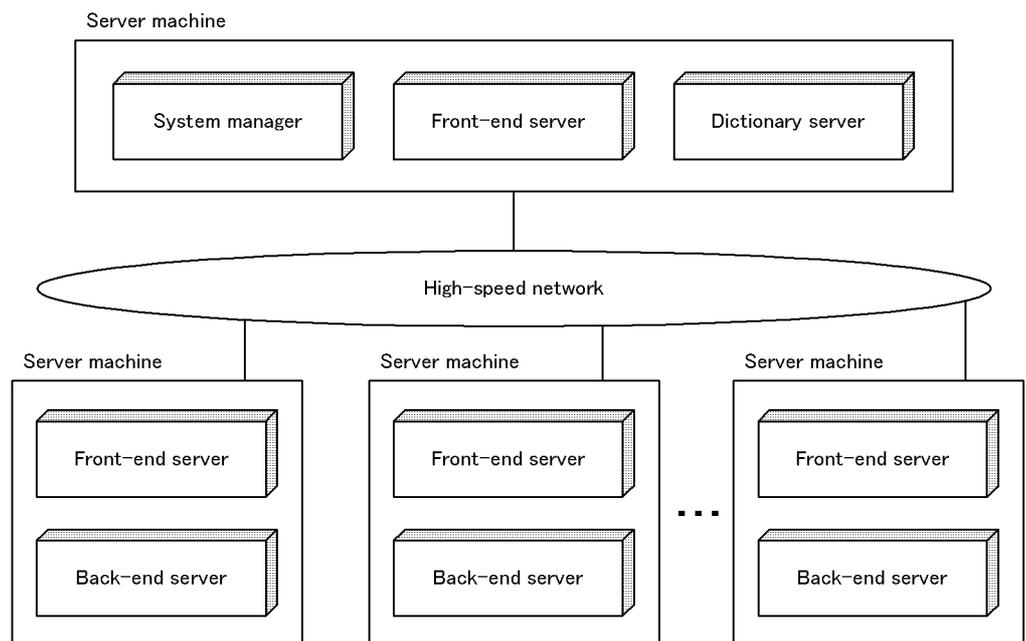
Rules

A maximum of 1,024 front-end servers can be installed.

Relationship to server machine

Only one front-end server can be installed on a server machine. Figure 9-2 shows an example of a configuration that uses multiple front-end servers.

Figure 9-2: System configuration for a HiRDB/Parallel Server 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, etc., 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 8 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.

Table 9-2 describes the server that acquires the current timestamp depending on

whether or not USING BES is specified.

Table 9-2: Server that acquires the current timestamp depending on whether or not USING BES is specified

USING BES	Server that acquires the current timestamp	
	HiRDB/Single	HiRDB/Parallel Server
Not specified	Single server	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. Table 9-3 lists the required client environment definitions. For details about the client environment definitions, see the manual *HiRDB Version 8 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

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
PDSERVICEPORT	—	—	M
PDSERVICEGRP	—	M	M
PDSRVTYPE	—	—	M

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, UAP processing under OLTP, etc.

(b) HiRDB server connection time

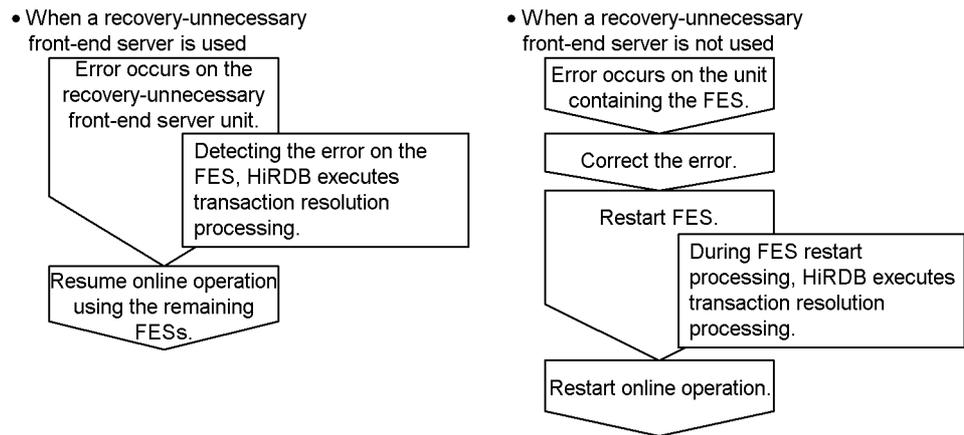
The HiRDB server connection time increases from 1 to 3 as follows (with 1 being the shortest):

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*. Figure 9-3 shows the operation depending on whether or not a recovery-unnecessary front-end server is used.

Figure 9-3: Operation based on whether or not 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

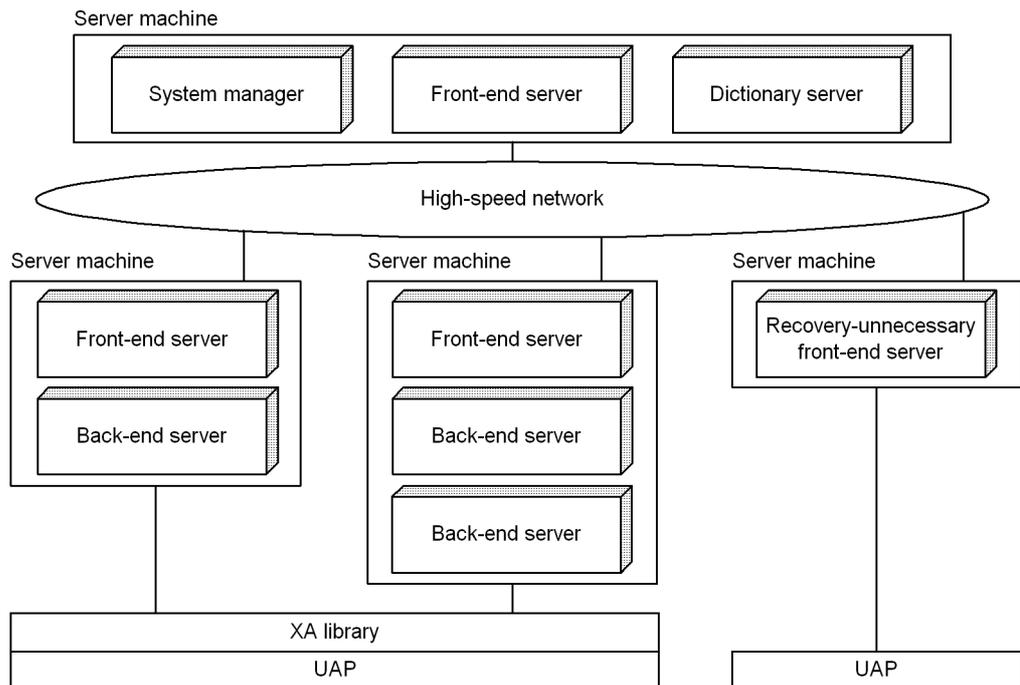
We recommend that you use recovery-unnecessary front-end servers in a system that requires non-stop operation 24 hours a day.

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.

Figure 9-4 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_sqlc` 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 `PDFSERVICEGRP` 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. For the back-end server or dictionary server that contains the data whose manipulation was attempted by the SQL request, `c900` may be displayed as the process termination status in the KFPS01820-E message. 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. If the status information for a recovery-unnecessary front-end server unit is `STOP(A)` due to a network error even though the unit is not running, the system manager automatically terminates that unit forcibly and then restarts it once the error has been recovered and the system manager can communicate with the unit.

If the unit is terminated before communication between the system manager and the unit becomes available, the system manager does not attempt forced termination nor does it restart the unit. If necessary, use the `pdstart` command to start the unit. In the following cases, take appropriate action depending on the situation:

- When an error message has been displayed for the applicable unit

Before starting the unit, check the message to determine the cause of the error and then take appropriate action. If the unit is stopped, the KFPS01841-I or KFPS01821-E message is displayed in the system log. The KFPS01853-W, KFPS01863-E, or KFPS05619-E message is not displayed when the `pdstart` command is executed.
 - When the system manager's attempt to restart the unit failed and the unit remained in terminated status

Check the message to determine the cause of the start error, and take appropriate action. After that, start the applicable unit with the `pdstart` command, if necessary.
7. For a transaction that is processed at the back-end server or dictionary server branched out from the recovery-unnecessary front-end server, its completion is synchronized with the target back-end server or dictionary 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 start, or the server is not ready for termination), processing may be queued with the first transaction status 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`, `pdrbk`, 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 manual *HiRDB Version 8 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. 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. HiRDB file system areas for RDAREAs must be created on a server machine where the following servers are defined:
 - Dictionary server
 - Back-end server
2. 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
3. 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

(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.

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. 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.
2. 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.

(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. 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. For details about the sizes of work table files, see *18. Determining Work Table File Size*.
2. 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

(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. 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 *16. Determining RDAREA Size*.

2. Create a HiRDB file system area for differential backup management files on the server machine where the system manager is located.
3. 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:

$$\text{Total number of records in system log files to be unloaded} \times \text{record length of system log files} \times \text{number of unload log files to be created} \times 1.2 \div 1048576$$
 - `-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 10.

(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.
2. 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. 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

Table 9-4 shows the maximum sizes of HiRDB file system areas.

Table 9-4: Maximum sizes of HiRDB file system areas

Condition	Maximum size of HiRDB file system area
pd_large_file_use=N specified (default)	2,047 MB
pd_large_file_use=Y specified	1,048,575 MB

Note

If the size of a HiRDB file system area exceeds 100 GB, specify `pd_ntfs_cache_disable=Y`. If you specify `pd_ntfs_cache_disable=Y`, do not store both a LOB RDAREA and another type of RDAREA in the same HiRDB file system area.

9.2.7 Notes on the use of the system switchover facility

The following points should be considered when the system switchover facility is used to create a HiRDB file system area:

- Use the `pd_fmks` command with the `-i` option specified. If the `-i` option is not specified, files might become corrupted if there is a power abnormality at the server machine while the HiRDB file system is being expanded.
- Do not specify `SVR` (the default value) in the `-k` option of the `pd_fmks` command. If `SVR` is specified in the `-k` option, then when the HiRDB file system area is created, HiRDB will continue processing at the point where the Windows file cache was written. Because of this, if an abnormality occurs before data is written to the disk, file integrity may be lost.

9.3 Designing system files

This section discusses design considerations for various system files.

9.3.1 Designing system log files

When system log files need to be swapped, HiRDB (the unit) will terminate abnormally if there are no swappable target system log files. To prevent this, HiRDB has a facility for monitoring the free area for system log files (*monitoring free area for system log files facility*). 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:

Output the KFPS01162-W warning message when the percentage of free area for the system log files reaches the warning level.

Level 2:

When the percentage of free area for the system log files reaches the warning level, suppress all further scheduling of new transactions, terminate forcibly all transactions in the server, and output the KFPS01160-E message. This controls the output volume of the system logs.

If level 2 is selected, all transactions in the server are terminated forcibly when there becomes insufficient free space in the system log files. Because of the severity of this action, the system log files should be designed quite carefully.

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 6-200 groups.
If HiRDB is restarting after an abnormal termination due to insufficient space for the system log files, the number of system log files to be added must be the same as the number that were already created. For example, if 50 groups of 50 system log files had been created, each of the maximum size (2 gigabyte), then 50 groups of 50 system log files of the maximum size should now be added. Therefore, it is recommended that system log files always be created in units of 100 groups.
4. The maximum total size of the system log files is 400 gigabytes per 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:

$$\text{Size of one system log file (bytes)} \geq \lceil (a + 368) \div c \rceil \times c \times b \times d$$

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 amount of space required for system log files (if dual system log files are used, the total amount of space for one version) must satisfy the condition shown in Formula 1 and 2 below.

Formula 1

$$\text{Total size of all system log files (bytes)} \geq \lceil a \div c \rceil \times 3 \times b$$

a: Output system log size

For details about determining this value, see *17.1.1(2) Determining the size of system log information*.

b: Record length of system log files specified in the `pd_log_rec_leng` operand

c: Substitute the following values:

If `pd_log_rec_leng` = 1024: 1000

If `pd_log_rec_leng` = 2048: 2000

If `pd_log_rec_leng` = 4096: 4000

Formula 2

$$\text{Total size of all system log files (bytes)} \geq 3 \times a \times (b + 1)$$

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. Table 9-5 shows the effects of the number of generations of system log files on operations. The total size of the system log files is unchanged.

Table 9-5: 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) Record length of a system log file

Use the `pdloginit` command's `-l` option to specify the record length of a system log file. You can select 1024, 2048, or 4096 as the record length. By changing the record length, you may be able to reduce the size of system log file.

(a) When constructing a new HiRDB

When constructing a new HiRDB, you should select 1024 as the record length. To do this, specify a value of 1024 in the `pd_log_rec_leng` operand in the server

definition.

(b) When already running a HiRDB

Change the record length of system log files on the basis of the average block length that is output to the system log file (`OUTPUT_BLOCK_LENGTH`), which is obtained from the statistical information about system activity by the statistics analysis utility (`pdstedit`). You should change the record length as follows:

- If the average block length is 1024 bytes or less, change the record length to 1024.
- If the average block length is 1025 to 2048 bytes, change the record length to 2048.
- If the average block length is 2049 bytes or greater, change the record length to 4096.

Notes about changing the record length:

- If records are short, the number of input/output operations in the file increases for large amounts of data; however, the amount of free space is reduced due to rounding up to the next HiRDB record unit, resulting in higher file utilization efficiency.
- If records are long, the number of input/output operations in the file decreases for large amounts data; however, the amount of free space is increased due to rounding up to the next HiRDB record unit, resulting in lower file utilization efficiency.

How to change the record length:

For details about how to change the record length of system log files, see the manual *HiRDB Version 8 System Operation Guide*.

(4) 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.

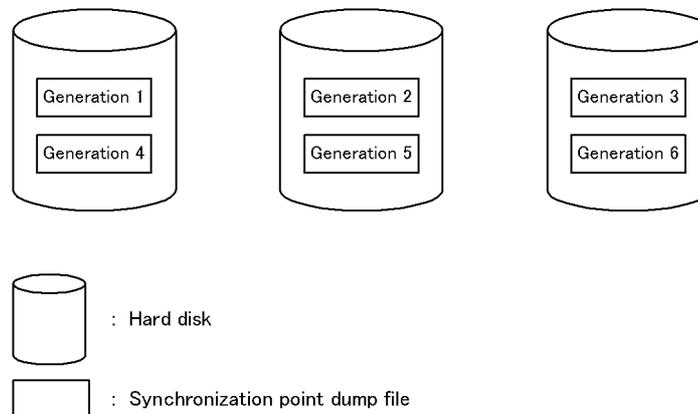
4. HiRDB uses synchronization point dump files in the order specified in the `pdlogadfg -d spd` operand.
5. If a shortage of space occurs in a synchronization point dump file, HiRDB cannot be restarted. For this reason, the size of a synchronization point dump file should be set to be greater than the 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 Figure 9-5.

Figure 9-5: Example of creating adjacent generations on separate hard disks (HiRDB/Parallel Server)



(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.

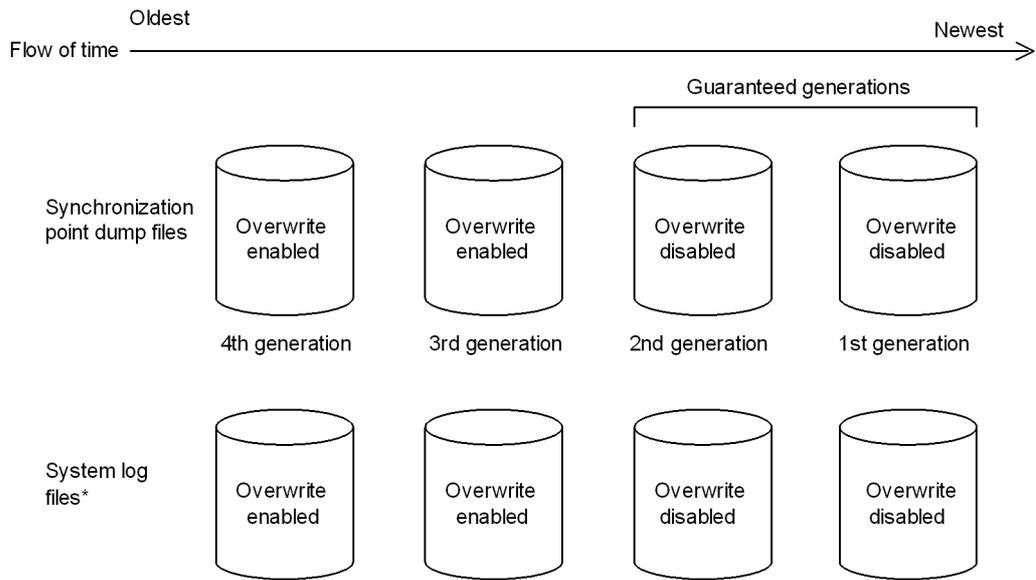
Specify the following operands in the server definition to enable duplexing of synchronization point dump files:

- `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 Figure 9-6.

Figure 9-6: Number of guaranteed-valid generations for synchronization point dump file (HiRDB/Parallel Server)



* These are the system log files that are relevant 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

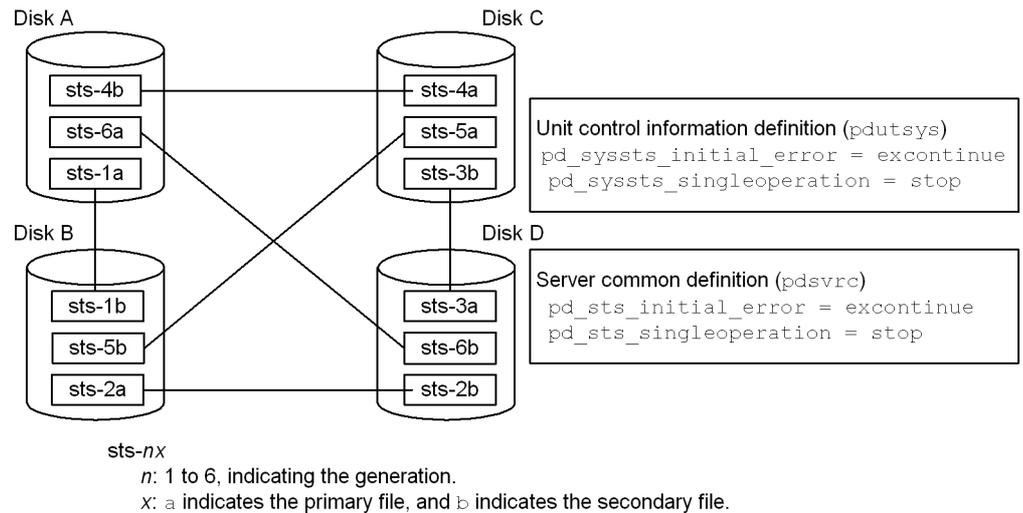
1. Provide at least three sets of status files (dual files \times 3 = 6 files) and place them in such a manner that corruption of all status files by disk errors is unlikely.
2. To prevent abnormal termination of HiRDB resulting from a space shortage, we recommend that you set the size of a status file 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 Figure 9-7. 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`).

Figure 9-7 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 `pdstsinic` 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 8 System Definition*.

(c) Notes about using single operation

Table 9-6 describes the HiRDB operation and HiRDB administrator's action that depend on whether or not single operation is used. For details about how to handle status file errors, see the manual *HiRDB Version 8 System Operation Guide*.

Table 9-6: HiRDB operation and HiRDB administrator's action that depend on whether or not 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 manual <i>HiRDB Version 8 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 administrator's action: Create spare files immediately and return HiRDB to the double operation mode.	HiRDB operation: Terminates abnormally. 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 manual <i>HiRDB Version 8 System Operation Guide</i> .	

Condition		Status-file single operation (pd_syssts_singleoperation or pd_sts_singleoperation operand value)	
		Used (continue specified)	Not used (omitted or stop specified)
	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 manual <i>HiRDB Version 8 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

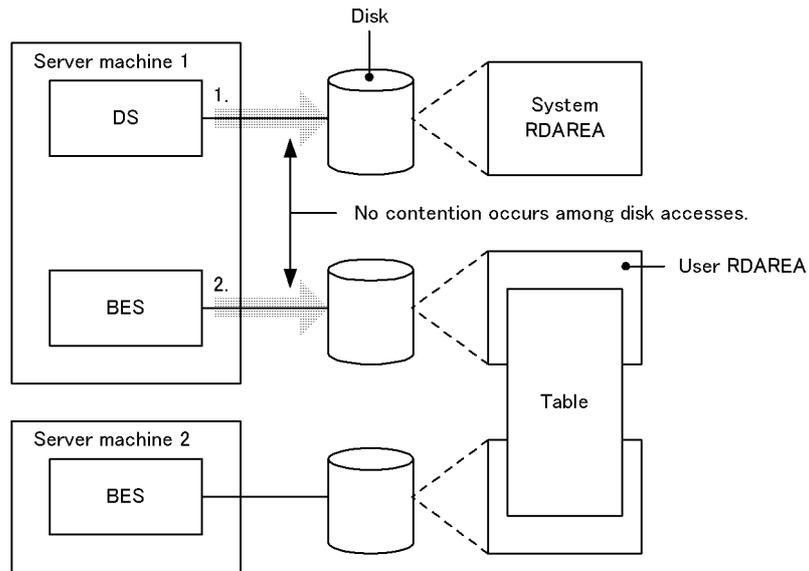
System RDAREAs should be placed taking into account the placement of user RDAREAs. Points to be considered when a system RDAREA is placed are discussed below.

- **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.

Figure 9-8 shows an example of system RDAREA placement that can avoid contention among disk accesses.

Figure 9-8: Example of system RDAREA placement (HiRDB/Parallel Server)



DS: Dictionary server

BES: Back-end server

➔ : Disk access

Explanation:

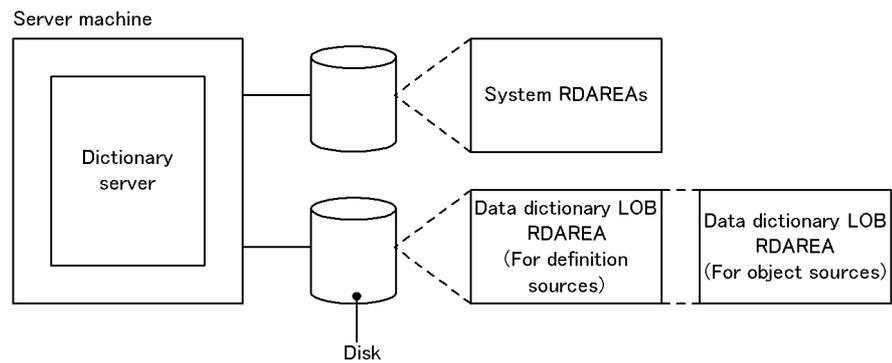
1. Access for the purpose of analyzing SQL statements, etc.
2. Access to the table

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.

Figure 9-9 shows an example of data dictionary LOB RDAREA placement.

Figure 9-9: Example of data dictionary LOB RDAREA placement (HiRDB/Parallel Server)



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.

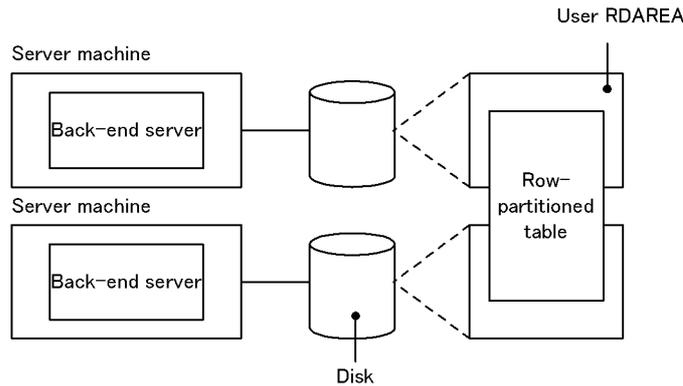
(2) Relationship with system RDAREAs

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. Figure 9-10 shows an example of user RDAREA placement.

Figure 9-10: Example of user RDAREA placement (HiRDB/Parallel Server)



(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.

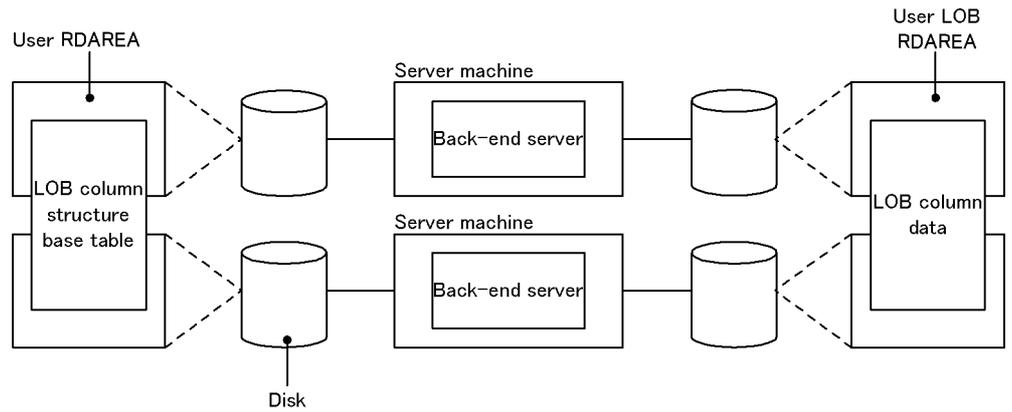
9.4.4 Placing user LOB RDAREAs

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, 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.

Figure 9-11 shows an example of user LOB RDAREA and user RDAREA placement.

Figure 9-11: Example of user LOB RDAREA and user RDAREA placement (HiRDB/Parallel Server)



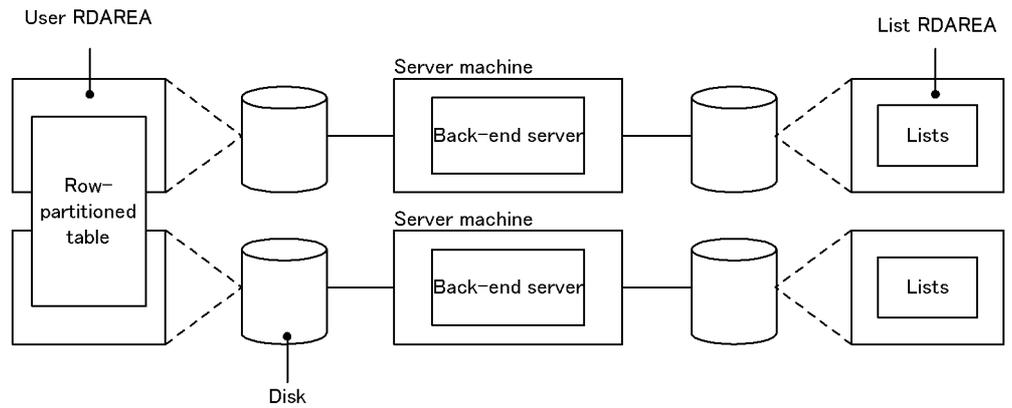
9.4.5 Placing list RDAREAs

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. Figure 9-12 shows an example of list RDAREA placement.

Figure 9-12: Example of list RDAREA placement (HiRDB/Parallel Server)



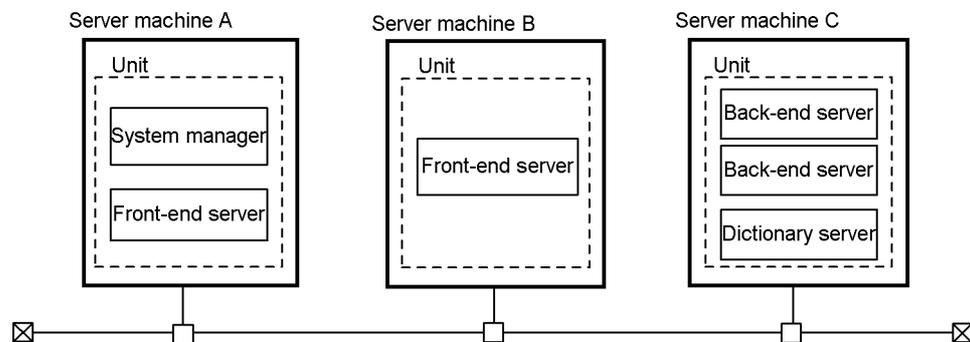
9.5 Heterogeneous system configuration for HiRDB/Parallel Server

Normally, all HiRDB/Parallel Server units must be on the same platform; however, you can configure a heterogeneous system for HiRDB/Parallel Server in which different platforms coexist, as long as the following conditions are met:

- All front-end servers run on the same platform (Windows Server 2003 (IPF) or Windows (x64)).
- All back-end and dictionary servers run on the same platform (Windows Server 2003 (IPF) or Windows (x64)).
- The system manager runs on the same platform as the front-end or back-end servers

Figure 9-13 shows an example of a heterogeneous system configuration.

Figure 9-13: Example of heterogeneous system configuration for HiRDB/Parallel Server



Explanation:

- This HiRDB/Parallel Server consists of three server machines. In this example, server machines A and B serve as application servers and server machine C serves as a database server.
- The platform of server machines A and B is Windows (x64), and the platform of server machine C is Windows Server 2003 (IPF).

Note:

If a heterogeneous system configuration is used, you must abide by the following rule when you move a back-end server or a unit containing a back-end server.

- When moving a back-end server or a unit containing a back-end server, the platform of the server machine before and after moving must be the same.

You should not move the server to a different platform.

Chapter

10. Designing a Multi-HiRDB

This chapter describes the system design considerations for a multi-HiRDB.

This chapter contains the following sections:

- 10.1 System design for a multi-HiRDB
- 10.2 Notes about operation

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) *Notes about HiRDB installation*

1. To install a HiRDB at the server machine where HiRDB has already been installed by the standard setup method, select **Setup with Identifier** during installation and specify the setup identifier. A setup identifier is a string of 1-4 alphanumeric characters (case sensitive) including no space that is unique within a single server machine.
2. A setup identifier is specified to identify a HiRDB environment.
3. Specify a different installation directory for each HiRDB to be installed.
4. HiRDB 05-05 or a later version allows multiple HiRDBs to be installed. Do not mix these versions with HiRDB 05-04 or an earlier version.
5. You cannot mix a HiRDB/Single Server with a HiRDB/Parallel Server.
6. To use a product that does not support multi-HiRDB, you need to select the standard installation method (format with no setup identifier) for the HiRDB.

For details about how to install HiRDB, see *2.2 HiRDB installation procedure*.

(3) *Specifying the communication port's service name and service port number (for a HiRDB/Parallel Server or for a HiRDB/Single Server that uses the system switchover facility)*

If you install a HiRDB specifying a setup identifier, you need to specify the communication port's service name and service number in the `SERVICES` file before starting HiRDB services. The following shows the procedure from selection to specification of the communication port's service name and service port number.

(a) **Checking the system configuration**

Determine the service name and service port number for the communication port that are the same in the HiRDB system and that are unique within the server machine.

(b) Registering the service port number in the OS

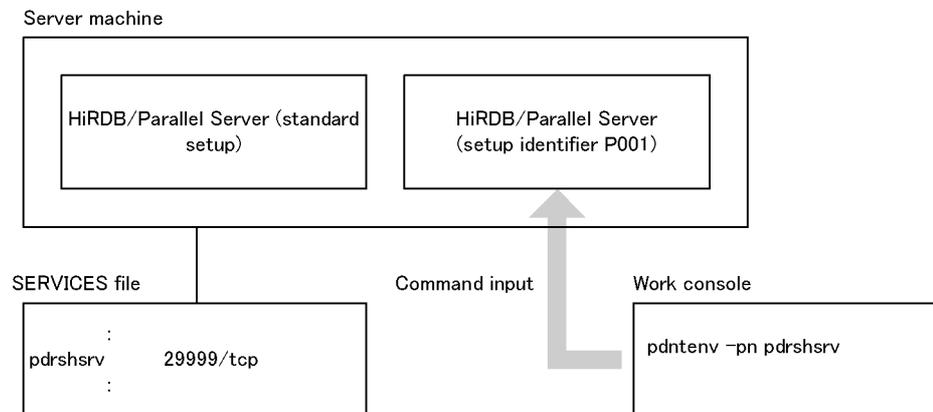
In the `SERVICES` file (under `%windir%\system32\drivers\etc`), register the communication port's service name and service port number determined in (a) previously that are the same in the HiRDB system and that are unique within the service machine.

(c) Changing the communication port's service name

Specify the communication port's service name, registered in (b) previously, for each unit. For the units in the HiRDB server, use the same service name and service port number for the communication port. To specify the service port number, use the `pdntenv` command.

Immediately after the installation, the system assumes `pdrshsrv` as the communication port's service name if you have used the standard setup method and `pdrshsrv` with the setup identifier if you have specified the setup identifier. Figure 10-1 shows an example of a service name and service port number for the communication port when a multi-HiRDB consists of a HiRDB/Parallel Server installed by the standard setup method and another HiRDB/Parallel Server installed with a setup identifier (P001) specified.

Figure 10-1: Example of service name and service port number for communication port



Explanation:

1. Specify the communication port's service name `pdrshsrv` and service port number `29999` in the `SERVICES` file.
2. Use the `pdntenv` command to change the communication port's service name to `pdrshsrv` for the HiRDB/Parallel Server with setup identifier `P001`.

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.

(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 manual *HiRDB Version 8 UAP Development Guide*.

10.2 Notes about operation

(1) Operation commands and utilities

To execute HiRDB operation commands and utilities on a HiRDB that has been installed with a setup identifier specified, you need to specify environment variables. HiRDB provides an environment variable specification batch file and a work console that specifies the environment variables according to the operation command and utility execution environment.

(a) How to use the work console

To start the work console, from **Start**, choose **Program, HiRDBSingleServer Setup Identifier** or **HiRDBParallelServer Setup Identifier**, then **HiRDB Command Prompt**. Use this work console to execute operation commands and utilities.

(b) How to use the environment variable specification batch file

Copy the environment variable specification batch file (%PDDIR%\SAMPLE\sampleconf\HiRDBCMD.BAT) under a different name and then customize the file contents. If you customize the original batch file, it is overwritten during update installation. Use the copy of the environment variable specification batch file for executing operation commands and utilities.

To execute operation commands and utilities at the command prompt, you need the following environment variables:

Environment variable	Description
PDDIR	Specify the HiRDB installation directory.
PDCONFPATH	Specify the absolute path of the directory where HiRDB system definition files are to be stored. For unit control information, the system uses a file under %PDDIR%\conf regardless of this specification.
PATH	Add %PDDIR%\BIN\; %PDDIR%\CLIENT\UTL; at the beginning of the PATH environment variable.
PDUXPLDIR	Specify %PDDIR%\UXPLDIR.
PDDFUXPLDIR	Specify %PDDIR%\DFUXPLDIR.

(2) Specifying the port number for connection from a client

For the HiRDB port number for connection from a client, specify a unique port number in each HiRDB. To specify the port number, use the `pd_name_port` operand in the system common definitions. For details about the `pd_name_port` operand, see the manual *HiRDB Version 8 System Definition*.

(3) Starting and terminating HiRDB with setup identifier

If you have installed a HiRDB specifying a setup identifier, the setup identifier is required to start or terminate a service. To start or terminate a service, use the service name with the specified setup identifier attached.

(4) Relationship with the system switchover facility

(a) How to register in MSCS

To use the HiRDB in the system switchover configuration by registering it in MSCS, the primary system and the secondary system must have the same setup identifier.

(b) Migrating to system switchover configuration

To use the HiRDB in the system switchover configuration using MSCS, start the same service name at a different server machine. If there is an existing environment, you need to change the existing service name to avoid duplication. To change a service name, uninstall the HiRDB and then reinstall it in the same directory. If the directory structure is duplicated, you need to migrate the HiRDB server and create a new environment. For details about HiRDB server migration, see the manual *HiRDB Version 8 System Operation Guide*.

Chapter

11. Designing Global Buffers and Local Buffers

This chapter describes global buffer and local buffer design.

This chapter contains the following sections:

- 11.1 Allocating global buffers
- 11.2 Setting the number of global buffer sectors
- 11.3 Specifying the prefetch facility
- 11.4 Specifying the asynchronous READ facility
- 11.5 Specifying deferred write processing
- 11.6 Specifying the facility for parallel writes in deferred write processing
- 11.7 Setting the commit-time reflection processing
- 11.8 Global buffer LRU management
- 11.9 Page access using the snapshot method
- 11.10 Global buffer pre-writing
- 11.11 Local buffers

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 `pddbufmod` command is used for dynamic updating. For details about dynamic updating of global buffers, see the manual *HiRDB Version 8 System Operation Guide*.

This section describes the various methods of allocating global buffers.

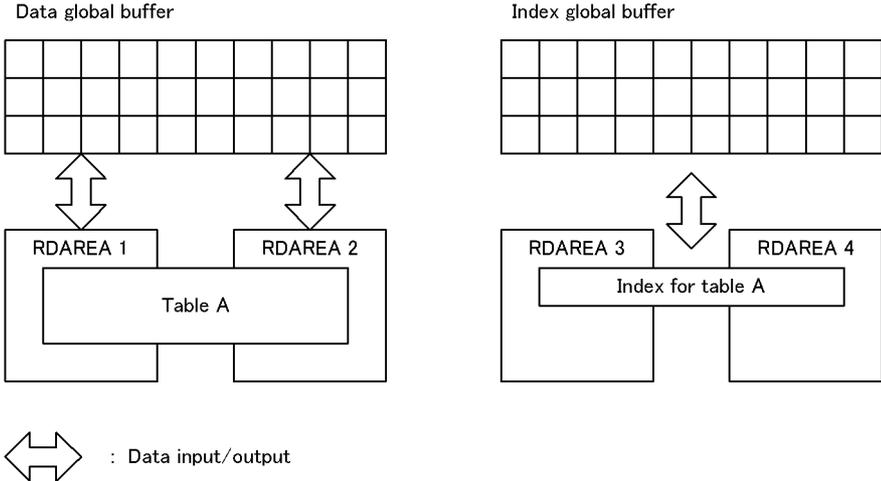
11.1.1 Allocating index global buffers

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.

Figure 11-1 provides an overview of a global buffer dedicated to an index.

Figure 11-1: Overview of global buffer for an index



11.1.2 Allocating data global buffers

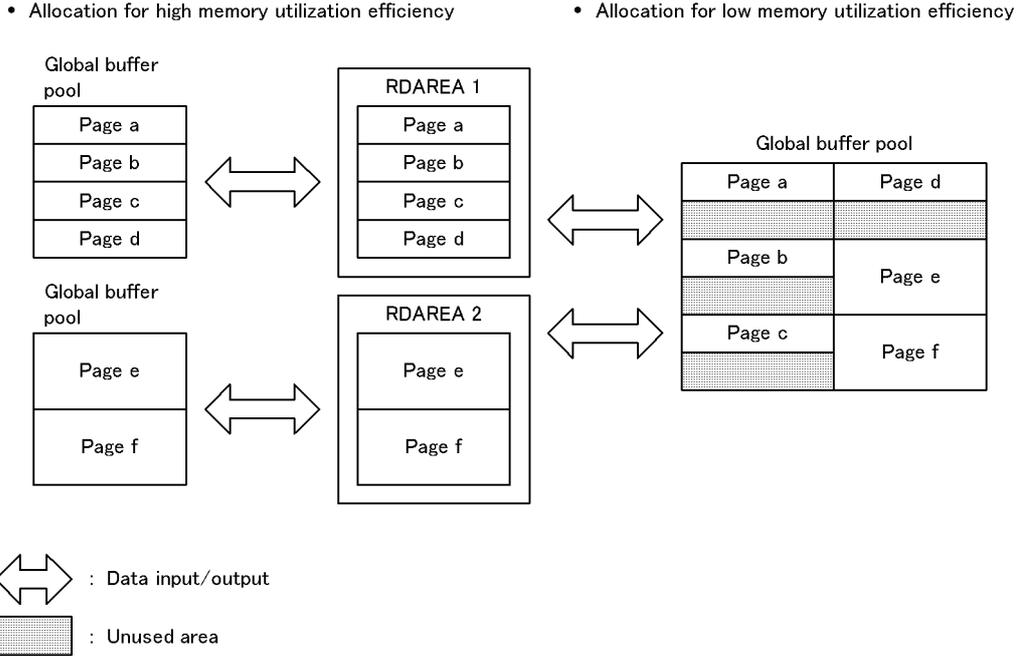
(1) Multiple RDAREAs with different page lengths

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.

Figure 11-2 shows an example of data global buffer allocation.

Figure 11-2: Example of data global buffer allocation



For a HiRDB/Parallel Server, 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 4096 bytes and the largest RDAREA page length on back-end server 2 is 8192 bytes, the global buffer sizes that could be allocated would be 4096 for back-end server 1 and 8192 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

The index identifier of the index for which a global buffer is to be allocated is specified in the `-i` option of the `pdbuffer` operand in the system common definition. In the case of a cluster key index, the index identifier assigned by HiRDB is specified. Once a table with a cluster key has been defined, the index identifier for the cluster key can be obtained by retrieving from the dictionary tables the `INDEX_NAME` column of the `SQL_INDEXES` table. In this case, the index identifier for the cluster key is displayed as follows:

(CLUSTER *table-number*)

For details about dictionary table retrieval and the `SQL_INDEXES` table, see the manual *HiRDB Version 8 UAP Development Guide*.

The number of global buffer sectors should at least equal the number of index pages (value obtained as the index storage page count). It may be necessary to increase or decrease this number depending on the importance of the index. The number of index pages in use can be checked with the database condition analysis utility (`pddbst`).

(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*

Global buffers are allocated in shared memory. This shared memory is allocated as files on the disk that contains the HiRDB directory. Therefore, you should provide sufficient free disk space.

A maximum of 16 shared memory segments can be allocated with the size specified with `SHMMAX` in the system common definitions. Therefore, you need to define a global buffer within the range of the `SHMMAX` value (MB) times 16.

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 `pdbuf1s` command or the statistics analysis utility (`pdstedit`) can be used after the HiRDB system has been started.

If the `pdbuf1s` 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 `pdbuf1s` 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 direct disk accesses (raw I/O facility), 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 (`pdreorg`) 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 *15. Storage Requirements for HiRDB*.

- Whether or not 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 the prefetch facility is used with batch input of multiple pages into global buffers, synchronous processing is used for batch input to pre-read from the database process into the batch input buffers. 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 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, 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 direct disk access (raw/IO facility), for which the processing time is high. On the other hand, there is little benefit for operations that require little input/output time, such as the Windows file system or Lightning/Thunder disks.

(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 *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

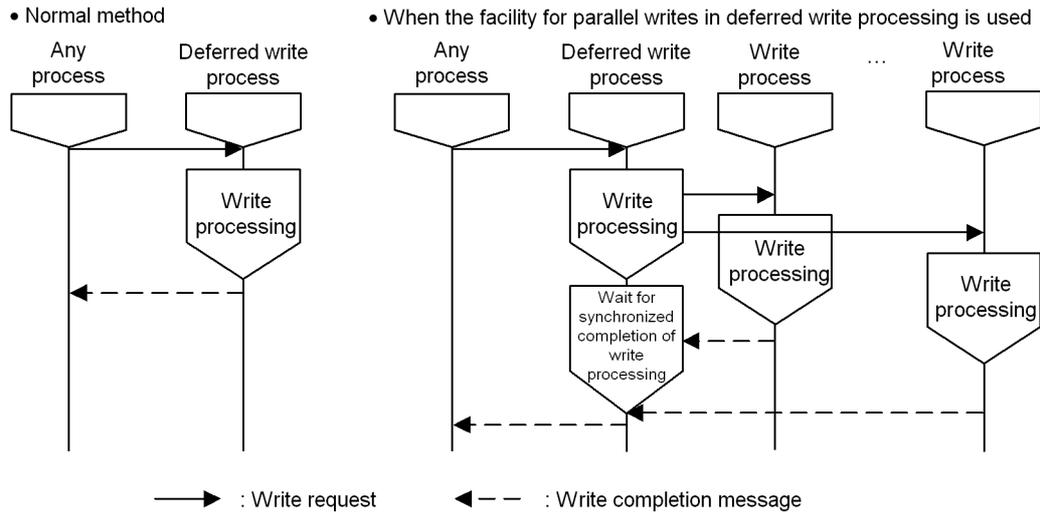
If the updated output page rate for deferred write trigger is set to a higher value than is necessary, disk write operations occur more frequently during deferred write processing. This may cause a concurrently executing transaction to be placed in input/output wait status, with adverse effects on the 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. Figure 11-3 provides an overview of the processing when the facility for parallel writes in deferred write processing is and is not used.

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

The time required for writing data to disk is reduced because write processing is executed as multiple processes.

(2) Specification

Specify the number of write processes in the `pd_dfw_await_process` operand. Also specify the deferred write trigger request ratio in the `pd_dbbuff_rate_updpage` operand. If the `pd_dfw_await_process` operand is omitted, the facility for parallel writes in deferred write processing is disabled.

(3) Considerations

When the facility for parallel writes in deferred write processing is specified, the number of processes increases, thereby increasing the CPU usage.

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 or not 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 `pdbuf1s` 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) **Notes**

If a large amount of update processing occurs, the reference buffer hit rate drops, slowing down retrieval processing.

(c) **Specification**

`SEPARATE` is specified in `pd_dbbuff_lru_option` in the system common definition.

(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) 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.

(c) 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.

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.

(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.

(3) 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 PDDBLOG 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.

$$\text{update-get-count} * \text{pd_log_rec_leng-operand-value}$$

* To determine the update-get-count, check the value of DIDUC in the UAP statistical report or in the UAP statistical information.

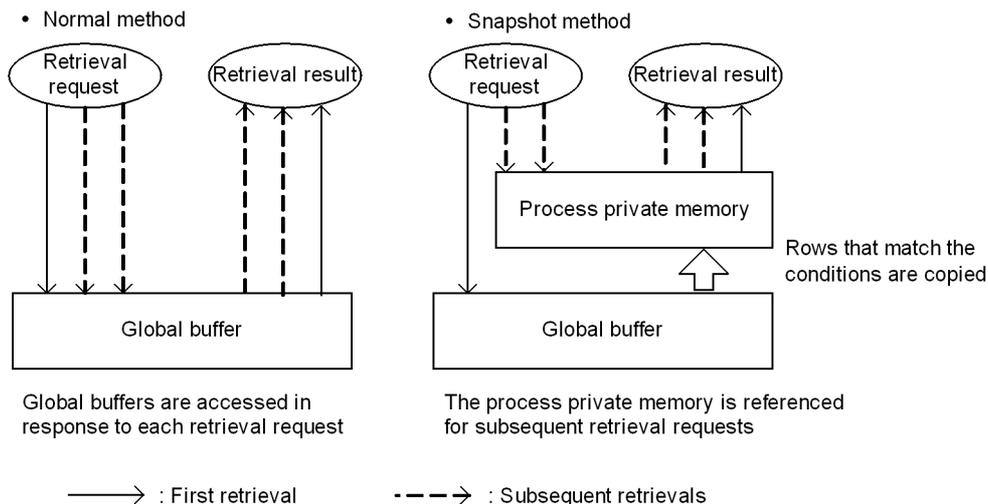
(4) Specification

Specify NO in the PDDBBUFLRU operand in the client environment definition.

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. Figure 11-4 provides an overview of the snapshot method.

Figure 11-4: Overview of the snapshot method



(1) Effectiveness of accesses using the snapshot method

Because the rows that match the conditions of a retrieval request are copied into the process private memory at the time of the first access, retrieval time for subsequent accesses is reduced. 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, 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, see 15.2.6(5) Procedure for obtaining the size of the memory required when the snapshot method is used.

(4) Snapshot method applicability

Table 11-1 describes the applicability of the snapshot method for retrievals.

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: <ul style="list-style-type: none"> • <code>WITHOUT LOCK NOWAIT</code> specified for retrieval • <code>LOCK TABLE</code> required for retrieval 		N/R	No
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`).

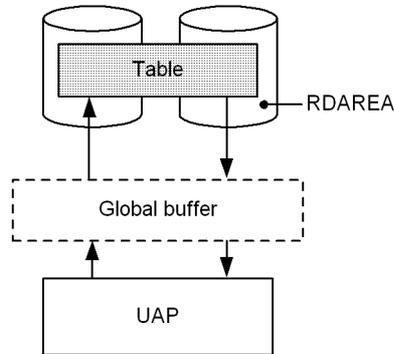
N/R: Not relevant or condition does not apply.

11.10 Global buffer pre-writing

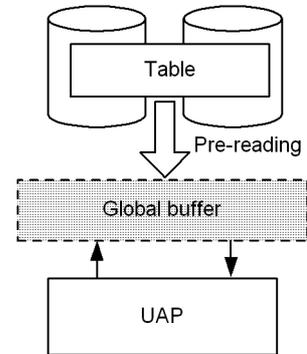
Global buffer pre-writing is a function for reading data from a specified table or index in advance and placing it in the global buffer. Figure 11-5 provides an overview.

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 (`pdpgbfont`).

(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 `pdpgbfont` serves no purpose when there are not enough global buffer sectors.
- When the global buffer residence utility (`pdpgbfont`) 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 8 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

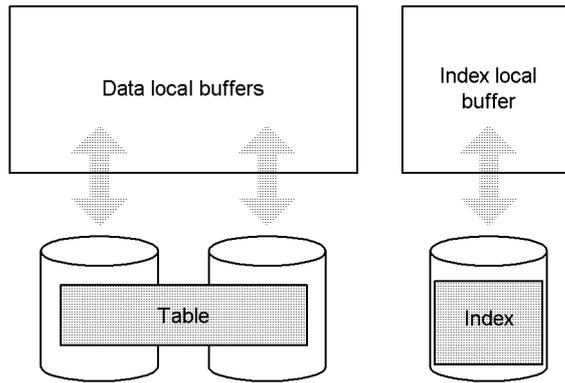
Because a UAP with an enduring connection to HiRDB has a large influence on the system (memory pressure, process contention, etc.), definition of local buffers in such a case is not appropriate.

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.

Figure 11-6 provides an overview of index local buffers where local buffers are allocated for both table data and index data.

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) terminates abnormally. If pages have been updated at the time the server process terminates abnormally, it may not be possible to recover with a rollback. 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 manual *HiRDB Version 8 System Operation Guide*.

Chapter

12. Designing Tables

This chapter explains items that should be examined during table design.

This chapter contains the following sections:

- 12.1 Items to be examined during table design
- 12.2 Normalizing a table
- 12.3 Table row partitioning
- 12.4 Table matrix partitioning
- 12.5 Defining a trigger
- 12.6 Creating a view table
- 12.7 Specifying the FIX attribute
- 12.8 Specifying a primary key
- 12.9 Specifying a cluster key
- 12.10 Specifying the suppress option
- 12.11 Specifying the no-split option
- 12.12 Specifying a binary data column
- 12.13 Specifying the WITHOUT ROLLBACK option
- 12.14 Specifying the falsification prevention facility
- 12.15 Table containing a repetition column
- 12.16 Table containing an abstract data type
- 12.17 Shared tables
- 12.18 Referential constraints
- 12.19 Check constraints

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. Table 12-1 lists the items that should be examined when designing a table.

Table 12-1: Items to be examined for database table design

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

Design task and items to be examined		Advantages	Disadvantages	Section
Table row partitioning	Specification of table row partitioning	<ul style="list-style-type: none"> • Operations can be performed in units of RDAREAs. • For HiRDB/Parallel Server, 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.	

Design task and items to be examined	Advantages	Disadvantages	Section
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 be combined with hash partitioning. 	<ul style="list-style-type: none"> • ALTER TABLE cannot be used to change table definitions. • Compared to normal row partitioning, operation and management become complex because this method allows more detailed partitioning of RDAREAs. 	12.4
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. • Foreign tables can be referenced or updated via their view tables. 	None	12.6

Design task and items to be examined	Advantages	Disadvantages	Section
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

12. Designing Tables

Design task and items to be examined	Advantages	Disadvantages	Section
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
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 WITHOUT ROLLBACK 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	12.13
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.	12.14
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	12.15

Design task and items to be examined	Advantages	Disadvantages	Section
Table with abstract data type	Data with a complicated structure can be stored in a table to process it as normal table data.	None	12.16
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.	12.17
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.	12.18
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.	12.19

* 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 manual *HiRDB Version 8 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. Figure 12-1 shows an example.

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.

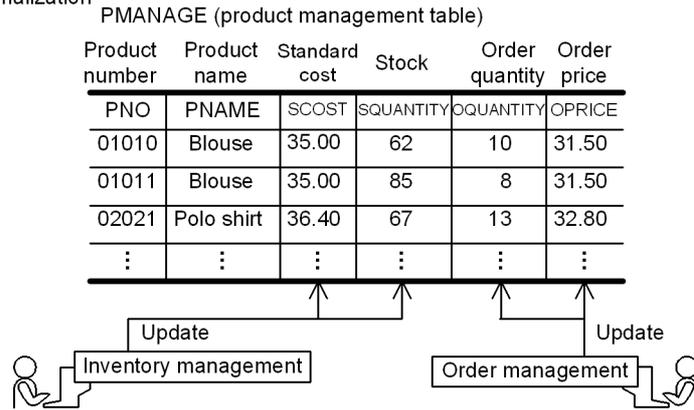
(2) Normalization for improved table processing efficiency

(a) Same table used by multiple applications

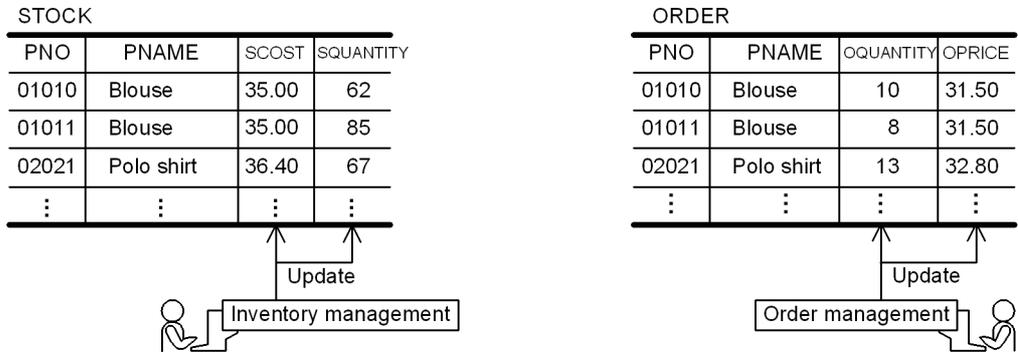
If the same table is used by multiple applications, normalization can result in a separate table for each application. This can improve the level of concurrent execution for each table. Figure 12-2 shows an example.

Figure 12-2: Same table used by multiple applications

- Before normalization



- After normalization



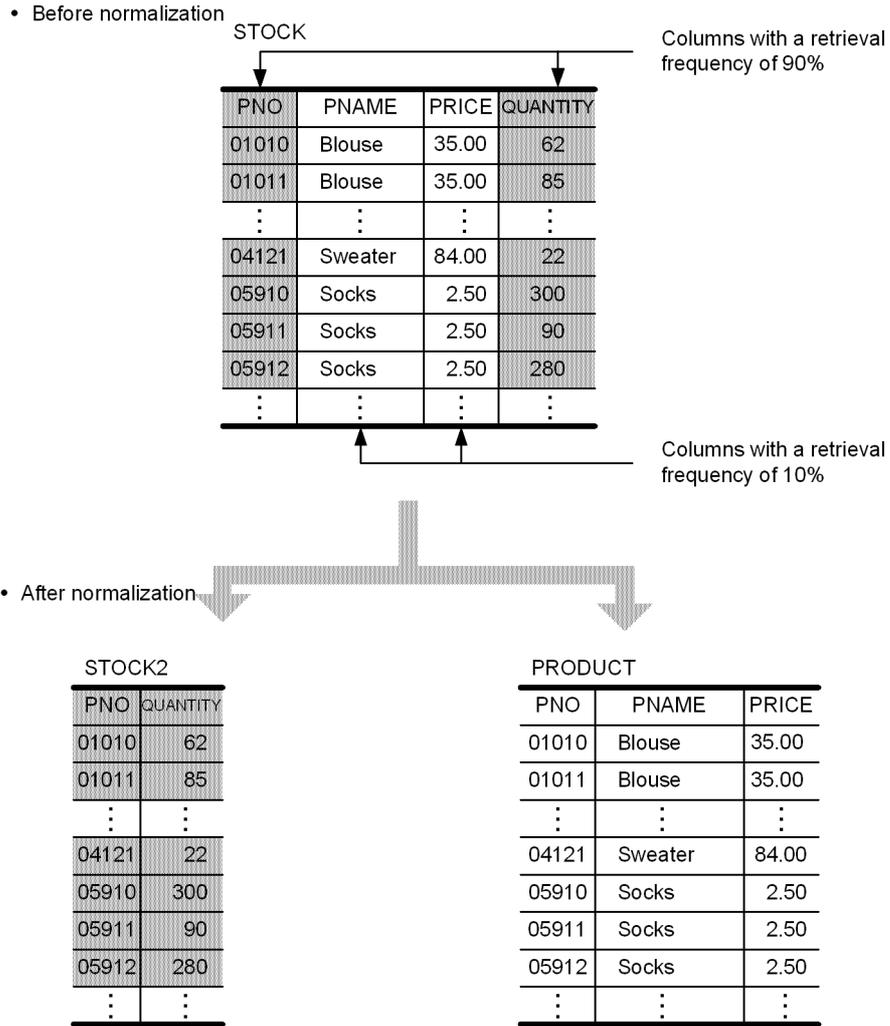
The `PMANAGE` (product management) table is used by the inventory management

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. Figure 12-3 shows an example.

Figure 12-3: Columns with different access frequencies



Taking the *STOCK* table shown in Figure 12-3, if the retrieval frequency ratio of the *PNO* and *SQUANTITY* 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

Twelve hash functions are available for hash partitioning, as listed in Table 12-2.

Table 12-2: Hash functions

Hash function	Explanation
HASH0	This hash function can be used with data types DATE, TIMESTAMP, CHAR (8) ^{#1} or CHAR (6) ^{#1} in the columns specified for partitioning. It uses year and month values to hash data and allocate, by circulating on a monthly basis, the RDAREAs in which the data is stored. You need to specify a single column for partitioning keys.
HASH1	This hash function can be used with all data types in the columns specified for partitioning. It hashes all bytes ^{#2} of the data from all columns specified for partitioning. This hash function can be specified for columns whose data length is 0 or more bytes.
HASH2	This hash function can be used with all data types in the columns specified for partitioning. It hashes all bytes ^{#2} of the data from all columns specified for partitioning. This hash function can be specified for columns whose data length is 0 or more bytes. You use this hash function when HASH1 is unable to store the data uniformly among the RDAREAs.
HASH3	Use this hash function when the data type of the columns to be partitioned is INTEGER or SMALLINT. It hashes the final 2 bytes ^{#2} of all partitioned columns. This hash function can be specified only for columns whose data length is at least 2 bytes.
HASH4	Use this hash function when the data type of the columns to be partitioned is DATE. It hashes the initial 4 bytes ^{#2} of all partitioned columns. This hash function can be specified only for columns whose data length is at least 4 bytes.
HASH5	Use this hash function when the data type of the columns to be partitioned is TIME. It hashes the initial 3 bytes ^{#2} of all partitioned columns. This hash function can be specified only for columns whose data length is at least 3 bytes.
HASH6	This hash function can be used with all data types in the columns specified for partitioning. The most appropriate type is DECIMAL. It hashes all bytes ^{#2} of the data from all columns specified for partitioning. This hash function can be specified for columns whose data length is 0 or more bytes.
HASHA	This hash function can be used with all data types in the columns specified for partitioning. It hashes all bytes ^{#2} of the data from all columns specified for partitioning. This hash function can be specified for columns whose data length is 0 or more bytes.
HASHB	This hash function can be used with all data types in the columns specified for partitioning. It hashes all bytes ^{#2} of the data from all columns specified for partitioning. This hash function can be specified for columns whose data length is 0 or more bytes. You use this hash function when HASHA is unable to store the data uniformly among the RDAREAs.

Hash function	Explanation
HASHC	Use this hash function when the data type of the columns to be partitioned is <code>INTEGER</code> or <code>SMALLINT</code> . It hashes the final 2 bytes ^{#2} of all partitioned columns. This hash function can be specified only for columns whose data length is at least 2 bytes.
HASHD	Use this hash function when the data type of the columns to be partitioned is <code>DATE</code> . It hashes the initial 4 bytes ^{#1} of all partitioned columns. This hash function can be specified only for columns whose data length is at least 4 bytes.
HASHE	Use this hash function when the data type of the columns to be partitioned is <code>TIME</code> . It hashes the initial 3 bytes ^{#2} of all partitioned columns. This hash function can be specified only for columns whose data length is at least 3 bytes.
HASHF	This hash function can be used with all data types in the columns specified for partitioning. The most appropriate type is <code>DECIMAL</code> . It hashes all bytes ^{#2} of all data of all partitioned columns. This hash function can be specified for columns whose data length is 0 or more bytes.

Note 1

- If the table is not a rebalancing table, specify one of `HASH0`-`HASH6`. `HASH6` provides the most uniform hashing result, so normally you should 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 one of `HASHA`-`HASHF`. `HASHF` provides the most uniform hashing result, so normally you should 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.

Note 2

A LOB column cannot be specified as the partitioning key.

#1: When the data type is `CHAR(8)` or `CHAR(6)`, use the following format:

- For `CHAR(8)`: 'YYYYMMDD'
 - For `CHAR(6)`: 'YYYYMM'
- YYYY: 0001 to 9999 (Year)
MM: 01 to 12 (month)
DD: 01 to 31 (date)

#2: For the `VARCHAR`, `MVARCHAR`, and `NVARCHAR` types, hashing ignores any trailing spaces. For the `DECIMAL`, `INTERVAL YEAR TO DAY`, and `INTERVAL HOUR TO`

SECOND types, if the sign is F, it must be changed to C for hashing.

(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 (`pddbst`) 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.

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 manual *HiRDB Version 8 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

Table 12-3 shows the differences among key range partitioning, flexible hash partitioning, and FIX hash partitioning.

Table 12-3: Differences among 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.	Only back-end servers that may contain the applicable data according to the search condition are subject to retrieval processing (= and IN predicates only). 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.

Difference	Key range partitioning	Flexible hash partitioning	FIX hash partitioning
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 manual *HiRDB Version 8 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.
 - The same RDAREA cannot be specified more than once.

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)

Figure 12-4 shows an example of key range partitioning (with storage condition specified).

Figure 12-4: Example of key range partitioning with 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

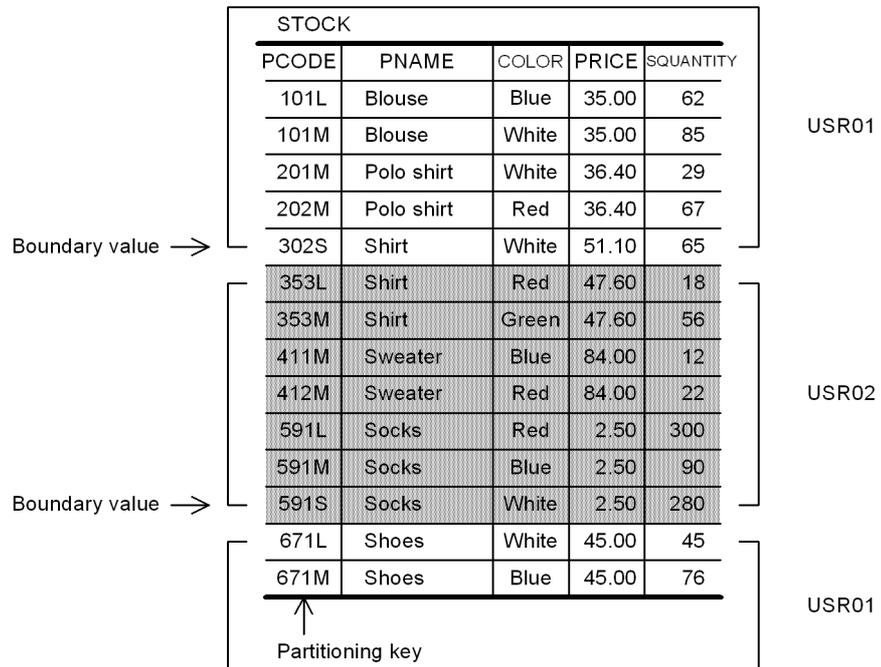
Explanation:

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)

Figure 12-5 shows an example of key range partitioning (with boundary value specified).

Figure 12-5: 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

Figure 12-6 shows an example of flexible hash partitioning and FIX hash partitioning.

Figure 12-6: 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.

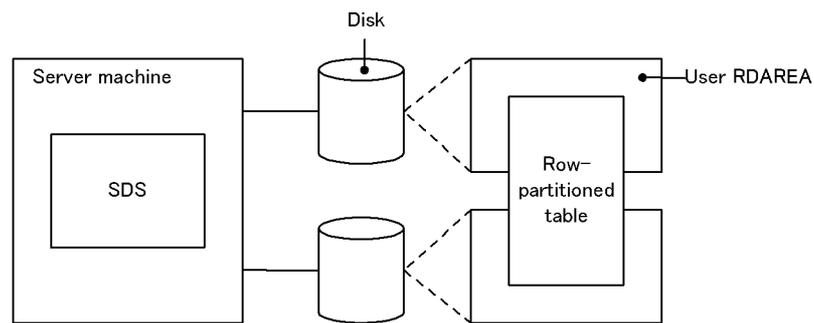
12.3.3 Forms of table row partitioning

Following are the basic forms of table row partitioning:

- Row partitioning within a server (applicable to a HiRDB/Single Server)
- Row partitioning among servers (applicable to a HiRDB/Parallel Server)

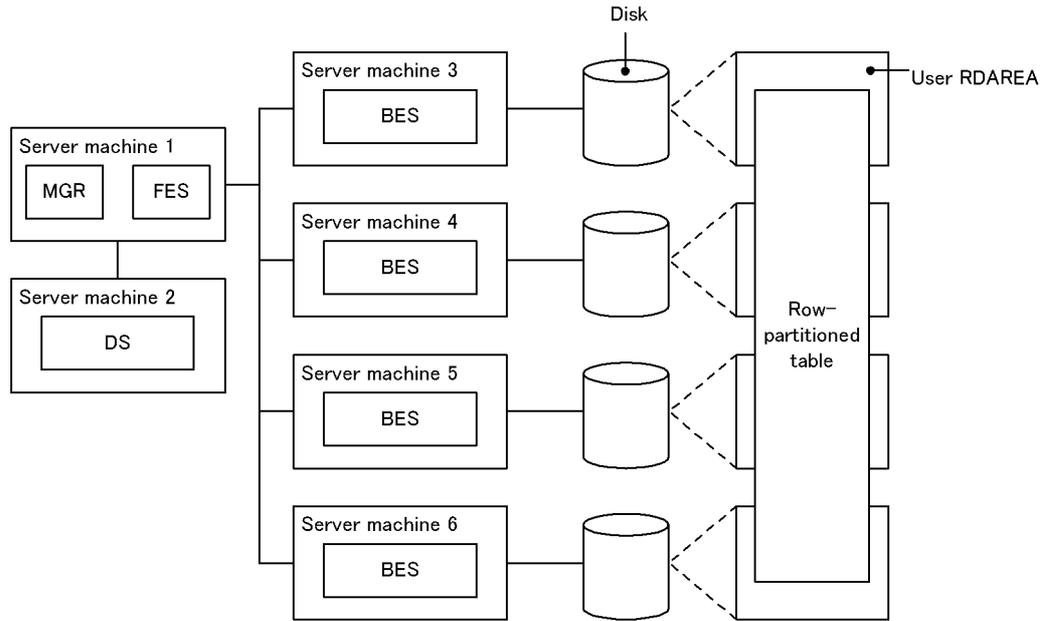
Figures 12-7 and 12-8 show these forms.

Figure 12-7: Table row partitioning form for a HiRDB/Single Server



SDS: Single server

Figure 12-8: Table row partitioning form for a HiRDB/Parallel Server



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

Improved operability

Data storage in table, table reorganization, making backups, etc., 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 manual *HiRDB Version 8 UAP Development Guide*.

(2) HiRDB/Parallel Server**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.

12.3.5 Design considerations**(1) Considerations common to both HiRDB/Single Server and HiRDB/Parallel Server**

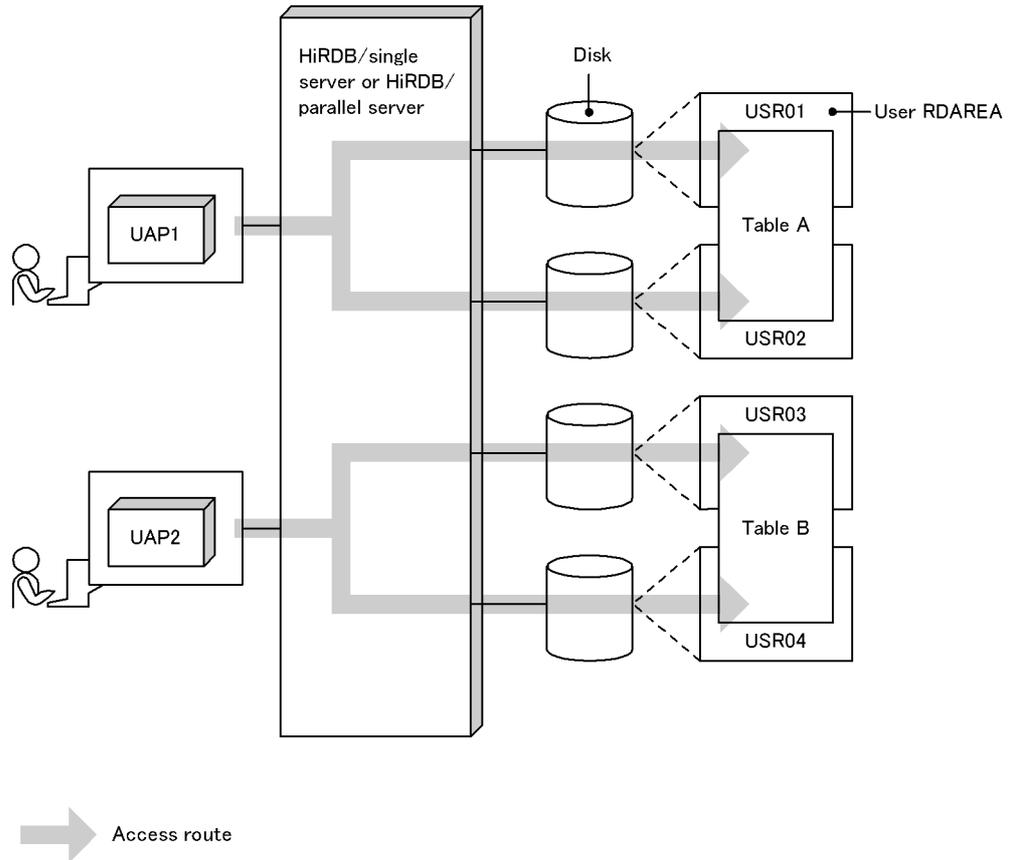
Following are the design considerations that are common to HiRDB/Single Servers and HiRDB/Parallel Servers:

(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.

Figure 12-9 provides an overview of row partitioning taking into account contention among disk accesses.

Figure 12-9: Overview of row partitioning taking into account contention among disk accesses



Explanation:

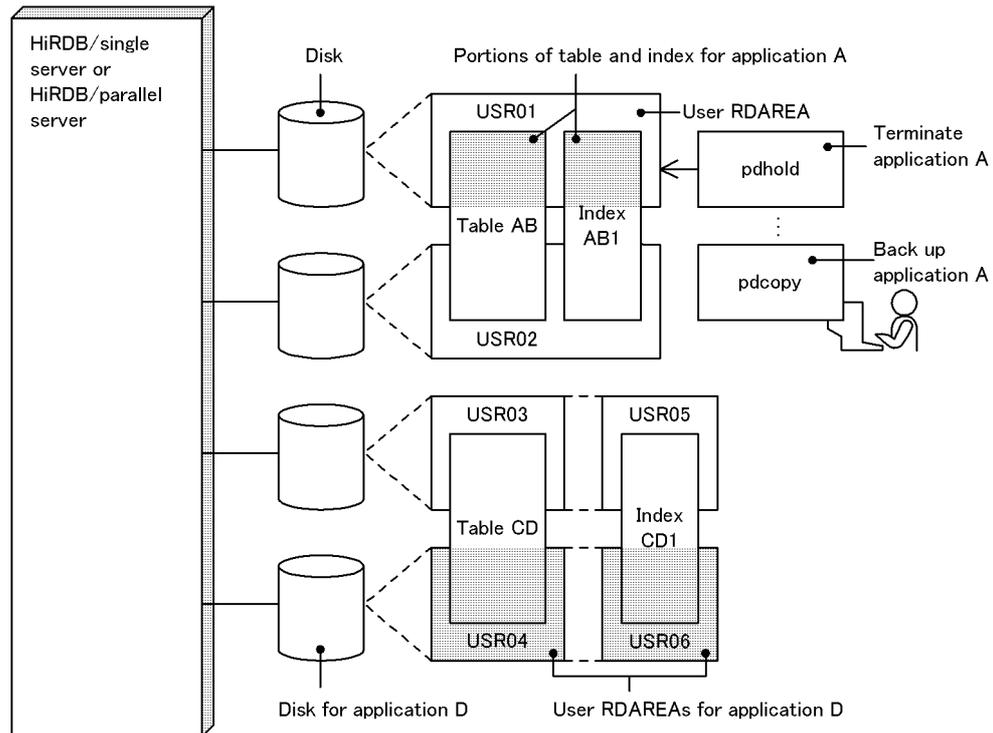
Tables A and B are partitioned and stored in two sets of user RDAREAs, USR01-USR02 and USR03-USR04, which are on separate disks. If UAP1 and UAP2 attempt to access tables A and B concurrently, no contention occurs, thereby reducing their wait time.

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

Figure 12-10 provides an overview of row partitioning taking into account operability.

Figure 12-10: 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, etc., 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 12-10, 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 12-10, 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

Following are the design considerations for HiRDB/Parallel Servers:

(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*

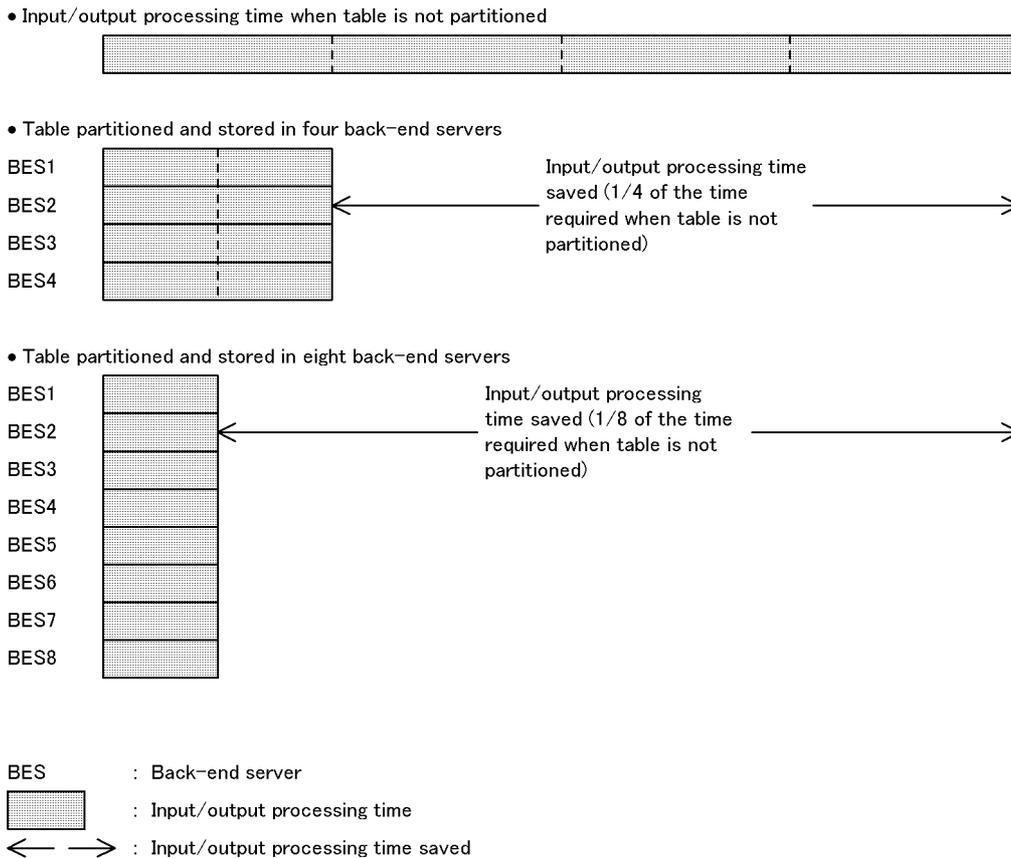
If a table stored in a user RDAREA at a back-end server in a server machine with a low CPU workload is accessed mostly for input/output operations, the workload of disk accesses is not uniform among the multiple servers, thereby affecting adversely the efficiency of parallel processing.

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. Figure 12-11 provides an overview of input/output processing performance based on the number of back-end servers used for table row partitioning.

Figure 12-11: Overview of input/output processing performance based on number of back-end servers used for table row partitioning



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.

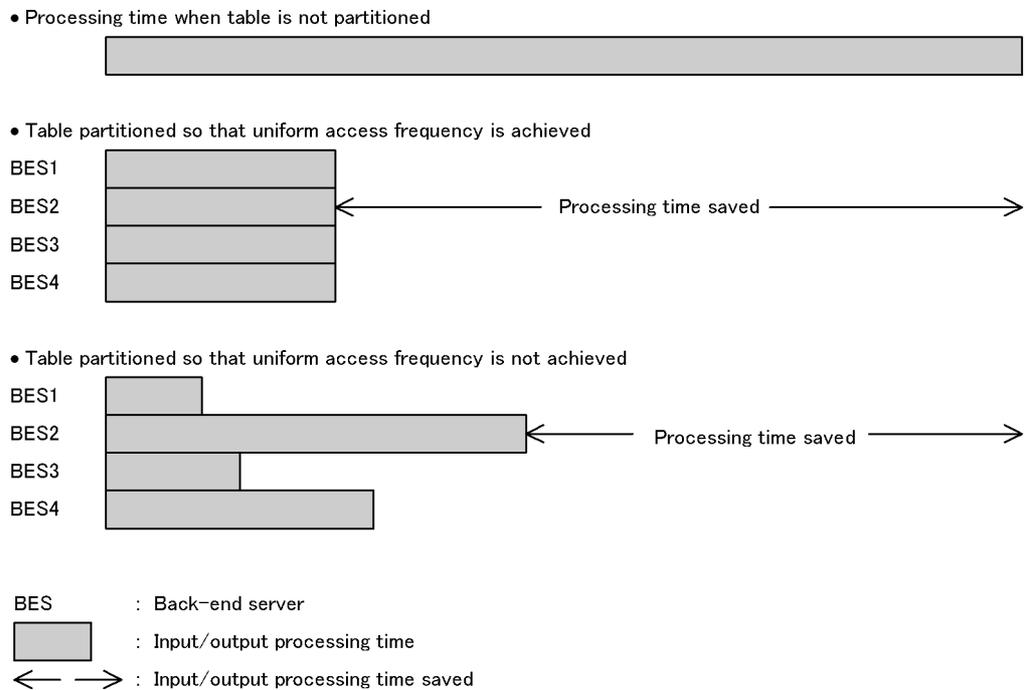
Flexible hash partitioning or FIX hash partitioning

- 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.

Figure 12-12 shows the performance of parallel processing depending on the table access frequency.

Figure 12-12: Performance of parallel processing depending on table access frequency



Explanation:

The processing time that can be saved depends on whether or not 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.

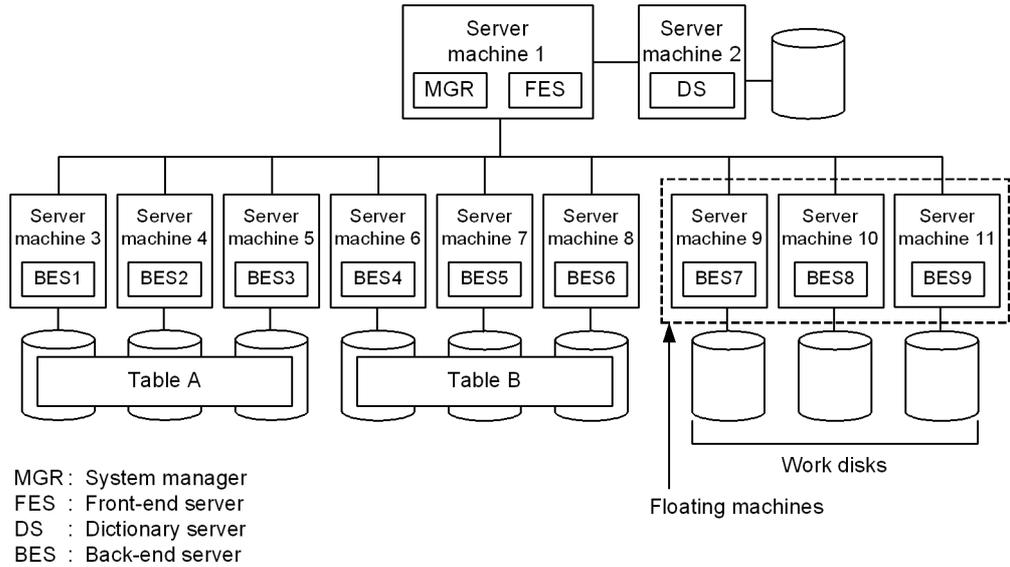
A floating machine is a server machine where a floating server is installed, which is a back-end server dedicated to complicated retrieval processing such as join processing. User RDAREAs cannot be allocated to a back-end server defined as a floating server.
4. Determining the number of work disks

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.

Figure 12-13 shows the concept of a system configuration involving table row partitioning taking into account complicated retrieval processing.

Figure 12-13: System configuration involving table row partitioning taking into account complicated 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 manual *HiRDB Version 8 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 Partitioning Option 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:

- Partitioning by the first dimension partitioning column results in a vast amount of data within each set of 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:

- Partitioning by the first dimension partitioning column results in a vast amount of data within each set of boundary values.
- You wish to uniformly segment the range of data that was partitioned by the first dimension partitioning column.

(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.
 - You can make multiple specifications for a single RDAREa, but you cannot make all specified RDAREAs into one RDAREa.
- 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.
 - The list of matrix partitioning RDAREAs cannot contain the same RDAREa more than once (you cannot specify the same RDAREa for multiple RDAREAs that store the range of data partitioned by the first dimension partitioning column).

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.

Figure 12-14 provides an example of matrix partitioning.

Figure 12-14: Matrix partitioning example (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 Second
 partitioning partitioning
 key 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 (number of boundary values + 1) × (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.

Figure 12-15 shows an example of matrix partitioning.

Figure 12-15: 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

12. Designing Tables

- 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

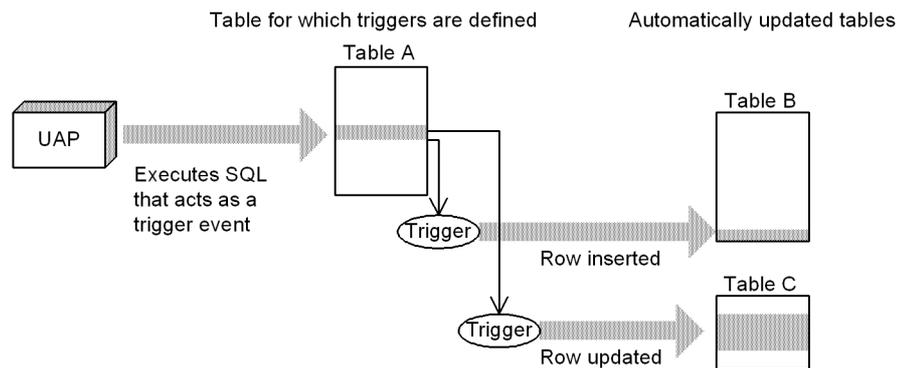
- Matrix partitioning table stored in USR08

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. Figure 12-16 provides an overview of triggers.

Figure 12-16: 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 8 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.

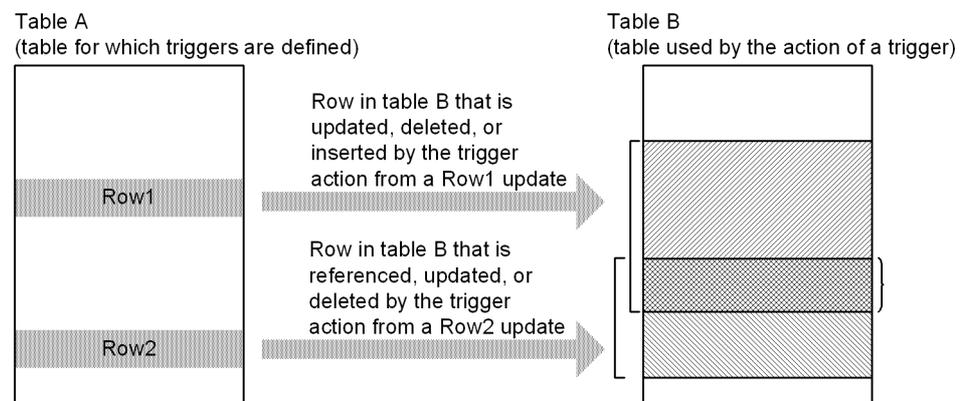
```

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.

- 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.

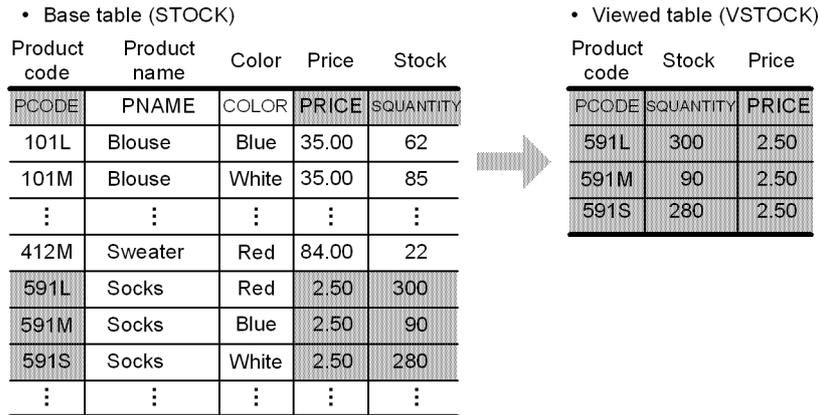
12.6 Creating a view table

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. The HiRDB External Data Access facility can be used to create view tables from foreign tables.

(1) Relationship between base tables and view tables

Figure 12-17 shows the relationship between a base table and a view table.

Figure 12-17: Relationship between base table and 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.

■ HiRDB External Data Access facility

You can create view tables from foreign tables in the same manner as from base tables. You can also refresh view tables created from foreign tables.

(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 and foreign 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 and foreign tables
- View tables that combine a maximum of 15 base tables or foreign tables
- View tables based on the result of grouping retrieval
- View tables based on base tables or foreign tables that are owned by other users (limited to base tables or foreign tables 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 or foreign tables owned

by another user holds the same privileges that he or she holds for those base tables or foreign tables. However, if the view table definition has any of the following definitions, only row retrieval is allowed, regardless of whether or not the security facility is used:

- View tables for which the columns contain multiple specifications of the same columns from base tables or foreign tables
- View tables for which the columns contain the results of literals, `USER`, `CURRENT_DATE`, `CURRENT_TIME`, arithmetic operations, date operations, time operations, concatenation operations, or scalar functions
- Multiple base tables or foreign 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*

The FIX attribute should be specified during table definition when all the following conditions are satisfied:

- No columns will be added to the table
- There will be no null values in any column
- No column is variable length

When these conditions are not satisfied, the following should be evaluated:

- 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 (i.e., CREATE FIX TABLE is specified).

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.

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 or not 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 (`pdreorg`) to determine whether or not 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.

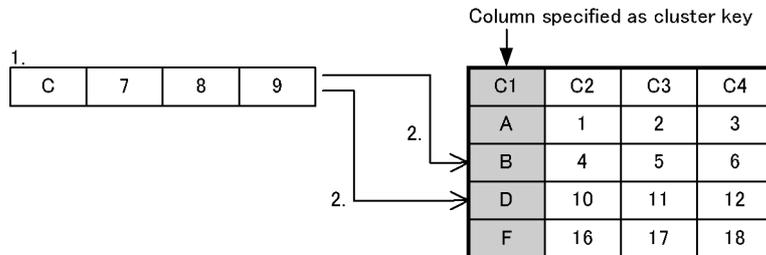
(4) Design considerations

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. Figure 12-18 provides an overview of such a situation.

Figure 12-18: 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, etc.

(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*

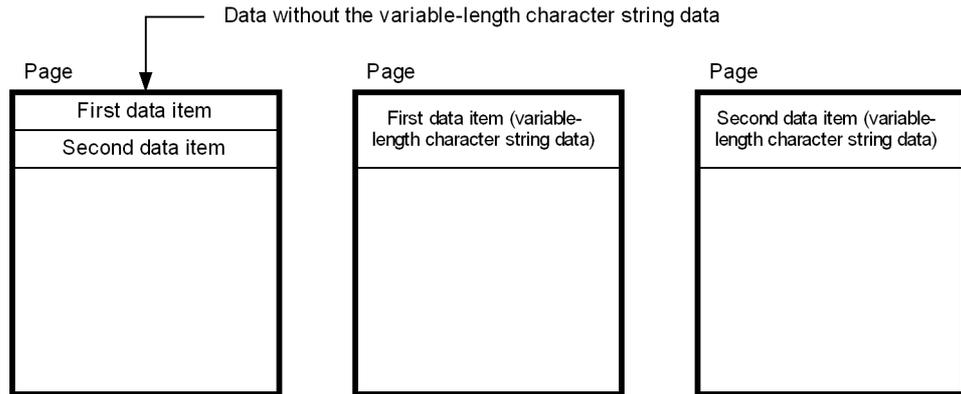
- If the number of significant digits in decimal data equals the defined length or equals 1, the data is stored with a length of *defined length* + 1. In this case, the length of the stored data is greater than when the suppression option is not specified.
- The suppression option cannot be specified for a table with the `FIX` attribute.

12.11 Specifying the no-split option

If any of the following data types 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. Figure 12-19 shows this data storage method.

- VARCHAR
- MVARCHAR
- NVARCHAR

Figure 12-19: Data storage method when actual variable-length character string data is 256 bytes or greater



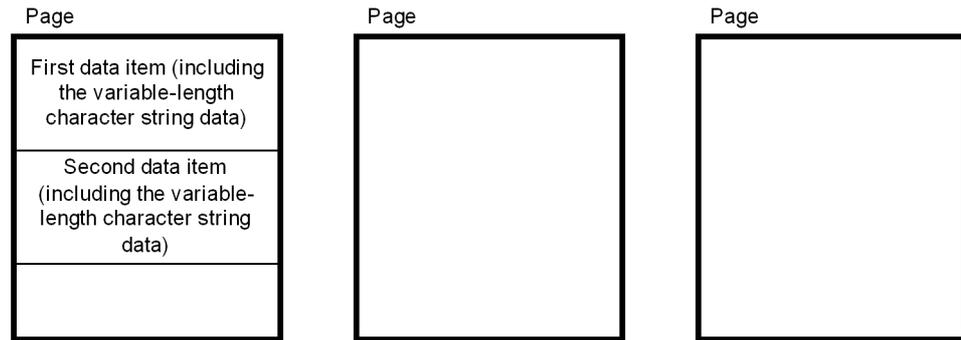
Explanation:

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. Figure 12-20 shows the data storage method when the no-split option is specified.

Figure 12-20: Data storage method 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.

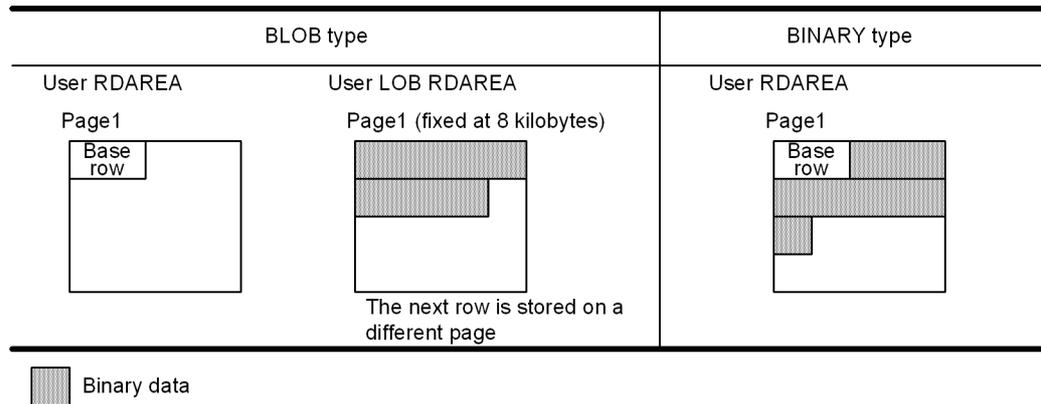
12.12 Specifying a binary data column

There are two data types for defining columns that store documents, images, audio, and other variable-length binary data:

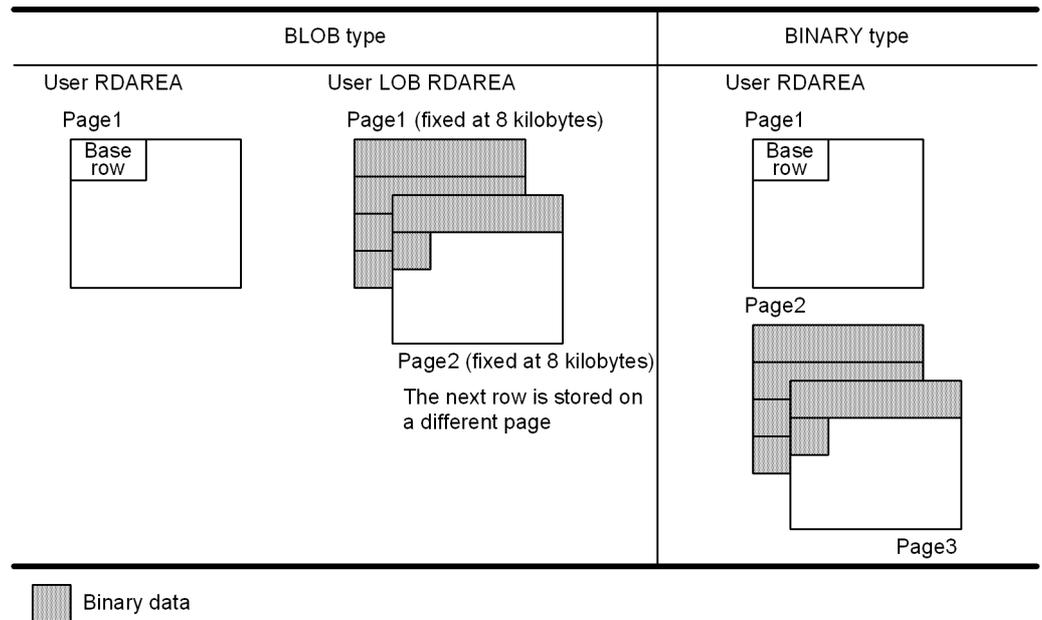
- BLOB type (columns specified as the BLOB type are called LOB columns)
- BINARY type

A comparison of how these types store data follows.

Data length (base row + binary data length) does not exceed 1 page



Data length (base row + binary data length) does exceed 1 page

**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 a dedicated `BINARY` segment.

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 dedicated BINARY segments, which are different from the segments that store the base rows. When binary data exceeds one page, even if the database analysis utility (`pddbst`) determines that there are unused pages, if there are no unused pages in the dedicated BINARY segment there may be insufficient space in the RDAREA.

12.12.3 BLOB type and BINARY type usage

Table 12-4 lists the recommended data types for each usage of binary data.

Table 12-4: 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 8 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.

Binary data usage	Average size of recommended data type		Explanation
	32,000 bytes or less	More than 32,000 bytes	
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 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, document numbers, etc., 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. Figure 12-21 shows an example of a numbering application.

Figure 12-21: 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).

 : 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 12-21:

```
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.

12.14 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. Table 12-5 lists the operations that are permitted on falsification prevented tables.

Table 12-5: Operations permitted on falsification prevented tables

Operation	Falsification prevented table	
	Deletion prevented duration specified	Deletion prevented duration not specified
Insert (<code>INSERT</code>)	Yes	Yes
Retrieve (<code>SELECT</code>)	Yes	Yes
Update by column (<code>UPDATE</code>)	Yes ¹	Yes ¹
Update by row (<code>UPDATE</code>)	No	No
Delete (<code>DELETE</code>)	Yes ²	No
Delete all rows (<code>PURGE TABLE</code>)	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.

12.14.1 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 *12.14.2(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 *12.14.2 Restrictions*.

12.14.2 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. Table 12-6 lists these definition SQL statements and their restrictions.

Table 12-6: Restricted definition SQL statements and the restrictions

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>12.14.3 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. Table 12-7 lists the restricted utilities and their restrictions. There are no restrictions on utilities not listed in the table.

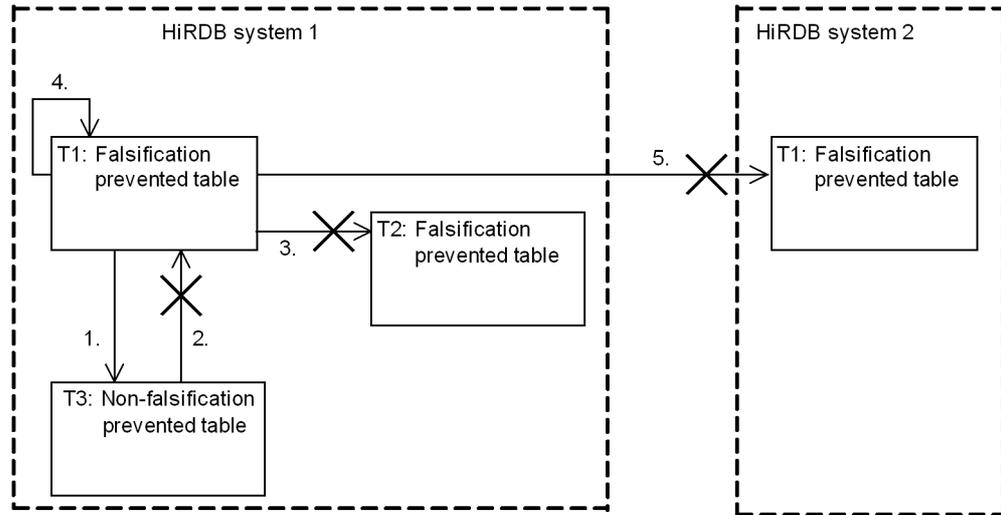
Table 12-7: Restricted utilities and the restrictions

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>).
Database reorganization utility (<code>pdreorg</code>)	<p>Table reorganization (<code>-k reorg</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 12-22). <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>pdrrbal</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 8 Command Reference*.

Figure 12-22: 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. Table 12-8 lists the restricted commands.

Table 12-8: Restricted commands and their restrictions

Operation command	Restrictions
RDAREA shutdown (<code>pdhold</code>)	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: <ul style="list-style-type: none"> Reference-possible shutdown: <code>-i</code> Backup shutdown: <code>-b</code>
RDAREA shutdown release (<code>pdrels</code>)	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.

(4) Restrictions on related products

Restrictions on related products include the following:

- 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.

12.14.3 Changing a falsification-unprevented table to a falsification prevented table

If a table contains no data, you can use the `ALTER TABLE` definition SQL statement to change its definition to a falsification prevented table. If the table contains data, `ALTER TABLE` cannot be used to change it to a falsification prevented table. In such a case, you must first unload the data, change the table to a falsification prevented table, and then load the data back into the table. The following example describes how to change an existing falsification-unprevented table containing data to a falsification prevented table:

To change a falsification-unprevented table to a falsification prevented table:

This example changes the existing falsification-unprevented table `T1` that is stored in RDAREA `RDAREA01` to a falsification prevented table.

- Use the `pdhold` command to shut down the RDAREA that contains the falsification-unprevented table.

```
pdhold -r RDAREA01 -c
```

- Use the database reorganization utility (`pdreorg`) 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`).

```
pdrorg -k unld -t T1 -w bin control_file
```

3. Use `PURGE TABLE` to delete all data from the falsification-unprevented table.

```
PURGE TABLE T1
```

4. If you want to make some columns updatable, use `ALTER TABLE` to change their column attribute to updatable column.

```
ALTER TABLE T1 CHANGE COLUMNA UPDATE
```

5. Use `ALTER TABLE` with the falsification prevention option specified to change the table to a falsification prevented table.

```
ALTER TABLE T1 CHANGE INSERT ONLY WHILE 10 YEARS BY COLUMNB
```

6. Use the database load utility (`pdload`) to load the data that was unloaded in step 2.

```
pdload -b -w T1 control_file
```

7. Because the database load utility (`pdload`) was executed, use the database copy utility (`pdcopy`) to make a backup.

```
pdcopy -m \hirdb\rdarea\mast\mast01 -M r -p \usr\hirdb\pdcopy\list\list01  
-b \usr\hirdb\pdcopy\backup\backup01 -r RDAREA01
```

8. Use the `pdrels` command to release the RDAREA from shutdown status.

```
pdrels -r RDAREA01 -o
```

To determine the timing of setting the falsification prevention option, check the values in the `SQL_TABLES` data dictionary table. Table 12-9 lists and describes the values in the `SQL_TABLES` table.

Table 12-9: 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

12.14.4 Error operation

Because RDAREAs that store falsification prevented tables cannot be reinitialized (`initialize rdarea`), these RDAREAs cannot be recovered using reinitialization recovery. Recovery must be with the database recovery utility (`pdrrstr`). If the RDAREA is full, expand it with the `expand rdarea` statement.

12.15 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 Figure 12-23. Figure 12-23 shows an example of tables defined without repetition columns.

Figure 12-23: 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.

Figure 12-24 shows an example of a table containing repetition columns.

Figure 12-24: 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 12-24. 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:
 - CHAR or MCHAR with a defined length of 256 bytes or greater
 - NCHAR with a defined length of 128 characters or greater
 - 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.

12.16 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.

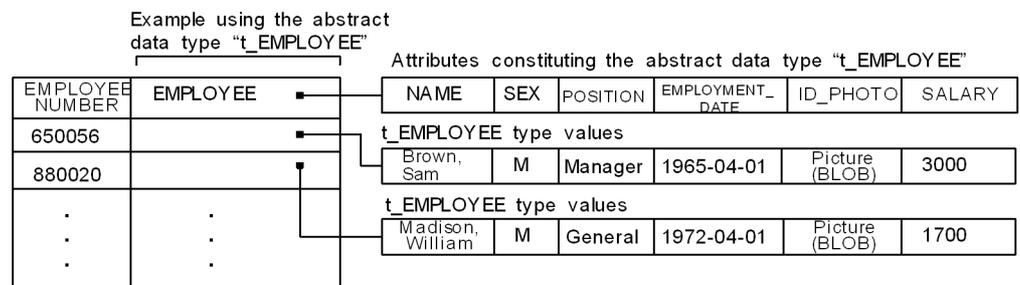
An abstract data type provides a structure that enables complicated data that cannot be handled at all by existing data types to be handled easily. HiRDB allows users to define such a data type as an abstract data type. Creating an abstract data type involves using the definition SQL to define attributes 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.

Figure 12-25 shows the data structure of a table containing an abstract data type. In this figure, the EMPLOYEE column of the STAFF_TABLE is set to abstract data type t_EMPLOYEE.

Figure 12-25: 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

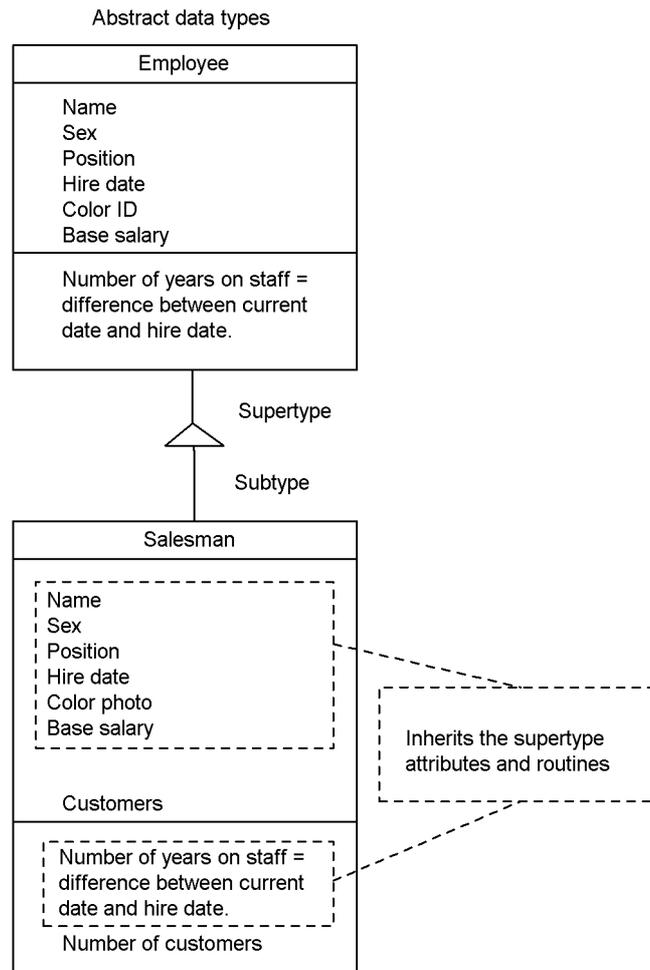
(a) Inheritance

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.

Figure 12-26 shows a hierarchical structure based on the relationship between a supertype abstract data type and a subtype. In this figure, the subtype `OPERATOR` is derived from the abstract data type `EMPLOYEE`.

Figure 12-26: Hierarchical structure based on the relationship between supertype and subtype abstract data types

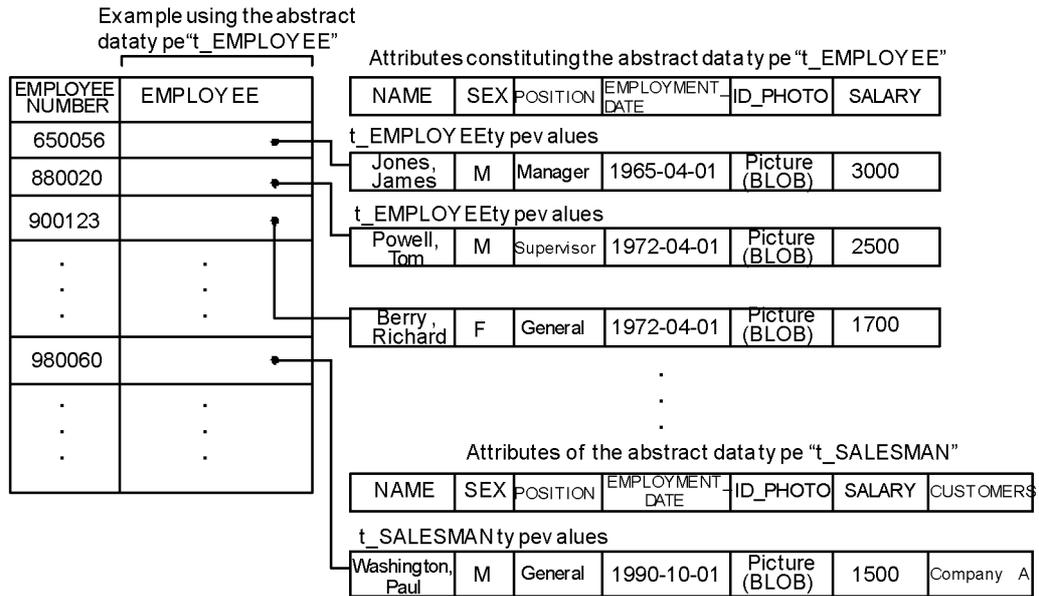


(b) Substitutability

The values of a subtype can be treated as values of its supertype. This is called substitutability. Figure 12-27 shows the data structure of a table containing an abstract data type in which values are inserted using substitutability.

Figure 12-27: Data structure of a table containing an abstract data type (using substitutability)

- Staff table, including 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. Table 12-10 shows the encapsulation levels and privileges.

Table 12-10: 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	—	—

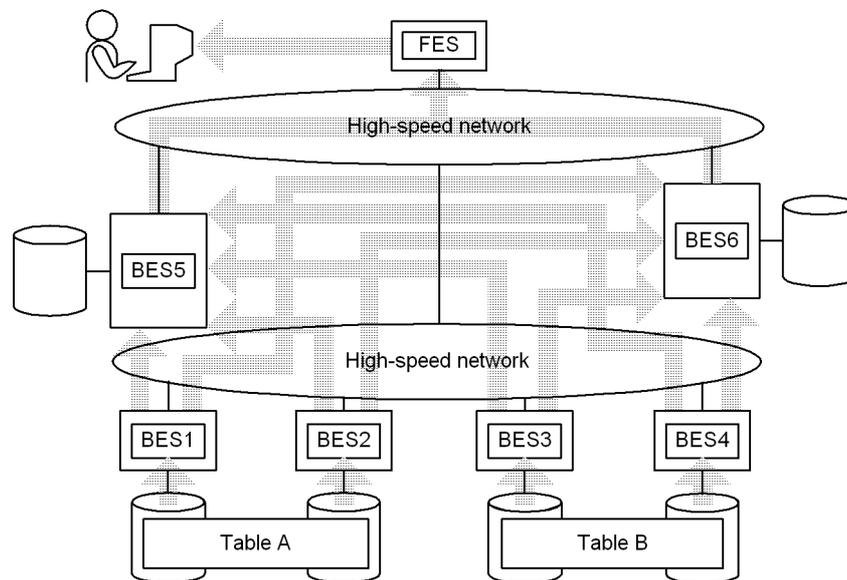
P: Accesses to attribute values and use of routines are permitted.

— : Accesses to attribute values and use of routines are not permitted (if violated, an SQL error results).

12.17 Shared tables

In the case of a HiRDB/Parallel Server, 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 12.17.3 *Manipulating shared tables*. Figure 12-28 shows join processing without using a shared table, and Figure 12-29 shows join processing using a shared table.

Figure 12-28: Join processing without using a shared table



Explanation:

This example joins tables A and B.

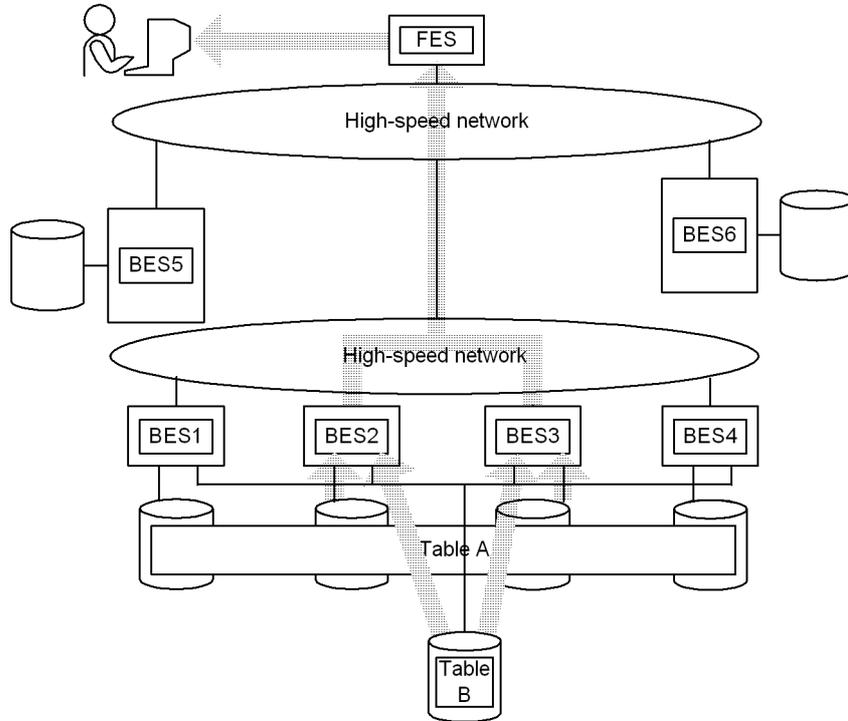
BES1, BES2: Retrieve data from table A and transfer it to BES5 and BES6 for matching.

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 12-29: 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. This provides SQL and UAP compatibility with a HiRDB/Parallel Server. Shared tables and

shared indexes are usually used with a HiRDB/Parallel Server because they are especially effective in HiRDB/Parallel Servers. The following subsections describe the use of shared tables with a HiRDB/Parallel Server. For details about using shared tables with a HiRDB/Single Server, see *12.17.6 Using shared tables with a HiRDB/Single Server*.

12.17.1 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.

12.17.2 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 a non-partitioned `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.

12.17.3 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.

(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 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.

12.17.4 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.

12.17.5 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 reusage facility 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`)

For details about the utilities and operation commands that cannot be executed on shared RDAREAs, see *14.6(7) Notes about using shared RDAREAs*.

12.17.6 Using shared tables with a HiRDB/Single Server

This subsection describes for a HiRDB/Single Server the differences from using shared tables with a HiRDB/Parallel Server.

About notes

The notes about using shared tables with a HiRDB/Single Server (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. The principal difference is that with a HiRDB/Single Server, deadlock between servers does not occur because there is only one server. Additionally, the notes about execution of operation commands for a HiRDB/Parallel Server do not apply to a HiRDB/Single Server.

About the RDAREAs for storing shared tables and shared indexes

In the case of a HiRDB/Single Server, 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 without issuing a LOCK statement and the server process terminates abnormally, HiRDB does not terminate abnormally with abort code Phb3008.

About migrating from HiRDB/Single Server to HiRDB/Parallel Server

If you are migrating from a HiRDB/Single Server to a HiRDB/Parallel Server, 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 system. The migration procedure is as follows:

1. Check the HiRDB/Single Server 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.

12. Designing Tables

3. Use the database structure modification utility (`pdmod`) to migrate the HiRDB/Single Server to a HiRDB/Parallel Server.
4. Define shared RDAREAs in the HiRDB/Parallel Server, check and, if necessary, revise the shared tables and shared indexes, then store them in shared RDAREAs.

12.18 Referential constraints

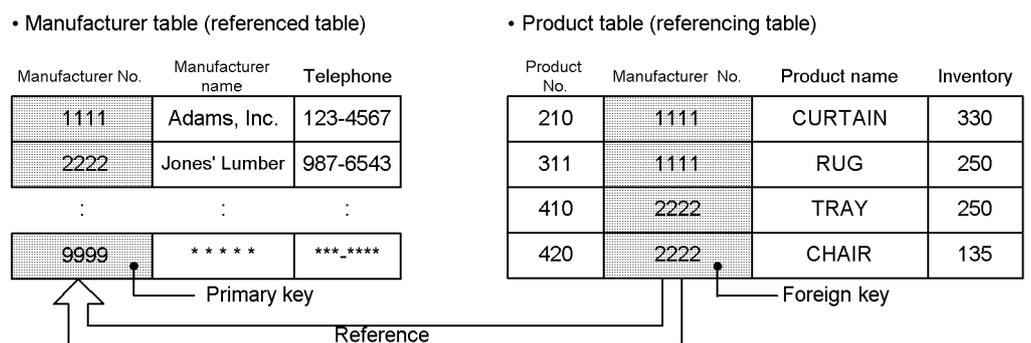
12.18.1 About referential constraints

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 *12.18.3 Check pending status*. For details about operations that cause loss of guaranteed data integrity, see *12.18.4 Data manipulation and integrity*.

Figure 12-30 shows an example of a referenced table 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 12-30: 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, if the referenced and referencing tables are updated, the processing time required for checking increases because data integrity is also checked.

12.18.2 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.

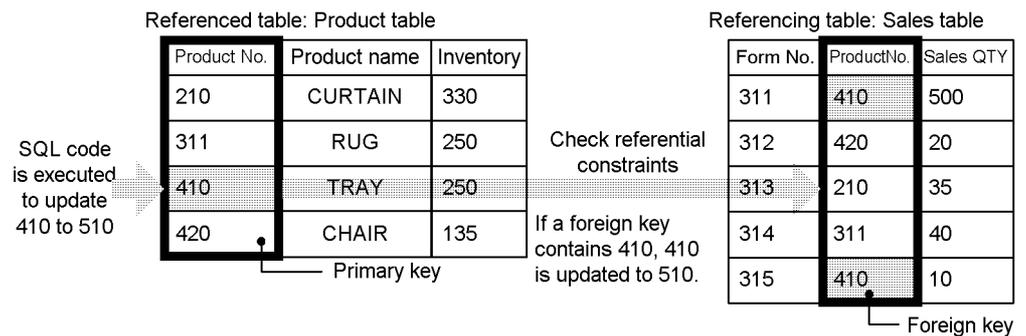
The following subsections explain the actions in the referenced and referencing tables when `CASCADE` or `RESTRICT` is specified.

(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. In this case, a check is performed to determine if the updated primary key value is the same as a value in a foreign key. The related value in the referencing foreign key is not changed if such a change would result in a referential constraint violation.

Figures 12-31 and 12-32 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.

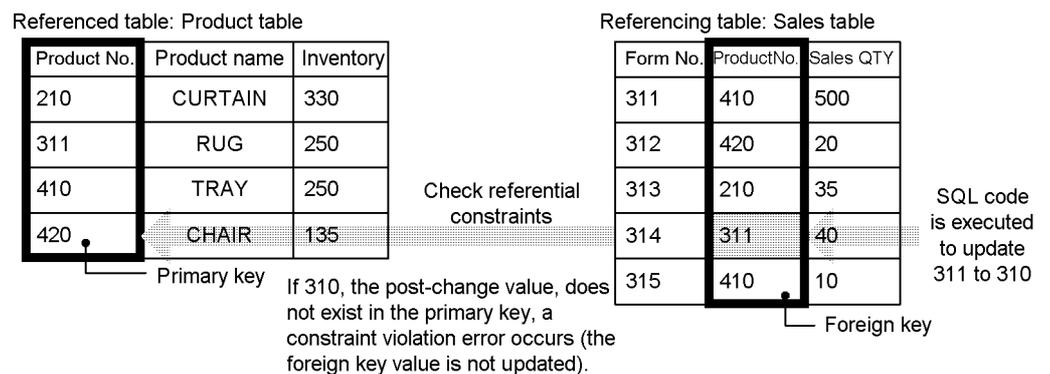
Figure 12-31: Example of the actions that occur when update SQL code is executed on a referenced table (with CASCADE specified)



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 12-32: 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 12-11 lists primary key operations and describes the resulting actions that occur in the referencing table when `CASCADE` is specified. Table 12-12 lists foreign key operations and describes the resulting actions that occur in the referenced table when `CASCADE` is specified.

Table 12-11: 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 (<code>INSERT</code> statement)	None	Y	None
Update (<code>UPDATE</code> statement), delete (<code>DELETE</code> 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 12-12: 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 (<code>INSERT</code> 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 (<code>UPDATE</code> statement)	The referenced table has a value in its primary key that is the same as the updated foreign key value.		Y	None

Foreign key manipulation	Relationship between rows in referenced and referencing tables		Result of foreign key operation	Action in referenced table
	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.

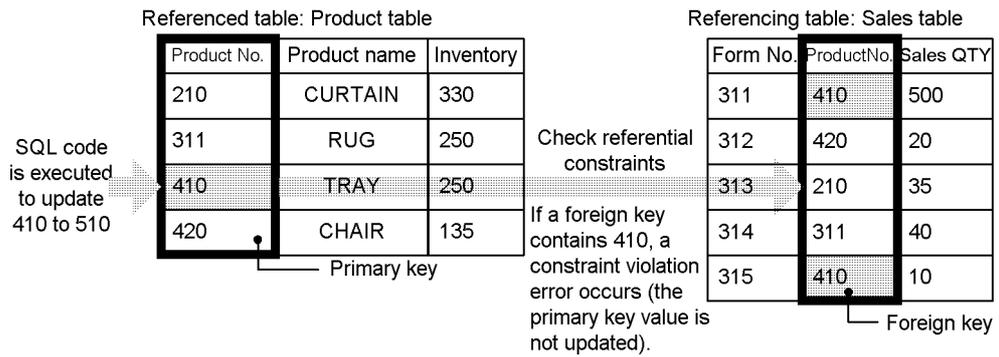
Note that when you specify `CASCADE`, HiRDB internally generates a trigger during table definition to update the foreign key value with the change made in the primary key value. For details about triggers for referential constraint actions and about user-defined triggers, see *12.18.6 Referential constraints 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.

Figure 12-33 shows an example of the actions that occur if `RESTRICT` is specified when SQL code is executed on a referenced table. The actions in a referencing table are the same as those when `CASCADE` is specified (see Figure 12-32).

Figure 12-33: 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.

Table 12-13 lists primary key operations and describes the resulting actions that occur in the referenced table when RESTRICT is specified. 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 12-12).

Table 12-13: 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)**

- In a single table, a foreign key cannot be defined for a column for which a primary key is defined.
- 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.

(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 8 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.

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 manual *HiRDB Version 8 System Operation Guide*.

(5) Referential constraint definition examples

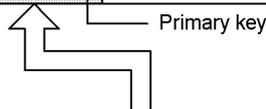
The following section provides examples of how to define referential constraints.

(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.

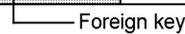
- Manufacturer table (MANUFACTURER)

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)

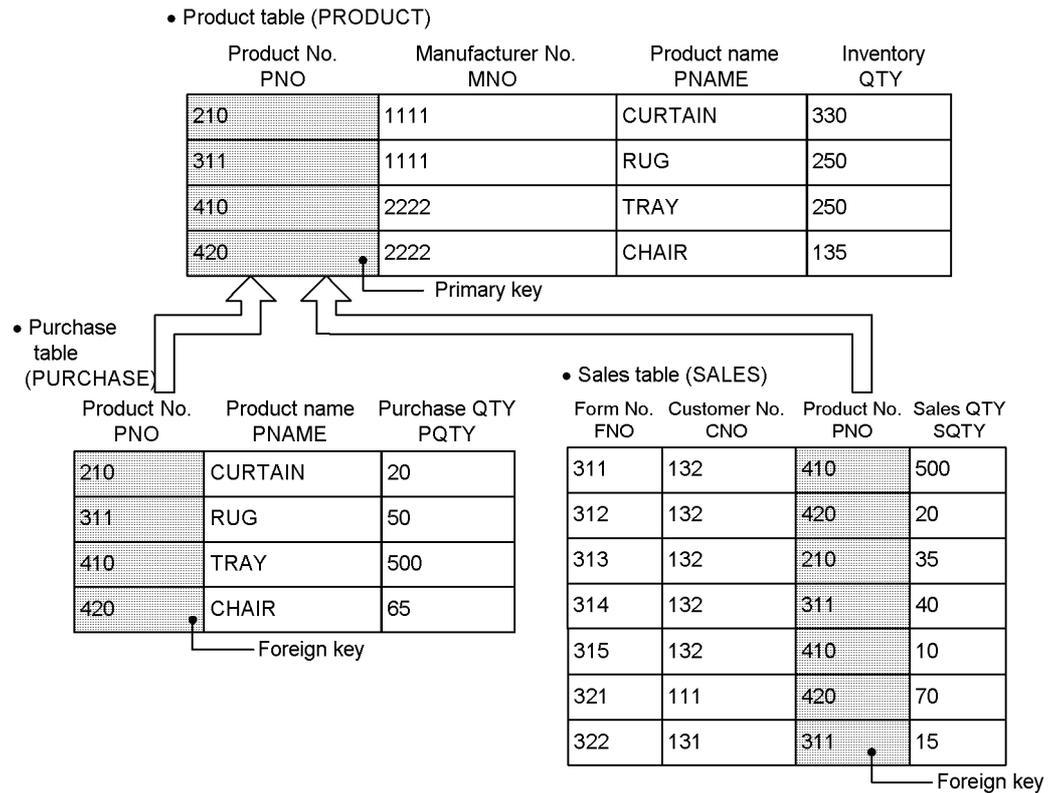
```
CREATE TABLE MANUFACTURER
(MNO CHAR(4), MNAME NCHAR(6), TELEPHONE CHAR(12))
PRIMARY KEY (MNO) ...Specification of the primary key
CREATE TABLE PRODUCT
(PNO CHAR(4), MNO CHAR(4), PNAME NCHAR(10), QTY INTEGER)
CONSTRAINT PRODUCT_FK ... Specification of the constraint name
FOREIGN KEY (MNO) ... Specification 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`.

12.18.3 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 *12.18.4 Data manipulation and integrity*. For details about how to check data integrity, see *12.18.5 Procedure for checking table integrity*.

(1) Trigger for setting or clearing check pending status

In addition to the integrity check utility, you can also use the following utilities, commands, and SQL statements to place a referencing table into check pending status or to clear a check pending status that has been applied to a referencing table.

- 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)

- The `PURGE TABLE` statement
- The `ALTER TABLE (CHANGE RDAREA)` statement

For details about utilities and commands, see the manual *HiRDB Version 8 Command Reference*. For details about SQL, see the manual *HiRDB Version 8 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.

Table 12-14 lists the storage locations of check pending status information and their contents.

Table 12-14: 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 non-partitioned 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

Table 12-15 lists operations that are disabled for tables in 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 12-15: 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 (pdrorg)	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:

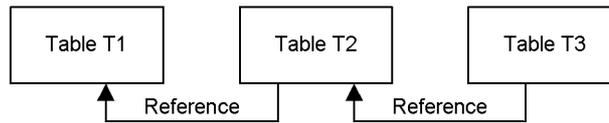
- The target table is a partitioned table, and the partitioning condition is key range partitioning or FIX hash partitioning.
- The target RDAREA is not in check pending status.

#2

Reorganizing a table partitioned using flexible hash partitioning may not be possible. For details, see *Rules and notes* in the *Database Reorganization Utility (pdrorg)* chapter of the manual *HiRDB Version 8 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

Table 12-16 lists operations restricted for each table when only table T2 is in check pending status.

Table 12-16: 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. <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 T2 is referenced to perform data integrity checking.
	DELETE (deletes rows from the target table)	
Table T2	SELECT statement (searches the target table or a list created from the target table)	These operations can be performed only when both of the following conditions are met: <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.

Target table	Restricted operation	Contents
	Reorganization by the database reorganization utility (<code>pdroorg</code>)	Reorganization may not be possible for a table partitioned using flexible hash partitioning. For details, see <i>Database Reorganization Utility (pdroorg)</i> in the manual <i>HiRDB Version 8 Command Reference</i> .
Table T3	There is no restricted operation. For <code>INSERT</code> and <code>DELETE</code> , referenced table T2 is referenced to perform data integrity checking.	

(b) When only table T3 is in check pending status

Table 12-17 lists operations restricted for each table when only T3 is in check pending status.

Table 12-17: Restricted operations when table T3 is in check pending status

Target table	Restricted operation	Contents
Table T1	<code>UPDATE</code> (updates the target table)	When the referential constraint action defined for table T2 and T3 is <code>CASCADE</code> , 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.
	<code>DELETE</code> (deletes rows from the target table)	
Table T2	<code>UPDATE</code> (updates the target table)	Restrictions depend on the referential constraint action specification defined for tables T2 and T3. <ul style="list-style-type: none"> If <code>CASCADE</code> is specified: <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> If <code>RESTRICT</code> is specified: <p>These operations can be performed. Referencing table T3 is referenced to perform data integrity checking.</p>
	<code>DELETE</code> (deletes rows from the target table)	

Target table	Restricted operation	Contents
Table T3	SELECT statement (searches the target table or a list created from the target table)	These operations can be performed only when both of the following conditions are met: <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)	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 8 Command Reference</i> .	

(c) When both tables T2 and T3 are in check pending status

Table 12-18 lists operations restricted for each table when tables T2 and T3 are in check pending status.

Table 12-18: Restricted operations when tables T2 and T3 are in check pending status

Target table	Restricted operation	Contents
Table T1	UPDATE (updates the target table)	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. 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.
	DELETE (deletes rows from the target table)	

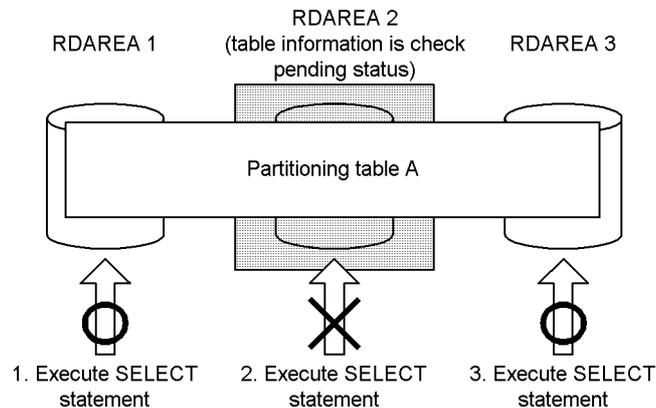
Target table	Restricted operation	Contents
Table T2	SELECT statement (searches the target table or a list created from the target table)	These operations can be performed only when both of the following conditions are met: <ul style="list-style-type: none"> The target table is a partitioned table and 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 8 Command Reference</i> .
Table T3	SELECT statement (searches the target table or a list created from the target table)	These operations can be performed only when both of the following conditions are met: <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 8 Command Reference</i> .

(5) When a partitioned table is used

Since the check pending status is managed for each RDAREA, if a partitioned table is used, and the table information in the RDAREA actually used is in check pending status, operation on the partitioned table may be restricted. The following subsections explain these cases.

Figure 12-34 shows an example wherein part of an RDAREA that stores data in a partitioned table is in check pending status.

Figure 12-34: 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.

(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 *12.18.5 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

When the open trigger attribute is `DEFER` or `SCHEDULE`:

- RDAREA is not in hold status
- RDAREA is in updatable backup hold status

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 8 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.

12.18.4 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 Tables 12-19 and 12-20 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 12-19: 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.

Operation on table or RDAREA		Condition for loss of data integrity
Database reorganization utility (pdorg)	Reload (-k reld)	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 (-k rorg)	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 (pmod)	Reinitialization of RDAREA (initialize rdarea)	The referencing table is stored in an RDAREA that is different from the re-initialized RDAREA.
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.

Table 12-20: 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 (pload)	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 (pdorg)	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.

(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. Table 12-21 lists conditions that cause moving of mismatched data between RDAREAs.

Table 12-21: 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 (pdrorg)	Reorganization (-k rorg)	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: <ol style="list-style-type: none"> 1. Perform data load for each RDAREA. 2. In HiRDB/Single Server, execute reorganization for each table.[#] In HiRDB/Parallel Server, specify the -g option to execute reorganization for each table.[#]
Rebalancing utility (pdrbal)		You add an RDAREA for a table that has mismatched data to execute the rebalancing utility (pdrbal). [#]

#

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 *12.18.5 Procedure for checking table integrity*. These conditions are the same regardless of whether the referential constraint action is RESTRICT or CASCADE.

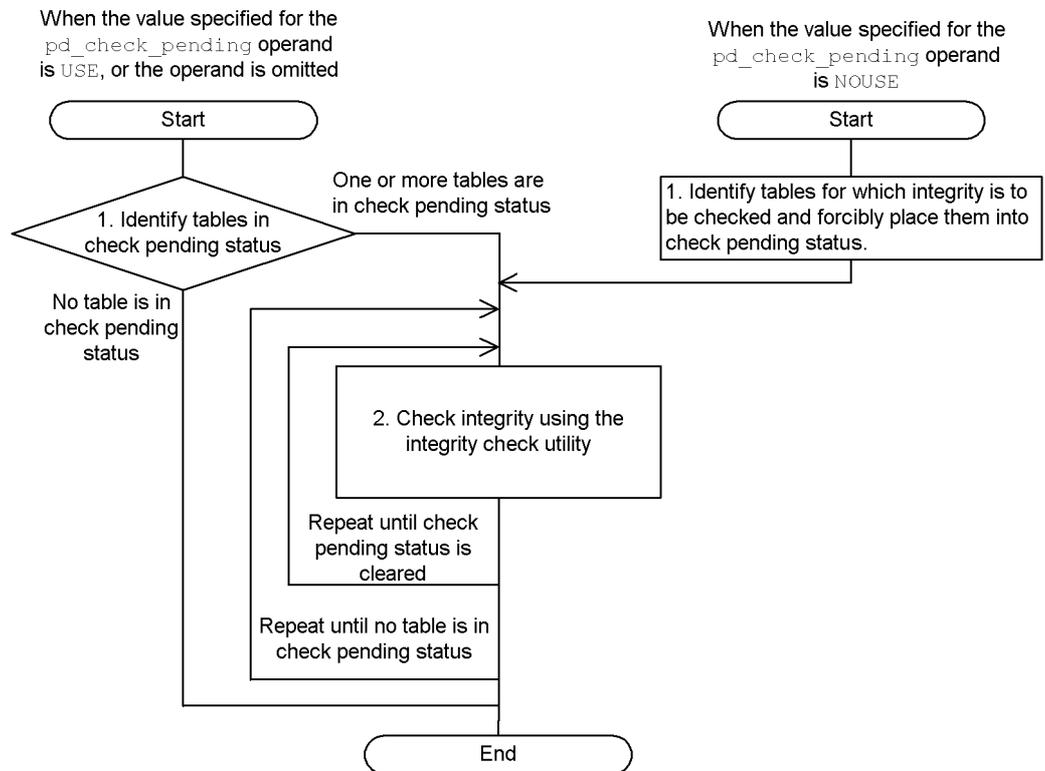
- There are two transactions, one for deleting rows from the referencing table and the other for updating or deleting the referenced table, and these transactions are executed at the same time.
- A value in the primary key column of a row that is to be deleted from the referencing table is the same as 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.

12.18.5 Procedure for checking table integrity

Figure 12-35 shows the procedure for checking data integrity.

Figure 12-35: 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.

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 *12.18.5(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).

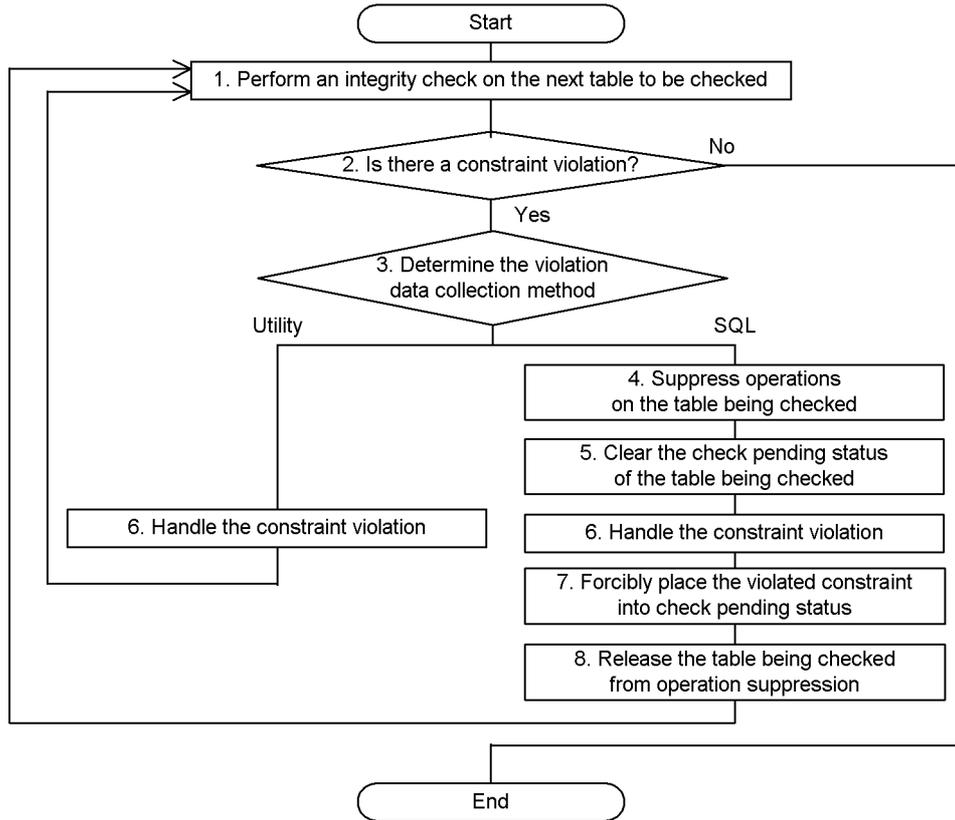
2. Use the integrity check utility to check data integrity.

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 *12.18.5(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)

Figure 12-36 shows how to use the integrity check utility to check data integrity when the value specified in the `pd_check_pending` operand is `USE`, or the operand is omitted.

Figure 12-36: Procedure for checking data integrity when check pending status is used (referential constraint)



1. Check the data integrity of the next table to be checked.
Check the data integrity for each table and constraint.
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 (pdrorg) to delete unnecessary data.

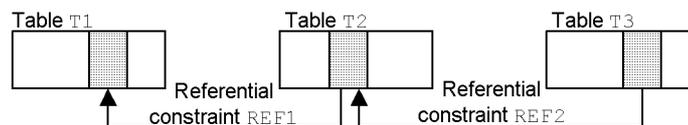
Using SQL code:

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

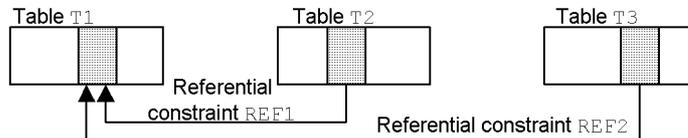
DELETE RESTRICT is specified in REF2, first delete the corresponding data in table T3 and then delete the data in table T2. If you use the UPDATE statement to correct the data, if ON UPDATE RESTRICT is specified in REF2, first delete the data in table T3 that corresponds to the pre-update data, and then 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.

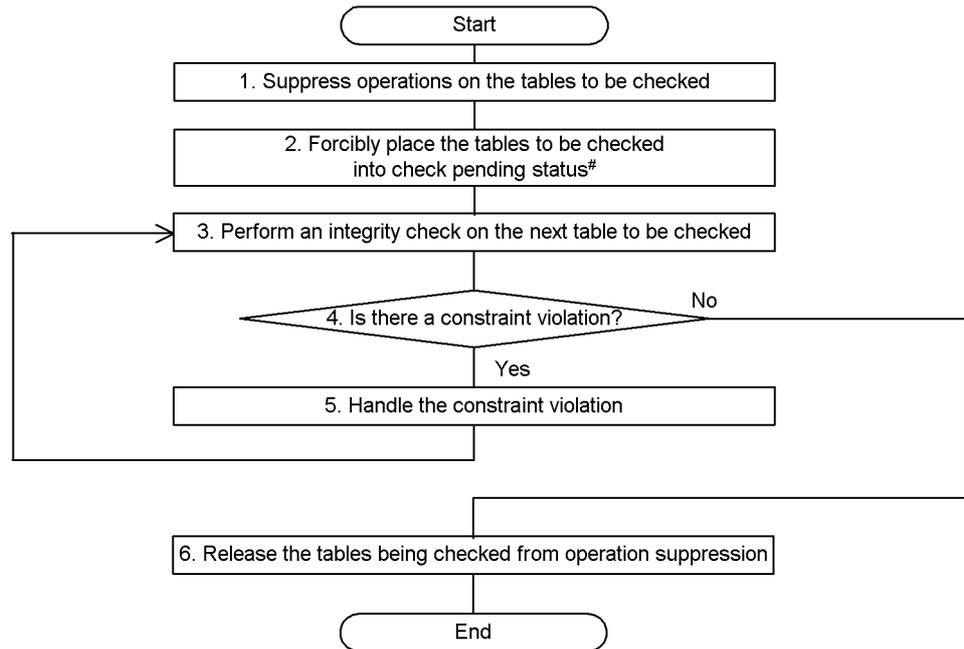
8. Release the stopped operations.

Resume performance of stopped tasks. Return to step 1 to perform data integrity checking and to check for violating data.

(2) Procedure for checking data integrity when check pending status is not used

Figure 12-37 shows how to use the integrity check utility to check data integrity when the value specified in the `pd_check_pending` operand is `NOUSE`.

Figure 12-37: Procedure for checking data integrity when check pending status is not used



The step is not required when a constraint name-basis integrity check is performed.

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.
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 12.18.5(1) *Procedure for checking data integrity when check pending status is used (referential constraint)* to correct data that violates

constraints.

6. Release the stopped operations.
Resume performance of stopped tasks.

12.18.6 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.

- 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.

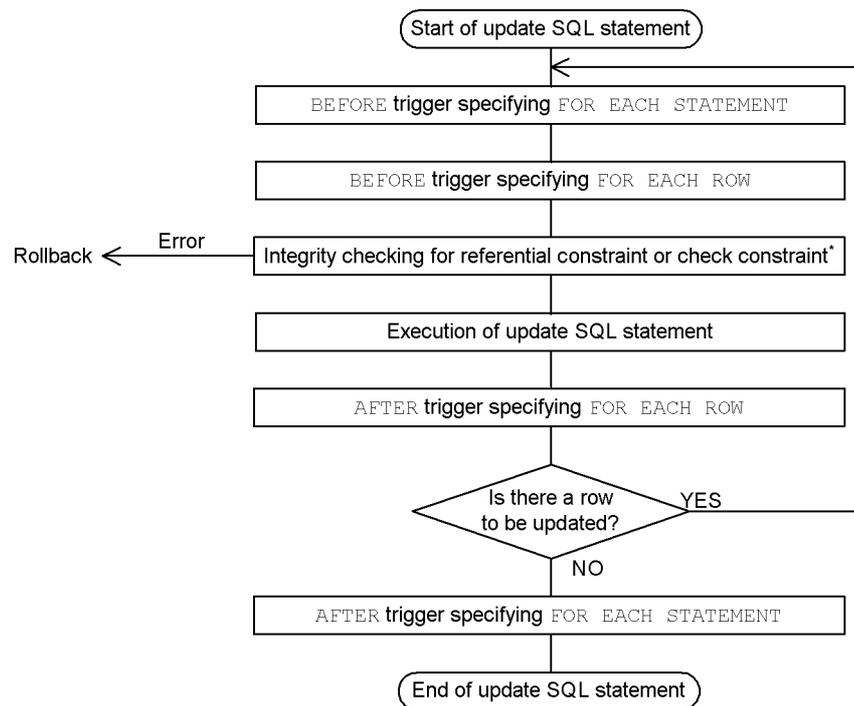
Condition for pattern 2:

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 or not the update (`INSERT`, `UPDATE`) data is contained

in the referenced table

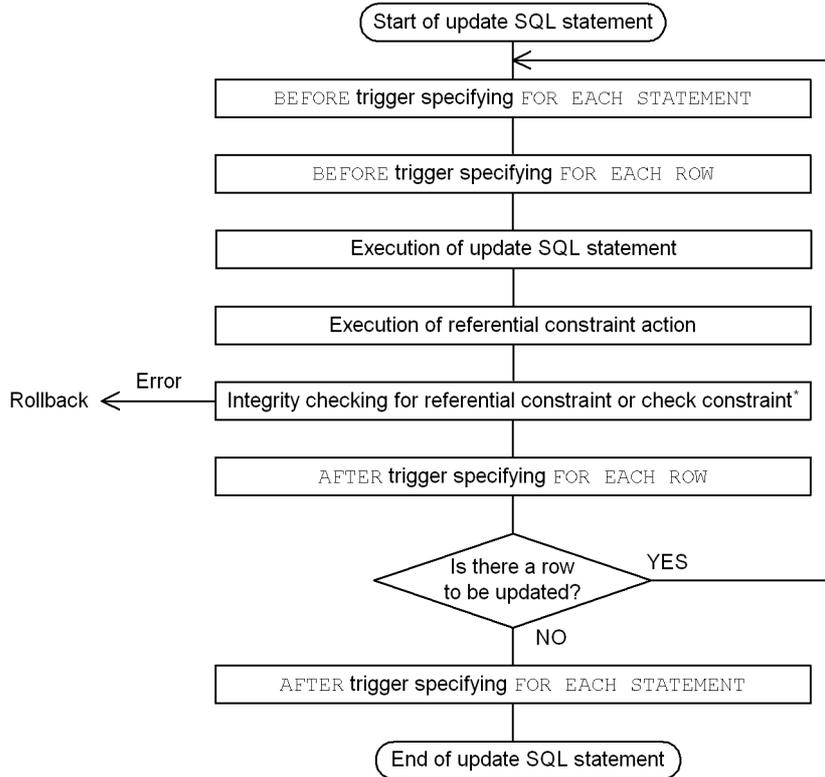
2. When the update target is the referenced table

Checking for whether or not 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.

12.18.7 Notes about linkage with related products

The following notes explain restrictions when linking with related products.

- 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 *12.18.5 Procedure for checking table integrity*.

12.19 Check constraints

12.19.1 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 *12.19.3 Check pending status*; for details about operations that cause the loss of guaranteed data integrity, see *12.19.4 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.

12.19.2 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 8 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 manual *HiRDB Version 8 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.

12.19.3 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 *12.19.4 Data manipulation and integrity*. For details of how to check data integrity, see *12.19.5 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.

Table 12-22 lists the storage locations of check pending status information and their contents.

Table 12-22: 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 non-partitioned 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 12.18.3(3) *Operations that are restricted for tables in check pending status*.

(3) When a partitioned table is used

These restrictions are the same as those for the referential constraint. See 12.18.3(5) *When a partitioned table 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 12.19.5 *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.

12.19.4 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 Table 12-23, 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.

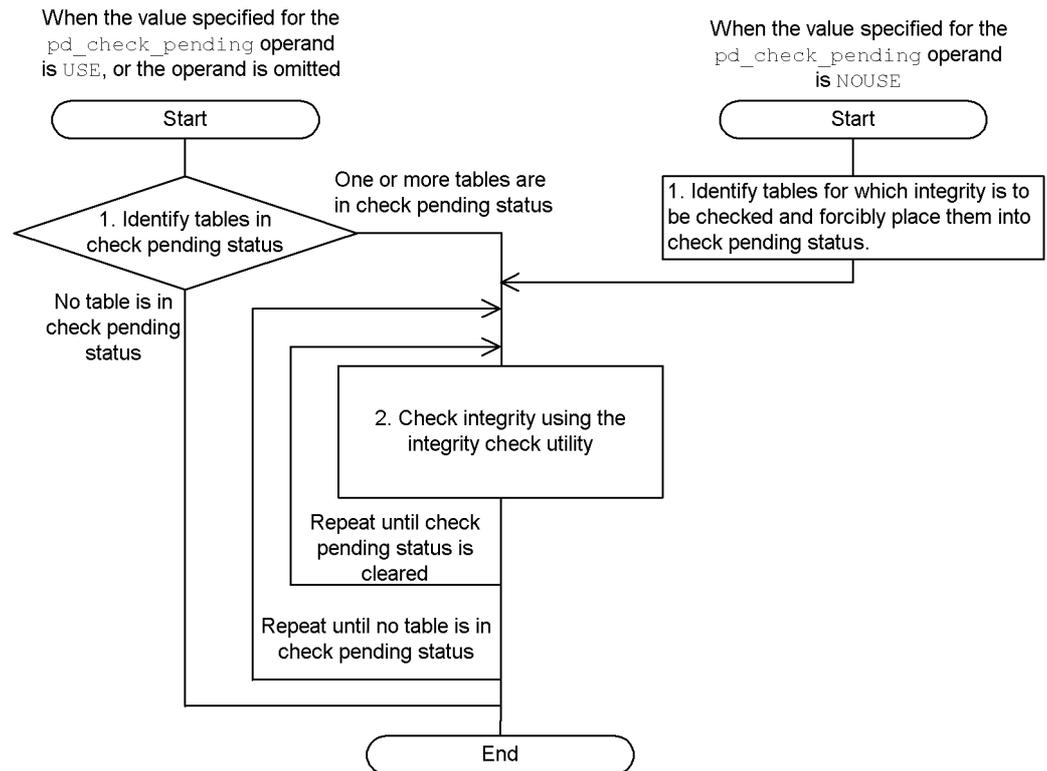
Table 12-23: Operations on check constraint tables that nullify the guarantee of data integrity and the conditions under which loss of data integrity occurs

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>pdroorg</code>)	Reload (<code>-k reload</code>)	Data that does not satisfy search conditions specified in the check constraint definition is reloaded.

12.19.5 Procedure for checking table integrity

Figure 12-38 shows the procedure for checking data integrity.

Figure 12-38: 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.

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 (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).

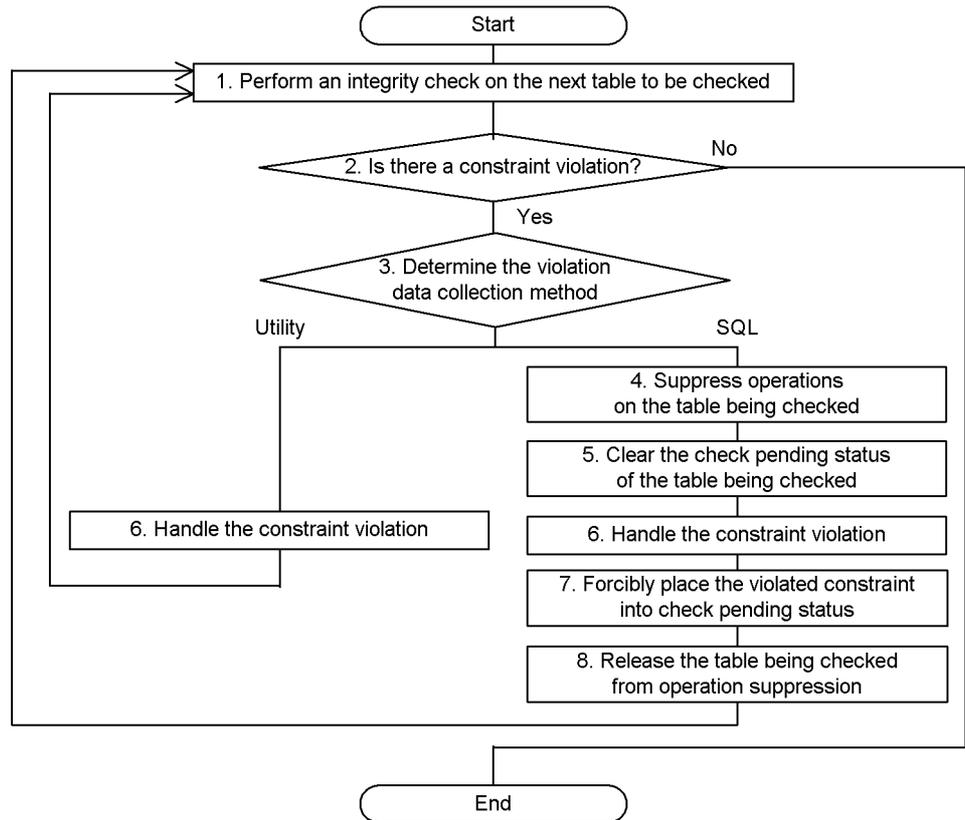
2. Use the integrity check utility to check integrity.

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 *12.18.5(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)

Figure 12-39 shows how to use the integrity check utility to check data integrity when the value specified in the `pd_check_pending` operand is `USE`, or the operand is omitted.

Figure 12-39: Procedure for checking data integrity when check pending status is used (check constraint)



1. Check the data integrity of the tables to be checked.
Check the data integrity for each table and constraint.
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.

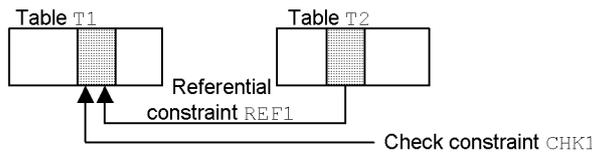
Using SQL code:

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.

12.19.6 Notes about linkage with related products**■ 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.

Chapter

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.

This chapter contains the following sections:

- 13.1 Items to be examined during index design
- 13.2 Index
- 13.3 Index row partitioning
- 13.4 Plug-in index
- 13.5 Plug-in index row partitioning

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.

Table 13-1 lists the items to be examined during index design.

Table 13-1: Items to be examined during index design

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 (`pdgetcst` 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 8 Command Reference*.

(5) Data types for which indexes cannot be defined

Indexes cannot be specified for columns of the following types:

- DECIMAL with a precision greater than 20 digits
- BLOB
- BINARY
- BOOLEAN
- 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)
 $\leq \text{MIN} \{ (\text{index-storage-RDAREA-page-size} \div 2) - 1242, 4036 \}$

If the page size of the index storage RDAREA is 4096 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.

Table 13-2 shows the priority order of index usage by HiRDB.

Table 13-2: Priority order of 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.

13. Designing Indexes

Priority	Index used by HiRDB	Example of condition specification for index column (C1)
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

Priority	Index used by HiRDB	Example of condition specification for index column (C1)
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%'
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 or not 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 manual *HiRDB Version 8 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 12-14; if negation is included, the priority order is 12 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 or not HiRDB uses an index whose exception key value is the null value

Limitation condition specified in the component column		Whether or not 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 3-14 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 Table 13-4. 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, 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)

Figure 13-1 shows an example of a partitioning key index based on the STOCK table.

Figure 13-1: Partitioning key index (single-column 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 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

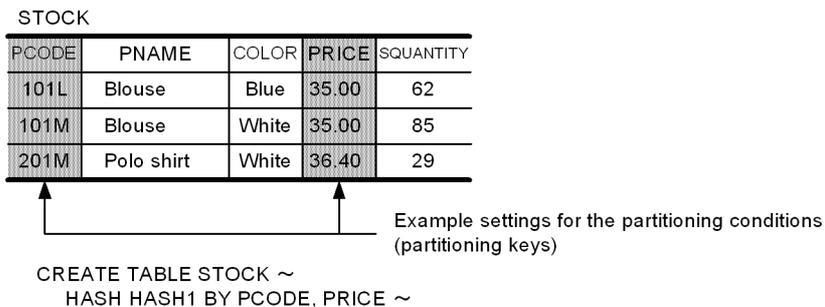
An index that satisfies the following condition is a partitioning key index:

Condition:

- Index created on multiple columns that includes all columns specified for partitioning in the same order, beginning with the partitioning key

Figure 13-2 shows an example of a partitioning key index based on the STOCK table.

Figure 13-2: Partitioning key index (multicolumn partitioning)



Explanation:

```
CREATE INDEX A12 ON STOCK (PCODE ASC, PRICE DESC) ...1
CREATE INDEX A12 ON STOCK (PCODE ASC, PRICE DESC,
                           SQUANTITY ASC) ...2
CREATE INDEX A12 ON STOCK (PRICE DESC, PCODE ASC) ...3
CREATE INDEX A12 ON STOCK (PCODE ASC, SQUANTITY
                           DESC, PRICE ASC) ...4
```

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 Table 13-5.

Table 13-5: Index partitioning guidelines

Type of index	HiRDB/Single Server	HiRDB/Parallel Server	
		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 8 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.

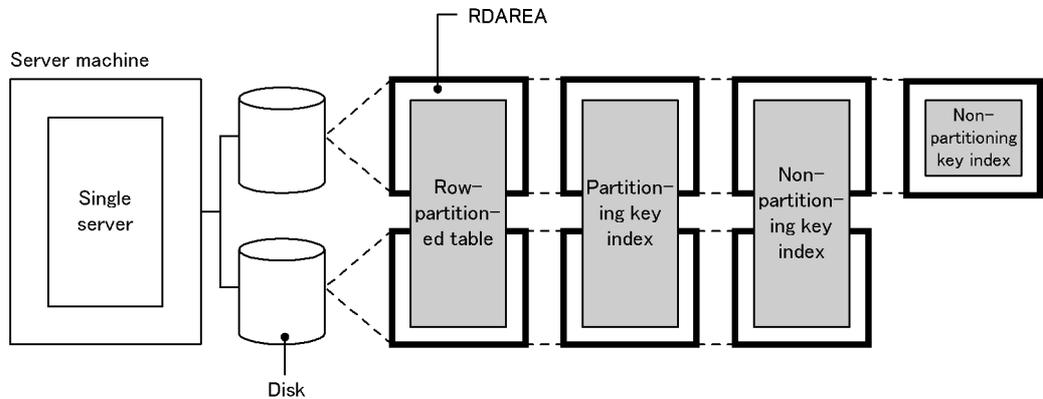
(3) Design considerations

- 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` cannot be specified for the index of a table partitioned using flexible hash partitioning.

(4) Example of index row partitioning (HiRDB/Single Server)

Figure 13-3 shows an example of index row partitioning (HiRDB/Single Server).

Figure 13-3: Example of index row partitioning (HiRDB/Single Server)



Explanation:

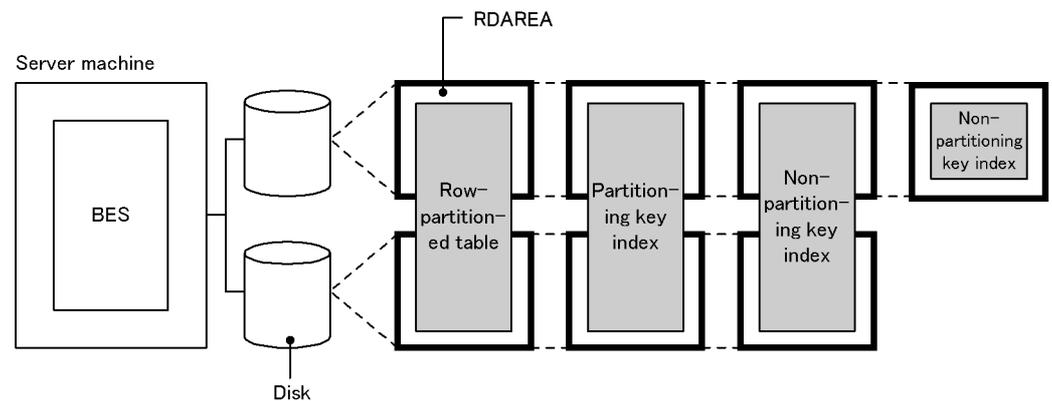
- To avoid disk access contention, place the RDAREAs storing the partitioned table on a disk separate from the RDAREAs storing its index.
- Row-partition the partitioning key index.
- If performance is more important than operability, do not row-partition the non-partitioning key index.
- If operability is more important than performance, row-partition the non-partitioning key index.

(5) Example of index row partitioning (HiRDB/Parallel Server)

(a) Partitioning a table within one server

Figure 13-4 shows an example of index row partitioning (within one server).

Figure 13-4: Example of index row partitioning (within one server)



BES: Back-end server

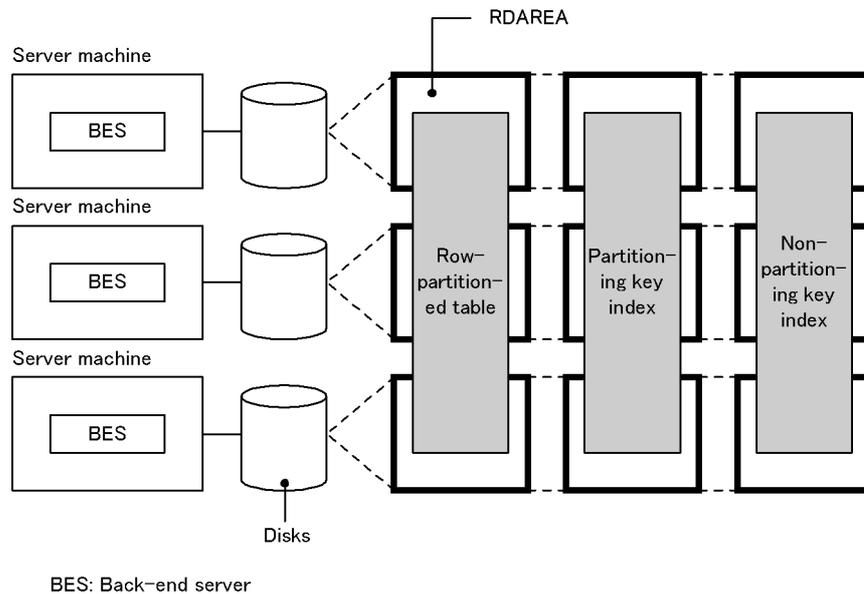
Explanation:

- 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

Figure 13-5 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

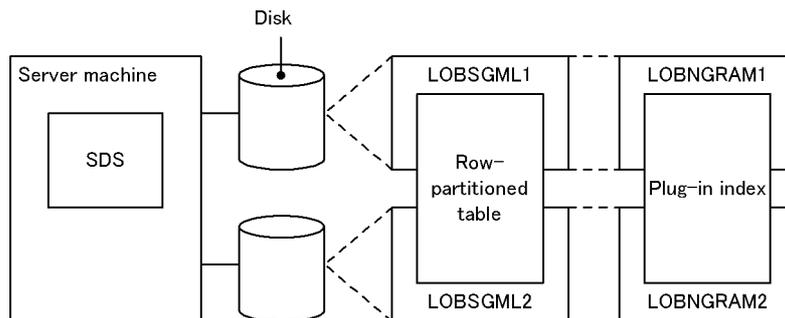
The forms of plug-in index row partitioning are described below for a HiRDB/Single Server and for a HiRDB/Parallel Server.

(a) HiRDB/Single Server

For a HiRDB/Single Server, 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)



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.

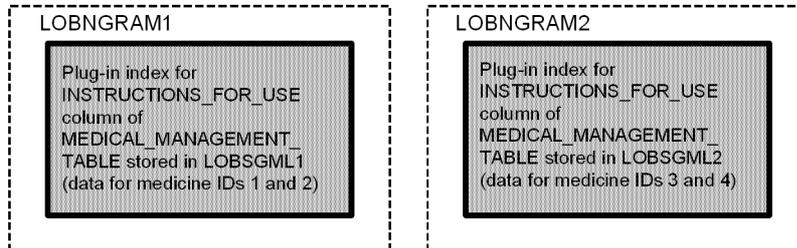
Figure 13-7: Example of plug-in index row partitioning (key range partitioning) (HiRDB/Single Server)

- 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 LOGNGRAM2.

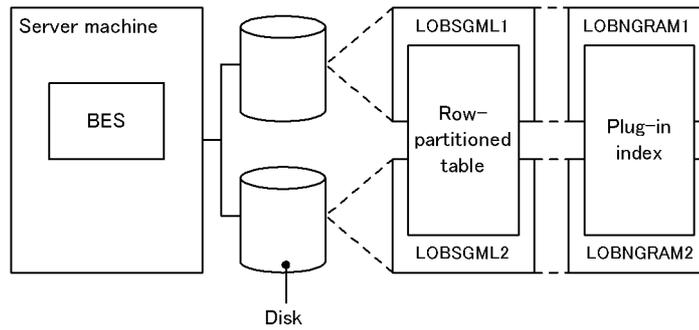
(b) HiRDB/Parallel Server

For a HiRDB/Parallel Server, 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.

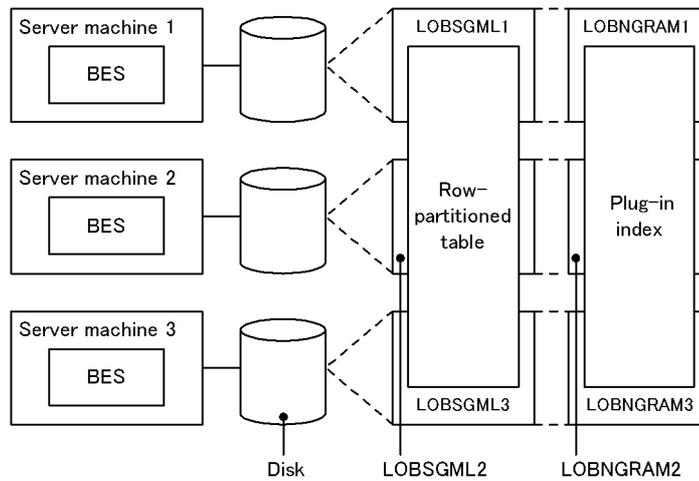
Figure 13-8 shows a form of plug-in index row partitioning. Figure 13-9 shows an example of plug-in index row partitioning based on the form shown in Figure 13-8.

Figure 13-8: Form of plug-in index row partitioning (HiRDB/Parallel Server)

- 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.

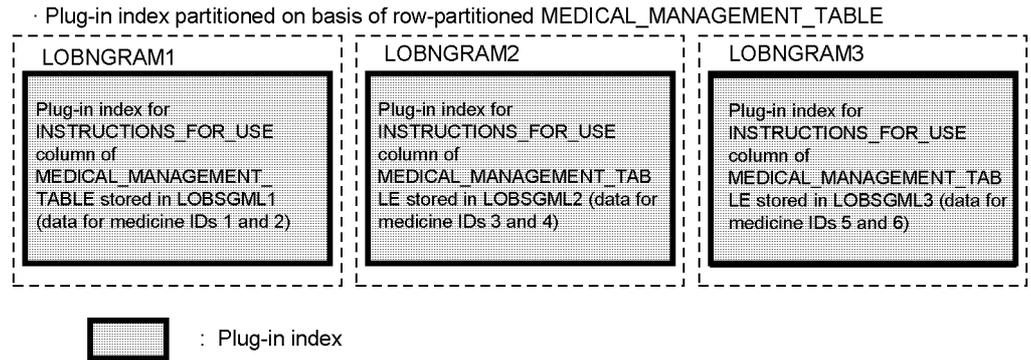
Figure 13-9: Example of plug-in index row partitioning (key range partitioning) (HiRDB/Parallel Server)

- The product management table is row partitioned onto user LOBRDAREAs LOBSGML1-LOBSGML3.

LOBSGML1Productmanagementtable	
MEDICINE_ID	INSTRUCTIONSFORUSE
Medicine1	<data><indications>Diarhea,foodpoisoning,contaminatedwateringestion,...</indications> <directions anddosage> Adults (20years andup): 10 tablets,...aftersupper.</directionsand dosage> : <warnings> Keepoutofthereachofchildren... Keeprefrigeratedafteropening...</warnings></data>
Medicine2	<data><indications>Headache,toothache,arthritis,backache,...</indications> <directions anddosage> Adults (20years andup): 5 packets..., maximumof5 packetsin a 24-hourperiod.</directionsanddosage> : <warnings> Keepoutofthereachofchildren... Forotherthanheadache,...</warnings></data>
:	:

LOBSGML2Productmanagementtable	
MEDICINE_ID	INSTRUCTIONSFORUSE
Medicine3	<data><indications>Commoncold,cold-likesymptoms,...</indications> <directions anddosage> 5times daily..within 5 minutesafterameal...</directions anddosage> : <warnings> Keepoutofdirectsunlight...Donotoperateautomobiles, etc.,aftertakingthis medicine...</warnings></data>
Medicine4	<data><indications>Eye strain,tightsoulder,...</indications> <directions anddosage> Adults (20years andup): 10 tabletsdaily...</directions anddosage> : <warnings> Keepoutofthereachofchildren...</warnings></data>
:	:

LOBSGML3Productmanagementtable	
MEDICINE_ID	INSTRUCTIONSFORUSE
Medicine5	<data><indications>Contusion, strain,musclepain,...</indications> <directions and dosage> Use appropriate size only,... 1-2 at a time ...</directions and dosage> : <warnings> Useonlyasdirected. ...</warnings></data>
Medicine6	<data><indications>Abrasions,contusions,...</indications> <directions anddosage> Adults (20years andup): Apply1 swab/day, soakedin ...</directions anddosage> : <warnings> In caseofeye contact ...</warnings></data>
:	:



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.

Chapter

14. Designing RDAREAs

This chapter explains items that should be examined while designing the segments and pages that constitute RDAREAs.

This chapter contains the following sections:

- 14.1 Items to be examined during RDAREA design
- 14.2 Segments
- 14.3 Pages
- 14.4 Designing list RDAREAs
- 14.5 Free space reusage facility
- 14.6 Shared RDAREAs (HiRDB/Parallel Server only)

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 lists the items to be examined during RDAREA design, and Table 14-2 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 a row length 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. 	

Design task and items to be examined		Advantages	Disadvantages	Section
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 FIX attribute, storage efficiency is poor.	14.3.2
	Percent-age of unused space in page set to 0	For a table with the FIX 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 reuse	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
	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.	

Design task and items to be examined		Advantages	Disadvantages	Section
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.	

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 manual *HiRDB Version 8 System Operation Guide*.
3. The structure of an index depends on whether or not 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

Table 14-3 shows the various segments statuses.

Table 14-3: Segment statuses

Segment status	Explanation
Used segment*	This is a segment that stores table or index data. A segment that is full, such that no more data can be added, is called a <i>full segment</i> . A segment from which all the data has been deleted, such that all pages are free pages (used free pages or unused pages), is called a <i>used free segment</i> .
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

The segment size for RDAREAs should be approximately 10-20 pages per segment. This section 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 row length 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.

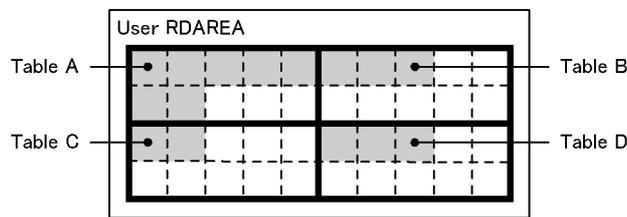
Notes

- 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.

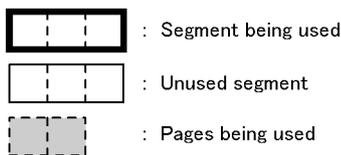
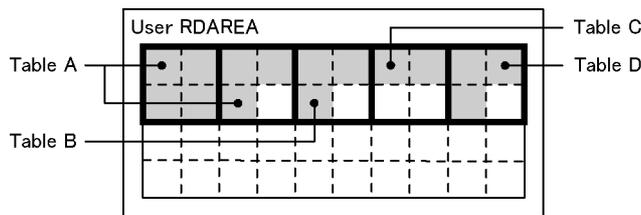
Figure 14-1 provides an overview of user RDAREAs depending on the segment size.

Figure 14-1: Overview of RDAREAs depending on segment size

- Large segment size



- Small segment size



(3) Setting procedure

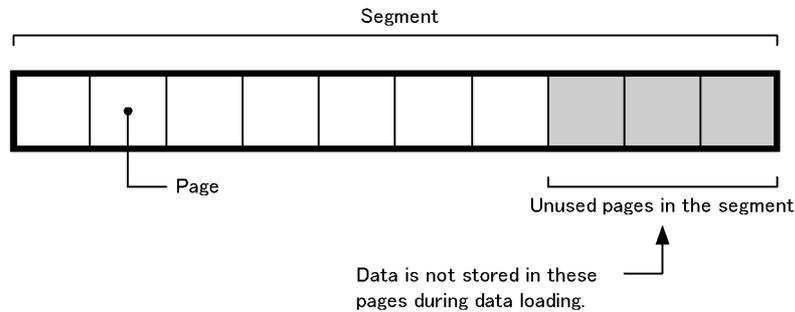
The `create rdarea` statement of the database initialization utility (`pdinit`) or the

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. Figure 14-2 provides an overview of the percentage of 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`), etc.
- 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 (`pdorg` 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 manual *HiRDB Version 8 System Operation Guide*.

Segments are released if you perform the following types of operations in addition to table reorganization:

- Execution of the `PURGE TABLE` statement
- Reinitialization of RDAREAs
- Deletion of table or index definitions
- Deletion of index definitions
- Execution of data loading in the creation mode (`-d` option specified)

14.3 Pages

Table 14-4 shows the page statuses.

Table 14-4: Page statuses

Page status	Explanation
Used page	This is a page that stores table or index data. A page that is full, such that no more data can be added, is called a <i>full page</i> . A page from which all the data has been deleted, such that there is no data on the page, is called a <i>used free page</i> .
Unused page	This is a page that has never been used.
Free page	This is a page that does not store any data. Used free pages and unused pages are <i>free pages</i> .

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:

Page length - length of control information section × number of rows that can be stored per page.
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 `MVARCHAR` 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 `MVARCHAR`, 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 *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 manual *HiRDB Version 8 System Operation Guide*.

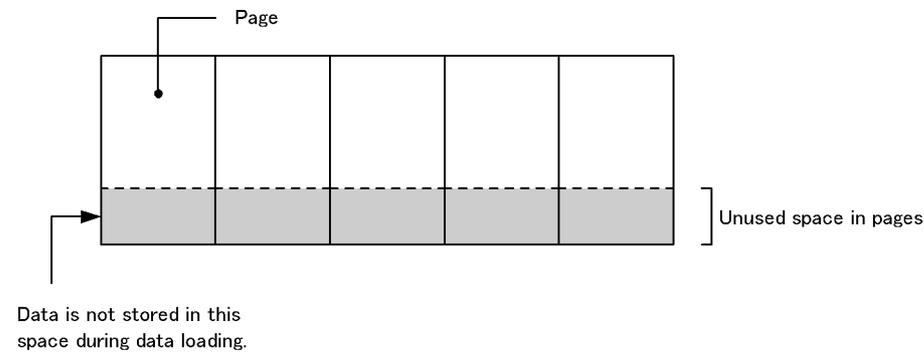
14.3.2 Setting the percentage of unused space in a page

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.

Figure 14-3 shows an overview of 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 CREATE TABLE or CREATE INDEX definition SQL statement.

(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 manual *HiRDB Version 8 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 50000.

(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 3000	4096	1
Average number of rows in a list created within the server is 3000 to 6000	4096	2
Average number of rows in a list created within the server is more than 6000	See (a)	See (b)

(a) Obtaining the page length when the average number of rows in a list is more than 6000

Specify the page length in the range of 4096 to 8192. 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 6000

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:

$$\text{Number of rows that can be stored per list segment} = \text{number of rows that can be stored in one list page} \times \text{segment size}$$

(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

$$\text{Number of segments required for a list RDAREA} = \uparrow \{ \uparrow a \div b \uparrow \times (c + 0.5) \} \uparrow$$

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:

\uparrow average number of rows in list within the server \div number of rows that can be stored per list segment \uparrow

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 reuse facility

The free space reuse 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 reuse 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 allocation 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.

- 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.

14.5.2 Free space reuse facility

The *free space reuse 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 allocation mode for allocation of a new unused segment.

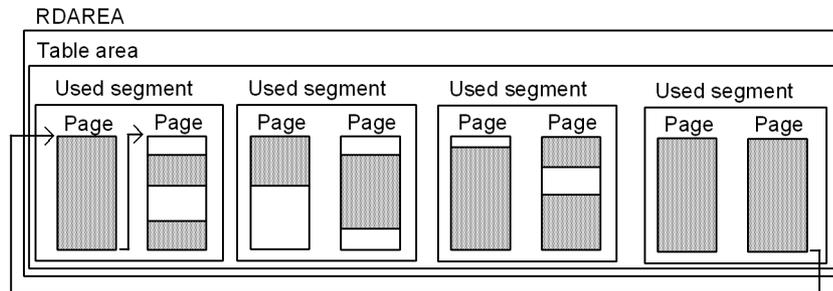
If the number of segments is not specified, free space is not reused until there are no more unused pages in an RDAREA. When the free space reuse facility is not used, free space is searched for each time from the beginning of the used segments. When the facility is used, the search efficiency is better than when it is not used, because after switching to the free page reuse mode the first time, the search start position is

remembered for the subsequent search.

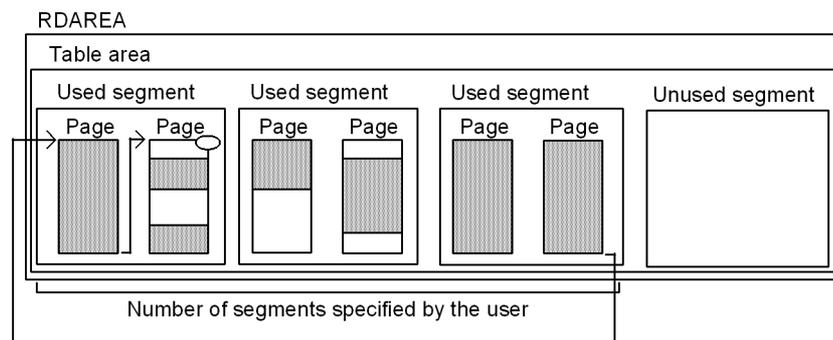
Figure 14-4 provides an overview of the free space reuse facility.

Figure 14-4: Overview of the free space reuse facility

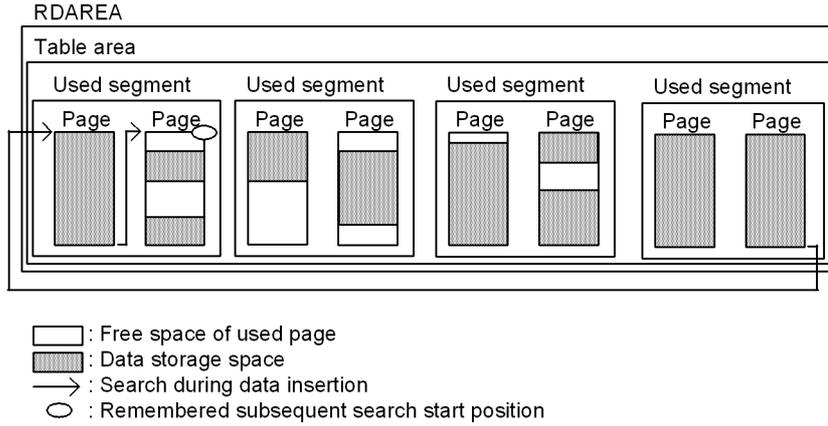
- Free space reuse facility not used



- 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

- 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 section 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 (`pdroorg`) can be executed quickly, the free space reuse facility would not be appropriate, and database reorganization would provide better performance.

14.5.4 Considerations

The free space reuse 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.

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. For a HiRDB/Parallel Server, 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 `KFP2030-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 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.
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 reusage 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 reusage 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 reusage facility, specify `ALTER TABLE` with `SEGMENT REUSE NO` specified. When this is done, the free space reusage facility will stop immediately, and unused segments will be newly allocated.

14.5.6 Checking execution status

You can check whether the free space reusage 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 `KFPH22031-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 reusage 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.

Item	Explanation	Measure
Number of failed page searches by the free space reuse facility	<p>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 conducted even though there is no free space, pointless search processing is performed.</p> <p>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.</p>	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

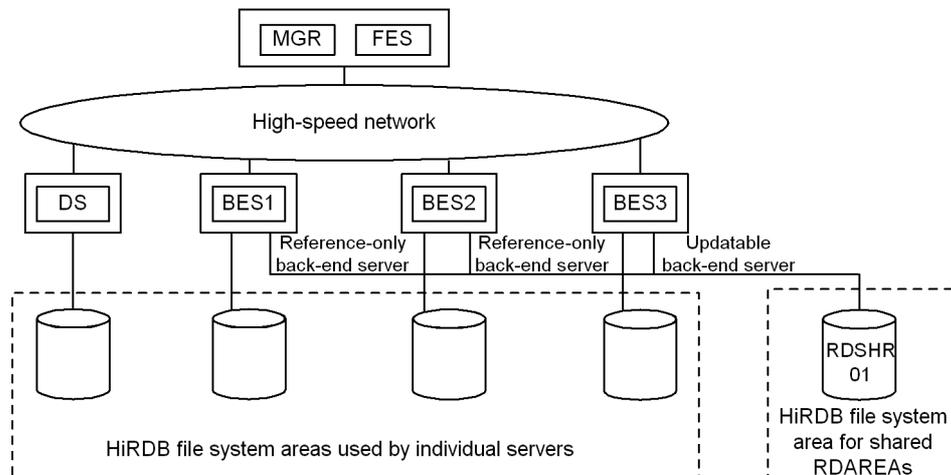
- In the following cases, the free space reuse facility will not function:
 - When data is stored using the hash facility for hash row partitioning
 - When data dictionary tables are stored
 - When data is stored in tables by means of data loading or by the database reorganization facility (`pdorg`)
- 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`).
- When the free space reuse facility is used with variable-length rows (including `BINARY` type), even if there is the same volume of additions and deletions, the number of segments used may increase.
- Even when searching is executed in the free page reuse mode, deleted space cannot be reused within the same transaction.

14.6 Shared RDAREAs (HiRDB/Parallel Server 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. Figure 14-5 provides an overview of a shared RDAREA.

Only a HiRDB/Parallel Server 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. 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 14-5:

```
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 HiRDB 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 *12.17.3 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 `pdsqls -m` command to display the status of the shared RDAREA for all the back-end servers.

(6) 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.

(7) Notes about using shared RDAREAs

1. To back up a shared RDAREA, you must use the reference/update-impossible mode or reference-possible mode. The following specifications are not permitted:
 - `pdcopy -M s` (updatable mode)
 - `pdhold -bu` (updatable backup-hold)
 - `pdhold -buw` (updatable backup-hold (WAIT mode))
2. 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.

3. No floating servers can be installed because a shared RDAREA is placed in all back-end servers.
4. A shared table cannot be the replication target.

5. When you use local buffers to update a table and index in a shared RDAREA, first issue the `LOCK TABLE` statement. If a shared table or index is updated without issuing the `LOCK TABLE` statement, and the server process terminates abnormally, it may not be possible to restore the updated pages from the global buffer when rollback processing executes restore processing. 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.

Chapter

15. Storage Requirements for HiRDB

This chapter explains the HiRDB storage requirements.

This chapter contains the following sections:

- 15.1 Estimating the memory size required for a HiRDB/Single Server
- 15.2 Estimating the memory size required for a HiRDB/Parallel Server

15.1 Estimating the memory size required for a HiRDB/Single Server

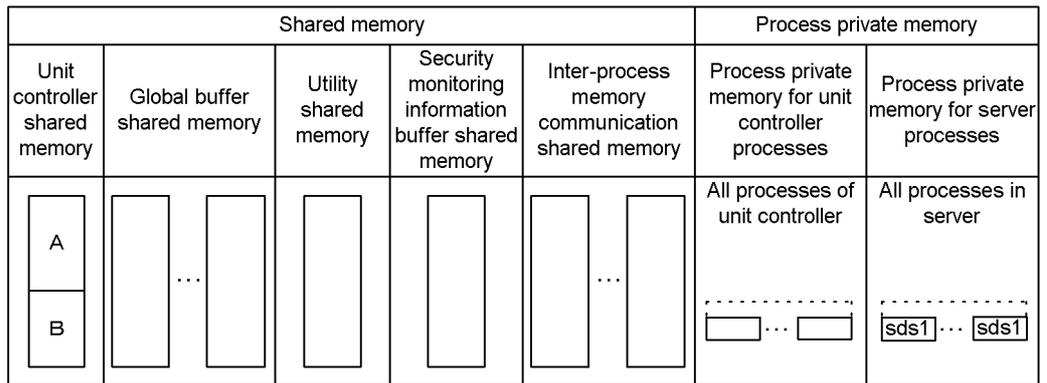
This section explains how to estimate the size of the memory required for a HiRDB/Single Server. The topics covered include:

- 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

Figure 15-1 shows the memory allocation for a HiRDB/Single Server.

Figure 15-1: Memory allocation for a HiRDB/Single Server



A: Part used by a unit controller process.
 B: Part used by a Single Server process.
 sds: Single Server.

Table 15-1 provides the details of the shared memories for a HiRDB/Single Server.

Table 15-1: HiRDB/Single Server 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
Purpose	System control	Global buffers	Communication between the unit controller and utilities	Security monitoring information buffer	Client-server inter-process memory communication
Processes	All HiRDB processes	Single Server	Utility processes	Single server	Single Server, client processes
Number of segments	1	<ul style="list-style-type: none"> When the global buffer dynamic update facility is not used: 1-16 When the global buffer dynamic update facility is used: 32-bit mode: 1-516 64-bit mode: 1-1,016 	1	1	Number of clients connected using the PDIPC=MEMORY environment variable (0-2000) × 2
Maximum value per segment	See Table 15-2 <i>Size of memory required for a HiRDB/Single Server</i> . It may not be possible to allocate this size depending on the OS environment	Divide the segment by the SHMMAX operand value. It may not be possible to allocate this size depending on the OS environment.	See Table 15-2 <i>Size of memory required for a HiRDB/Single Server</i> .	See Table 15-2 <i>Size of memory required for a HiRDB/Single Server</i> . It may not be possible to allocate this size depending on the OS environment.	See Table 15-2 <i>Size of memory required for a HiRDB/Single Server</i> . It may not be possible to allocate this size depending on the OS environment.

15. Storage Requirements for HiRDB

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
Allocation conditions	None	There must be a global buffer definition	Specify <code>pd_utl_exec_mode=1</code>	Specify the <code>pd_aud_file_name</code> operand as the HiRDB file system area name for the audit trail file.	There are clients connected using the <code>PDIPC=MEMORY</code> environment variable
Creation timing	At unit activation (including standby unit activation when the rapid system switchover facility is used)	<ul style="list-style-type: none"> At server activation (including standby unit activation when the rapid system switchover facility is used) When <code>pdbufmod -k {add upd}</code> is executed 	When utilities are executed	When a HiRDB/Single Server starts up	When client and server are connected
Deletion timing	At next unit activation (including standby unit activation when 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

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
Indication by pdls -d mem	Indicated	Indicated	Indicated	Indicated	Not indicated
SHM-OWNER of pdls -d mem	MANAGER	Server name	UTILITY	AUDEF	Not indicated
Related operands	<ul style="list-style-type: none"> pd_sds_shmpool_size 	<ul style="list-style-type: none"> pd_dbbuff_modify pdbuffer SHMMAX 	<ul style="list-style-type: none"> pd_utl_exec_mode 	<ul style="list-style-type: none"> Operands related to the security audit facility[#] 	<ul style="list-style-type: none"> PDIPC PDSENDMEMSIZE PDRECVMEMSIZE
Remarks	—	—	Can be created only when pd_utl_exec_mode=1 (when pd_utl_exec_mode=0, the relevant space is allocated in the unit controller shared memory).	—	—

Legend:

— : Not applicable.

#

For details, see the manual *HiRDB Version 8 System Definition*.

15.1.2 Calculation of required memory

The size of the memory required for a HiRDB/Single Server is the sum of the items listed in Table 15-2.

Table 15-2: Size of memory required for a HiRDB/Single Server

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 $E + \uparrow \{(64 + 48 \times (u + 1)) \times (\text{value of } pd_max_server_process - b - 3) + w\} \div 1024 \uparrow$ 64-bit mode $E + \uparrow \{(64 + 64 \times (u + 1)) \times (\text{value of } pd_max_server_process - b - 3) + w\} \div 1024 \uparrow$ <ul style="list-style-type: none"> When using plug-ins, add: + 1400 When using the asynchronous READ facility, add: + <i>r</i> When <i>fixed</i> was specified in the <i>pd_process_terminator</i> operand, add: + <i>F</i> × (value of <i>pd_process_terminator_max</i> - 1) 	
	Process private area used by single server process ¹	<i>pd_work_buff_mode=each</i> specified	<ul style="list-style-type: none"> 32-bit mode $\{G + g + (a + 9) \times c + h + i + m + p + q + s\} \times (b + 3) + \uparrow (64 + 48 \times (b + 1)) \div 1024 \uparrow \times (v + 3)$ 64-bit mode $\{G + g + (a + 9) \times c + h + i + m + p + q + s\} \times (b + 3) + \uparrow (64 + 64 \times (b + 1)) \div 1024 \uparrow \times (v + 3)$
		<i>pd_work_buff_mode=pool</i> specified or omitted	<ul style="list-style-type: none"> 32-bit mode $(G + g + a + 9 + \uparrow a \div 128 \times 0.1 \uparrow + 11 + h + i + m + p + q + s) \times (b + 3) + n + \uparrow (64 + 48 \times (b + 1)) \div 1024 \uparrow \times (v + 3)$ 64-bit mode $(G + g + a + 9 + \uparrow a \div 128 \times 0.1 \uparrow + 15 + h + i + p + q + s) \times (b + 3) + n + \uparrow (64 + 64 \times (b + 1)) \div 1024 \uparrow \times (v + 3)$

Item		Required memory (KB)
Shared memory	Space used by the unit controller in the unit controller shared memory	$\uparrow d \div 1024 \uparrow$
	Space used by the Single Server in the unit controller shared memory	e
	Global buffer shared memory	f
	Utility shared memory	t
	Security audit information buffer shared memory	<ul style="list-style-type: none"> ■ For automatic calculation by the system: $\uparrow 0.3 + \text{MAX}\{(H + 100), (H \times 1.2)\} \times 0.25 \uparrow$ ■ For user -specified values (specify the <code>pd_audit_def_buffer_size</code> operand): $\uparrow 0.3 + H \times 0.25 \uparrow$
	Inter-process memory communication shared memory ²	$j \times k$

¹ Add a value of 300 per process if you use plug-ins.

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 manual *HiRDB Version 8 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

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 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 `PDSENDMEMSIZE` + 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 and version. For details about the memory requirement for your Java virtual machine, see the applicable manual.

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 + $\uparrow(\text{work table extended buffer size} \div 128) \times 0.1 \uparrow$

- 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 manual *HiRDB Version 8 UAP Development Guide*.

- Work table extended buffer size based on the increase in the number of work tables = $\text{MIN}\{ (\text{number of work tables used} \times 128 - \text{value of the } \text{pd_work_buff_size} \text{ operand}), (\text{value of } \text{pd_work_buff_expand_limit} \text{ operand} - \text{value of } \text{pd_work_buff_size} \text{ operand}) \} \times (\text{number of users such that the number of work tables is greater than the value of the } \text{pd_work_buff_size} \text{ operand} \div 128)$

Number of work tables used = $\text{MAX}(\text{number of work table files used per SQL statement}, \text{number of work table files used by the } \text{ASSIGN LIST} \text{ statement})$

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)*.

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:

$$\begin{aligned} & (90 + \\ & 90 \\ & \sum_{i=1} \text{Memory used by the RDAREA for management of the HiRDB file system area}) \\ & \times \text{value of } \text{pd_max_ard_process} \end{aligned}$$

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 following formula based on the initial settings:

$$\{(\text{Number of files}^1 + \text{number of extensions}^2) \div 64\} \times 1.5^3$$

¹ Value specified by `pdfmkfs -l`.

² Value specified by `pdfmkfs -e`.

³ Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2048.

s: HiRDB file system memory size

Determine with the following formula (in kilobytes):

$$347 + \text{Memory used by the work tables for management of the HiRDB file system area} + \text{Memory used by the system logs for management of the HiRDB file system area} + 90 \sum_{i=1} \text{Memory used by the RDAREA for management of the HiRDB file system area}$$

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 following formula based on the initial settings:

$$\{(\text{Number of files}^1 + \text{number of extensions}^2) \div 64\} \times 1.5^3$$

¹ Value specified by `pdfmkfs -l`.

² Value specified by `pdfmkfs -e`.

³ Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2048.

t: When value of `pd_utl_exec_mode` is 0: 0

When value of `pd_utl_exec_mode` is 1: $\uparrow \{(b \times 2000 + 136) \div 1024\} \uparrow \times 1024$

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 following formula to determine the size (in bytes):

$A + B$

- If `commit` is specified in the `pd_dbsync_point` operand, add:
 $+ 112 \times (\text{value of } pd_max_users \times 2 + 7)$
- If the number of RDAREA storage areas created in a HiRDB file system area using the raw I/O facility is 1001 or greater, 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 $291722 + 4 \times \text{value of } pd_max_rdarea_no$ $+ \{48 \times (\text{value of } pd_max_rdarea_no + \text{number of tables}) + 304\} \times (\text{value of } pd_max_users \times 2 + 7)$ • 64-bit mode $370682 + 8 \times \text{value of } pd_max_rdarea_no$ $+ \{64 \times (\text{value of } pd_max_rdarea_no + \text{number of tables}) + 512\} \times (\text{value of } pd_max_users \times 2 + 7)$ <p>Number of tables: Number of data dictionary tables + MAX {value of <code>pd_max_access_tables</code>, 500}</p>

Variable	Value
<p><i>B</i></p>	<p>$b1 \times X + b2 \times Y$</p> <p><i>b1</i>: When the record length of the server status file < 4096 $\text{MAX}(\lfloor (3400 \div (\lfloor ((\text{record length} - 40) - 308) \div 20 \rfloor)) \rfloor$ $+ (\lfloor (\text{record length} - 40) \div 20 \rfloor) \times (\text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) - 1))$ $+ 0.7 \rfloor, 1) \times \text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) \times (\text{record length} - 40)$ When $4096 \leq \text{record length of server status file} < 12288$ $\text{MAX}(\lfloor (3400 \div (\lfloor (((\text{record length} - 40) - 308) \div 20) \rfloor) + 0.7) \rfloor, 1)$ $\times (\text{record length} - 40)$ When $12288 \leq \text{record length of server status file}$ $\text{MAX}(\lfloor (3400 \div (\lfloor (((\text{record length} - 40) - 836) \div 20) \rfloor) + 0.7) \rfloor, 1)$ $\times (\text{record length} - 40)$</p> <p><i>X</i>: When the number of RDAREAs ≤ 3400: 1 When $3401 \leq \text{number of RDAREAs} \leq 6800$: 2 When $6801 \leq \text{number of RDAREAs}$: 3</p> <p><i>b2</i>: When the record length of the server status file < 4096 $(\lfloor (5662310 \div (\lfloor ((\text{record length} - 40) - 308) \div 20 \rfloor)) \rfloor$ $+ (\lfloor (\text{record length} - 40) \div 20 \rfloor) \times (\text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) - 1))$ $+ 0.7 \rfloor) \times \text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) \times (\text{record length} - 40)$ When $4096 \leq \text{record length of server status file} < 12288$ $\lfloor (5662310 \div (\lfloor (((\text{record length} - 40) - 308) \div 20) \rfloor) + 0.7) \rfloor$ $\times (\text{record length} - 40)$ When $12288 \leq \text{record length of server status file}$ $\lfloor (5662310 \div (\lfloor (((\text{record length} - 40) - 836) \div 20) \rfloor) + 0.7) \rfloor$ $\times (\text{record length} - 40)$</p> <p><i>Y</i>: When the number of RDAREAs ≤ 10200: 0 When $10201 \leq \text{number of RDAREAs} \leq 5672510$: 1 When $5672511 \leq \text{number of RDAREAs} \leq 11334820$: 2 When $11334821 \leq \text{number of RDAREAs}$: 3</p>
<p><i>D</i></p>	<ul style="list-style-type: none"> • 32-bit mode <p>$12012 \times (\lceil (\text{number of RDAREA storage areas created in a HiRDB file system area using the raw I/O facility} - 1000) \div 1000 \rceil)$</p> <ul style="list-style-type: none"> • 64-bit mode <p>$16016 \times (\lceil (\text{number of RDAREA storage areas created in a HiRDB file system area using the raw I/O facility} - 1000) \div 1000 \rceil)$</p>

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
Windows (32-bit mode)	73,100	2,700	4,800
Windows Server 2003 (IPF)	154,500	5,000	11,500
Windows (x64)	90,600	4,000	10,600

H: The number of objects specified in a narrowed search using the security audit facility audit trail

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 INSERT (-SELECT) statement	<p>When none of 1-8 as follows are applicable: 0</p> <p>When any of 1-8 as follows are applicable: Sum of the applicable values from 1-8</p> <ol style="list-style-type: none"> 1. When multiple tables are joined for retrieval Number of additional work tables = (Number of joined tables - 1) × 2 + 1 2. When specifying the ORDER BY clause Number of additional work tables = 2 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) × 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 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: $\{ \uparrow (432 + 304 \times n) \div 1024 \uparrow + 338 + x \} \times 1024$ Value of <code>pd_utl_exec_mode</code> set to 1: $\{ \uparrow (432 + 304 \times n) \div 1024 \uparrow + \uparrow (m \times 2000 + 136) \div 1024 \uparrow + y \} \times 1024$ x : Single server: $116 + 5 \times (m + 3) + 14$ y : Single server: $5 \times (m + 3) + 14$ m : Value of <code>pd_max_users</code> n : Number of servers in unit + number of utility servers in unit + 1 Number of utility servers in the unit: 26 + 12
Lock server	$192 + 48 + c + d + 48 + 4096 + g + 48 + i + 48 + 4096 + 48 + n + 16 + t + u + 16$ c : When <code>pd_lck_hash_entry</code> is omitted or 0 is specified: $(\downarrow (8 + 4 \times \text{MAX}(((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2) + \text{value of } \text{pd_lck_pool_size} \times 6) \div 10,11261)) \div 16 \downarrow + 1) \times 16$ When 2 or a greater non-prime number is specified for <code>pd_lck_hash_entry</code> : $(\downarrow (8 + 4 \times \text{largest prime number that is less than the value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$ When 1 or a prime number is specified for <code>pd_lck_hash_entry</code> : $(\downarrow (8 + 4 \times \text{value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$ d : $((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2) + \text{value of } \text{pd_lck_pool_size} \times 6) \times 96$ g : When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $((p + 3) \times 3 + p) \times 256$ When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $((p + 3) \times 3 + 32) \times 256$

Process item	Shared memory calculation formula (bytes)
	<p><i>i</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $((\text{value of } pd_lck_pool_size \times 8 + (p + 3)) \times (\text{value of } pd_max_access_tables + 4)) \times 2$ $+ p \times (\text{value of } pd_max_rdarea_no + 1) + (p + 3) \times 2 \times 2 \times 5$ rounded up to the next even value $\times 64$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $((\text{value of } pd_lck_pool_size \times 8 + (p + 3)) \times (\text{value of } pd_max_access_tables + 4)) \times 2$ $+ 32 \times (\text{value of } pd_max_rdarea_no + 1) + (p + 3) \times 2 \times 2 \times 5$ rounded up to the next even value $\times 64$</p> <p><i>n</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $((p + 3) \times 3 + p) \times 48$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $((p + 3) \times 3 + 32) \times 48$</p> <p><i>p</i>: value of <code>pd_max_users</code></p> <p><i>t</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $32 + ((p + 3) \times 3 + p) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow$ $\times 4$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $32 + ((p + 3) \times 3 + 32) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow$ $\times 4$</p> <p><i>u</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$: $32 + ((\text{value of } pd_lck_pool_size \times 8 + (p + 3)) \times (\text{value of } pd_max_access_tables + 4)) \times 2$ $+ p \times (\text{value of } pd_max_rdarea_no + 1) + (p + 3) \times 2 \times 2 \times 5$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors$ $\div 16 \uparrow \times 4$</p> <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$: $32 + ((\text{value of } pd_lck_pool_size \times 8 + (p + 3)) \times (\text{value of } pd_max_access_tables + 4)) \times 2$ $+ 32 \times (\text{value of } pd_max_rdarea_no + 1) + (p + 3) \times 2 \times 2 \times 5$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors$ $\div 16 \uparrow \times 4$</p>
Transaction manager	$288 + 32 + 192 \times m \times 2 + 1028$ $+ (420 + 528 + 256 + 384 \times 2 + 128) \times m \times 2 + 7) + 256 \times 2$ <i>m</i> : Value of <code>pd_max_users</code>
Timer server	$32 \times (\text{value of } pd_max_users + 3) \times (1 + \text{number of utility servers in unit} + 1) + 1440$ Number of utility servers in unit is 26 + 12

15. Storage Requirements for HiRDB

Process item	Shared memory calculation formula (bytes)
Statistics log server	$384 + 128 \times 16 + 32 + 288 \times 2 + 1024 + 128 \times 3$ + value of <code>pd_stj_buff_size</code> $\times 1024 \times 3 + 64 + 4096 + 8192$
Process server	$160 + 512 \times a + 80 + 256 + (\text{value of } pd_max_server_process + 50) \times (256 + 144)$ $+ 16 + 1 \times 34 + 16$ <i>a</i> : For a single server: 112
System manager	$640 + (44 + 4) \times a \times 2 + (100 + 4) \times (b + 30 + 2) + (100 + 4) \times (c + 1) + 40 \times b \times 14 + 256 + 256 + 36 \times d + 12 \times e + 8 + 5844 + 212 + f + 16 + 1024$ <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> : $2052 + 128 \times (g + 3)$ <i>g</i> : For a single server: 91
Name server	169984
Node manager	$\uparrow (1152 + 288 \times \text{total number of units} + 80 \times \text{total number of servers} + 1536 + 800 + 56 \times \text{number of servers in unit} + 240 \times A + 44 \times A + 28 \times A + 16 \times B + 32)$ $\div 1024 \uparrow \times 1024$ <i>A</i> : <code>pd_utl_exec_mode</code> = 0: 1024 <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 1024, use 1024 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> : <code>pd_large_file_use</code> = N specified (or omitted): $3248 + (14+16) \times 808 + 1 \times 272 + 534 \times 272 + 16 \times 272 + \text{value of } pd_max_file_no \times 808$ <code>pd_large_file_use</code> = Y specified: $3248 + (14+16) \times 972 + 1 \times 276 + 534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972$

Process item	Shared memory calculation formula (bytes)
Log server	$32 + 48 + 128 \times 19 + 384 + 128 \times 7 + 1024 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of pd_log_rec_leng} \uparrow$ $\times \text{value of pd_log_rec_leng} + 64 + 4096 \times 2 + (256 + 480) \times B + 128$ $\times \text{value of pd_log_write_buff_count}$ $+ (\text{value of pd_log_write_buff_count} + A)$ $\times \uparrow \{\text{value of pd_log_max_data_size} + (68 + 44 + 96 + 160)\} \div 4096 \uparrow \times 4096$ $+ C + \uparrow (512 + 256 + 128 \times B + 464 \times B) \div (8192 - 128) \uparrow \times 8192$ <p><i>A</i>: 16 <i>B</i>: Number of pdlogadfg -d sys operands specified <i>C</i>: 512</p>
Synchronization point dump server	$\uparrow (368 + 1456 \times 2) \div 1024 \uparrow \times 1024$ $+ \uparrow \{(96 + 80 + 208 + 208) + 192 \times (\text{number of pdlogadfg -d spd operands specified})$ $+ 416 \times (\text{number of pdlogadpf -d spd operands specified}) + 1023\}$ $\div 1024 \uparrow \times 1024$
Common to all units	$a + b + 64 + (m + 3) \times c + 64 + 48 + d + e$ $+ (\text{value of pd_max_server_process} + 100) \times (48 + 16) + 32$ $+ (\text{value of pd_max_server_process} + 100 + 384) \times 32 + f + g + h$ $+ (\text{value of pd_max_server_process} + 127) \times 32 + 32$ <p><i>a</i>: 26560 <i>b</i>: 2988 <i>c</i>: 1952 <i>d</i>: 32×32 <i>e</i>: $64 + 64 \times \{(m + 3) \times 2 + \text{MAX}(5, (\downarrow m + 3] \div 10 \downarrow) + 7\}$ <i>f</i>: $512 \times (13 + 3) \times 2$ <i>g</i>: $\{(96 + \text{value of pd_lck_until_disconnect_cnt} \times 112 + 4095) \div 4096\} \times 4096 \times 2$ <i>h</i>: $\uparrow (\text{number of port numbers specified with pd_registered_port} \times 16 + 32 + 1023) \div 1024 \uparrow \times 1024$ If pd_registered_port is omitted: 0 <i>m</i>: Value of pd_max_users</p>

Process item	Shared memory calculation formula (bytes)
Transaction log server	$1024 + 512 \times A$ $+ \{$ $128 \times B + 128$ $+ [F + \uparrow 512 \div \text{value of pd_log_lec_leng} \uparrow \times \text{value of pd_log_rec_leng}$ $+ \uparrow (\text{value of pd_log_max_data_size} + 68 + 44 + 96 + 160) \div \text{value of}$ $\text{pd_log_lec_leng} \uparrow \times \text{value of pd_log_rec_leng}] \times D \times 2$ $+ E + (48 + 8) \times B \times 2$ $\} \times D$ $+ 584 \times B + 128 \times B + 64 \times B \times C + 128$ $+ \{$ $F + \uparrow 512 \div \text{value of pd_log_lec_leng} \uparrow \times \text{value of pd_log_rec_leng}$ $+ \uparrow (\text{value of pd_log_max_data_size} + 68 + 44 + 96 + 160)$ $\div \text{value of pd_log_lec_leng} \uparrow \times \text{value of pd_log_rec_leng}$ $\} \times D$ $+ E + (48 + 8) \times (B \times 2 + 2)$ <p> <i>A</i>: 2 <i>B</i>: 7 + value of pd_max_users \times 2 <i>C</i>: 1 <i>D</i>: Single server: 1 <i>E</i>: 512 <i>F</i>: 512 </p>
Status server	$\uparrow 64 \div 32 \uparrow \times 32$
Security audit	$\uparrow A \div 1024 \uparrow \times 1024$ <p> <i>A</i>: 560 if the pd_aud_file_name operand is omitted 456 + (304 \times 200) + <i>B</i> if the pd_aud_file_name operand is specified <i>B</i>: 0 if the pd_aud_async_buff_size operand value is 0 The following value if the pd_aud_async_buff_size operand value is 4096 or greater: (160 \times value of pd_aud_async_buff_count operand) +{(\uparrow value of pd_aud_async_buff_size operand \div 4096 \uparrow \times 4096) \times value of pd_aud_async_buff_count operand} + 4096 </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 is the sum of the process items listed as follows:

Process item	Shared memory calculation formula (bytes)
Scheduler	Value of <code>pd_utl_exec_mode</code> set to 0: $\{ \lceil (432 + 304 \times n) \div 1024 \rceil + 338 + x \} \times 1024$ Value of <code>pd_utl_exec_mode</code> set to 1: $\{ \lceil (432 + 304 \times n) \div 1024 \rceil + \lceil (m \times 2000 + 136) \div 1024 \rceil + y \} \times 1024$ $x: \text{Single server: } 116 + 5 \times (m + 3) + 14$ $y: \text{Single server: } 5 \times (m + 3) + 14$ $m: \text{Value of } \text{pd_max_users}$ $n: \text{Number of servers in unit} + \text{number of utility servers in unit} + 1$ $\text{Number of utility servers in the unit: } 26 + 12$
Lock server	$272 + 80 + c + d + 64 + 8192 + g + 80 + i + 64 + 8192 + 64 + n + 16 + t + u + 16$ $c: \text{When } \text{pd_lck_hash_entry} \text{ is omitted or 0 is specified:}$ $(\lfloor (8 + 8 \times \text{MAX}(((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2) + \text{value of } \text{pd_lck_pool_size} \times 4) \div 10,11261)) \div 16 \rfloor + 1) \times 16$ $\text{When 2 or a greater non-prime number is specified for } \text{pd_lck_hash_entry}:$ $(\lfloor (8 + 8 \times \text{largest prime number that is less than the value of } \text{pd_lck_hash_entry}) \div 16 \rfloor + 1) \times 16$ $\text{When 1 or a prime number is specified for } \text{pd_lck_hash_entry}:$ $(\lfloor (8 + 8 \times \text{value of } \text{pd_lck_hash_entry}) \div 16 \rfloor + 1) \times 16$ $d: ((p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4 + 5 \times 2) + \text{value of } \text{pd_lck_pool_size} \times 4) \times 128$ $g: \text{When value of } \text{pd_utl_exec_mode} = 1 \text{ and } p > 32:$ $((p + 3) \times 3 + p) \times 320$ $\text{When value of } \text{pd_utl_exec_mode} = 0 \text{ or } p \leq 32:$ $((p + 3) \times 3 + 32) \times 320$

Process item	Shared memory calculation formula (bytes)
	<p><i>i</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $\begin{aligned} & ((\text{value of } pd_lck_pool_size \times 5 + (p + 3)) \\ & \times (\text{value of } pd_max_access_tables + 4)) \times 2) \\ & + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow + p \times (\text{value of } pd_max_rdarea_no + 1) \\ & + (p + 3) \times 2 \times 2 \times 5) \\ & \text{rounded up to the next even value} \times 112 \end{aligned}$ <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $\begin{aligned} & ((\text{value of } pd_lck_pool_size \times 5 + (p + 3)) \\ & \times (\text{value of } pd_max_access_tables + 4)) \times 2) \\ & + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow + 32 \times (\text{value of } pd_max_rdarea_no + \\ & 1) \\ & + (p + 3) \times 2 \times 2 \times 5) \\ & \text{rounded up to the next even value} \times 112 \end{aligned}$ <p><i>n</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $(p + 3) \times 3 + p) \times 80$ <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $(p + 3) \times 3 + 32) \times 80$ <p><i>p</i>: value of <code>pd_max_users</code></p> <p><i>t</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $32 + ((p + 3) \times 3 + p) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $32 + ((p + 3) \times 3 + 32) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p><i>u</i>: When value of <code>pd_utl_exec_mode</code> = 1 and $p > 32$:</p> $\begin{aligned} & 32 + ((\text{value of } pd_lck_pool_size \times 5 + (p + 3)) \\ & \times (\text{value of } pd_max_access_tables + 4)) \times 2) \\ & + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow + p \times (\text{value of } pd_max_rdarea_no + 1) \\ & + (p + 3) \times 2 \times 2 \times 5) \\ & \text{rounded up to the next even value} \times \uparrow \text{value of } pd_max_open_holdable_cursors \\ & \div 16 \uparrow \times 4 \end{aligned}$ <p>When value of <code>pd_utl_exec_mode</code> = 0 or $p \leq 32$:</p> $\begin{aligned} & 32 + ((\text{value of } pd_lck_pool_size \times 5 + (p + 3)) \\ & \times (\text{value of } pd_max_access_tables + 4)) \times 2) \\ & + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow + 32 \times (\text{value of } pd_max_rdarea_no + \\ & 1) \\ & + (p + 3) \times 2 \times 2 \times 5) \\ & \text{rounded up to the next even value} \times \uparrow \text{value of } pd_max_open_holdable_cursors \\ & \div 16 \uparrow \times 4 \end{aligned}$

Process item	Shared memory calculation formula (bytes)
Transaction manager	$304 + 32 + 192 \cdot m^2 + 1048$ $+ (416 + 688 + 256 + 392 \cdot 2 + 128) \cdot (m^2 + 7) + 256 \cdot 2$ <i>m</i> : Value of <code>pd_max_users</code>
Timer server	$32 \times (\text{value of } pd_max_users + 3) \times (1 + \text{number of utility servers in unit} + 1) + 1440$ $+ (48 - 32) \times 2$ Number of utility servers in unit is 26 + 12
Statistics log server	$424 + 128 \times 16 + 32 + 288 \times 2 + 1168 + 144 \times 3$ $+ \text{value of } pd_stj_buff_size \times 1024 \times 3 + 64 + 4096 + 8192$
Process server	$176 + 528 \times a + 80 + 256 + (\text{value of } pd_max_server_process + 50) \times (256 + 160)$ $+ 16 + 1 \times 34 + 16$ <i>a</i> : For a single server: 112
System manager	$544 + 40 \times a \times 2 + 104 \times \{b + (26 + b$ $672 + (48 + 8) \times a \times 2 + (112 + 8) \times (b + 30 + 2) + (104 + 8) \times (c + 1) + 40 \times b \times$ $14 + 256 + 256 + 40 \times d + 16 \times e + 8 + 5864 + 236 + f + 16 + 1024$ <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> : $2056 + 128 \times (g + 3)$ <i>g</i> : For a single server: 91
Name server	169984
Node manager	$\uparrow (1312 + 320 \cdot \text{total number of units in HiRDB system}$ $+ 96 \cdot \text{total number of servers in HiRDB system} + 2048 + 960$ $+ 72 \cdot \text{number of HiRDB servers in the unit} + 240 \cdot A + 44 \cdot A + 28 \cdot A + 16 \cdot B + 48)$ $\div 1024 \uparrow 1024$ <i>A</i> : <code>pd_utl_exec_mode = 0</code> : 1024 <code>pd_utl_exec_mode = 1</code> and the unit contains a single server: value of <code>pd_max_users</code> $\times 10 + 400$ <code>pd_utl_exec_mode = 1</code> 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 1024, use 1024 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

15. Storage Requirements for HiRDB

Process item	Shared memory calculation formula (bytes)
I/O server	$\uparrow (56 + (\uparrow (56 + A) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128$ <p><i>A</i>: pd_large_file_use = N specified (or omitted): $3248 + (14 + 16) \times 808 + 1 \times 272 + 534 \times 272 + 16 \times 272 + \text{value of pd_max_file_no} \times 808 + (48 - 32) \times 3$</p> <p>pd_large_file_use = Y specified: $3248 + (14+16) \times 972 + 1 \times 276 + 534 \times 276 + 16 \times 276 + \text{value of pd_max_file_no} \times 972 + (48 - 32) \times 3$</p>
Log server	$32 + 48 + 128 \quad 19 + 432 + 128 \quad 7 + 1168 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of pd_log_rec_leng} \uparrow \text{value of pd_log_rec_leng}$ $+ 64 + 4096 \quad 2 + (256 + 480) \quad B$ $+ 144 \text{ value of pd_log_write_buff_count}$ $+ (\text{value of pd_log_write_buff_count} + A)$ $\uparrow \{\text{value of pd_log_max_data_size} + (68 + 44 + 96 + 160)\} \div 4096 \uparrow 4096$ $+ C + \uparrow (512 + 256 + 128 \quad B + 464 \quad B) \div (8192 - 128) \uparrow 8192$ <p><i>A</i>: 16 <i>B</i>: Number of pdlogadfg -d sys operands specified <i>C</i>: 512</p>
Synchronization point dump server	$\uparrow (384 + 1536 \times 2) \div 1024 \uparrow \times 1024$ $+ \uparrow \{(128 + 80 + 240 + 240) + 192 \times (\text{number of pdlogadfg -d spd operands specified})$ $+ 416 \times (\text{number of pdlogadpf -d spd operands specified}) + 1023\}$ $\div 1024 \uparrow \times 1024$

Process item	Shared memory calculation formula (bytes)
Common to all units	$a + b + 80 + (m + 3) \times c + 64 + 48 + d + e$ $+ (\text{value of } pd_max_server_process + 100) \times (64 + 16) + 32$ $+ (\text{value of } pd_max_server_process + 100 + 384) \times 32 + 32 + f + g + h$ $+ (\text{value of } pd_max_server_process + 127) \times 48 + 32$ $a: 35968$ $b: 3480$ $c: 2760$ $d: 48 \times 32$ $e: 80 + 96 \times \{(m + 3) \times 2$ $+ \text{MAX}(5, (\downarrow m + 3] \div 10 \downarrow) + 7\}$ $f: 512 \times (13 + 3) \times 2$ $g: \{(128 + \text{value of } pd_lck_until_disconnect_cnt \times 112 + 4095) \div 4096\} \times 4096$ $\times 2$ $h: \uparrow (\text{number of port numbers specified with } pd_registered_port \times 16 + 32 + 1023)$ $\div 1024 \uparrow \times 1024$ <p>If <i>pd_registered_port</i> is omitted: 0</p> $m: \text{Value of } pd_max_users$
Transaction log server	$1168 + 688 \times A$ $+ \{$ $128 \times B + 144$ $+ [G + \uparrow 256 \div \text{value of } pd_log_lec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $+ \uparrow (\text{value of } pd_log_max_data_size + 68 + 44 + 96 + 160) \div \text{value of}$ $pd_log_lec_leng \uparrow$ $\times \text{value of } pd_log_rec_leng] \times D \times 2$ $+ E + (48 + 8) \times B \times 2$ $\} \times D$ $+ 600 \times B + 128 \times B + 64 \times B \times C + 144$ $+ \{$ $G + \uparrow 256 \div \text{value of } pd_log_lec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $+ \uparrow (\text{value of } pd_log_max_data_size + 68 + 44 + 96 + 160)$ $\div \text{value of } pd_log_lec_leng \uparrow \times \text{value of } pd_log_rec_leng$ $\} \times D$ $+ E + (48 + 8) \times (B \times 2 + 2)$ $A: 2$ $B: 7 + \text{value of } pd_max_users \times 2$ $C: 1$ $D: \text{Single server: } 1$ $E: 512$ $G: 512$
Status server	$\uparrow 64 \div 32 \uparrow \times 32$

Process item	Shared memory calculation formula (bytes)
Security audit	$\uparrow A \div 1024 \uparrow \times 1024$ <i>A</i> : 624 if the <code>pd_aud_file_name</code> operand is omitted 520 + (320 × 200) + <i>B</i> if the <code>pd_aud_file_name</code> operand is specified <i>B</i> : 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 4096 or greater: (176 × value of <code>pd_aud_async_buff_count</code> operand) +{(\uparrow value of <code>pd_aud_async_buff_size</code> operand \div 4096 $\uparrow \times$ 4096) × value of <code>pd_aud_async_buff_count</code> operand}

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.

For 32-bit mode (KB):

$$\text{Formula 1} + \uparrow \{(\uparrow (40 + (\text{value obtained by adding Formulas 2 through 5})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

For 64-bit mode (KB):

$$\text{Formula 1} + \uparrow \{(\uparrow (72 + (\text{value obtained by adding Formulas 2 through 5})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

Notes on Formulas 1 to 6:

- If `Y` is specified in the `pd_rdarea_open_attribute_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`.
- If the `pd_dfw_await_process` operand is specified, add Formula 5.
- If you omit the `pd_sds_shmpool_size` operand, the following value is set:

For 32-bit mode:

$$\uparrow \{(\uparrow (40 + (\text{total of Formulas 2 through 5})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

For 64-bit mode:

$$\uparrow \{(\uparrow (72 + (\text{total of Formulas 2 through 5})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

- If you omit the `pd_max_commit_write_reclaim_no` operand (except cases in which `v6compatible` or `v7compatible` is specified in the

pd_sysdef_default_option operand), or you specify a value other than 0 in the pd_max_commit_write_reclaim_no operand, add Formula 6. However, if you have already added Formula 4, you do not need to add this formula.

The following table shows Formulas 1 through 6.

Formula	Shared memory calculation formula
Formula 1 (KB)	<p>32-bit mode</p> $ \begin{aligned} & 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 \\ & + \{ \\ & \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)\} + 32 \\ & \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 88 + \text{MAX}\{a \times (e + 3), f \div 8\} \times 24 \\ & \quad + \uparrow \{(q \div 4) + 7\} \div 64 \uparrow \times 8 \\ & \quad + \uparrow \{[(r - (s \times 592 + t \times 916 + u \times 172)) \div 2] + 7\} \div 64 \uparrow \times 8 \\ & \quad + \text{MAX}\{13 \times (e + 3), q \div 4\} \times 88 \\ & \quad + \text{MAX}\{21 \times (e + 3), [r - (s \times 592 + t \times 916 + u \times 172)] \div 2\} \times 60 \\ & \quad + 44 + 256 + 1024 + 512^{\#1} \\ & \quad \} \div 1024 + y + 7.5 \\ & + \uparrow \{248 \times v \times w + 47 \times v + 72\} \div 1024 \uparrow \\ & + \uparrow \{ \uparrow (28 + (\uparrow (32 + ((\uparrow g \div 127 \uparrow + 1) \times 2048 + 128))) \div 32 \uparrow \times 32) \} \\ & \quad \div 128 \uparrow \times 128 \\ & \quad \} \div 1024 \uparrow \\ & D \\ & + \sum_{i=1} (E_i) \end{aligned} $ <p>Add this if the pd_def_buf_control_area_assign operand is specified as INITIAL or if the operand is omitted.</p> $ + \{ [(a + 12) \div 13] \times 1.1 + [(a + 62) \div 63] + 3.7 \} \times (e + 7) $

Formula	Shared memory calculation formula
	<p>64-bit mode</p> $ \begin{aligned} & b \times 1.3 + c + d + f + 1.6 + q + 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 \\ & \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 + 1536 + 512^{\#1} \\ & \} \div 1024 + y + 7.5 \\ & + \uparrow \{248 \times v \times w + 64 \times v + 72\} \div 1024 \uparrow \\ & + \uparrow \{ \uparrow (56 + (\uparrow (56 + ((\uparrow g \div 127 \uparrow + 1) \times 2048 + 128))) \div 32 \uparrow \times 32) \} \\ & \quad \div 128 \uparrow \times 128 \\ & \} \div 1024 \\ & D \\ & + \sum_{i=1} (E_i) \\ & i=1 \end{aligned} $ <p>Add this if the <code>pd_def_buf_control_area_assign</code> operand is specified as <code>INITIAL</code> or if the operand is omitted.</p> $ + \{ [(a + 12) \div 13] \times 1.2 + [(a + 62) \div 63] \times 1.5 + 4.1 \} \times (e + 7) $
Formula 2 (bytes)	<p>32-bit mode</p> $ \begin{aligned} & 500 \times 1024 \\ & 5072 \times (e + 15) + (\uparrow 372 \times g \div 16 \uparrow \times 16) + 48^{\#1} \times g + 328 \times h \\ & + 112 \times (p + 240)^{\#5} \\ & + 96 \times x + 32 \times j + 132 \times \{19 + (e + 3) \times 3\} \\ & + 48 \times n + 48 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, (e + 3) \div 10)\} \\ & + 68 \times B + 144 \times A + 80 + 32 \times g + 64^{\#2} + 368^{\#3} \\ & + ((\downarrow (\uparrow (g \div 8) \uparrow + 3) \div 4 \downarrow) \times 4) \times j \end{aligned} $

Formula	Shared memory calculation formula
	64-bit mode 500×1024 $9416 \times (e + 15) + (\uparrow 472 \times g \div 16 \uparrow \times 16)$ $+ (\uparrow 56^{\#1} \times g \div 16 \uparrow \times 16) + 344 \times h$ $+ (\uparrow 136 \times (p + 240) \div 16 \uparrow \times 16)^{\#6}$ $+ 144 \times x + 48 \times j + 240 \times \{19 + (e + 3) \times 3\}$ $+ 64 \times n + 96 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, (e + 3) \div 10)\}$ $+ 68 \times B + 160 \times A + 96 + 48 \times g + 64^{\#2} + 448^{\#3}$ $+ ((\downarrow (\uparrow (g \div 8) \uparrow + 7) \div 8 \downarrow) \times 8) \times j$
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)	32-bit mode $88 + 52 \times C + 3400$ 64-bit mode $112 + 56 \times C + 7200$
Formula 6 (bytes)	32-bit mode $(32 + 16 \times q) \times F + 16$ 64-bit mode $(48 + 32 \times q) \times F + 16$

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
- i*: Total number of indexes in the server (for a partitioning key index, this is the number of partitions in the server)
- 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^{#4}
- u*: Total number of parameters for the plug-in functions used with DML^{#4}
- 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
- 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_await_process` operand
- D*: Total number of specified `pdplgprm` operands
- E_i*: Size of the shared memory specified by the *i*th `pdplgprm` operand.
- F*: Value of the `pd_max_commit_write_reclaim_no` operand
- #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_await_process` operand.
- #3: Add this value if the facility for predicting reorganization time is used.

#4: 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')
```

#5: When you specify `v6compatible` or `v7compatible` in the `pd_sysdef_default_option` operand, use $112 \times \text{MAX}(p, i \times 1.2)$ for calculating instead of $112 \times (p + 240)$.

#6: When you specify `v6compatible` or `v7compatible` in the `pd_sysdef_default_option` operand, use $(\uparrow 136 \times \text{MAX}(p, (i \times 1.2)) \div 16 \uparrow \times 16)$ for calculating instead of $(\uparrow 136 \times (p + 240) \div 16 \uparrow \times 16)$.

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 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> $\sum_{i=1}^n \{ \uparrow \{ 672 + 64 + (280 + 56^{#1}) \times (P_i + 4) + (124 + 80^{#2} + 96 \times A \times M_i) \times U_i \} \div 4096 \uparrow \times 4096 + S_i \times \{ P_i + 4 + (U_i \times M_i \times A) \} \} \div 1024$ <p>64-bit mode</p> $\sum_{i=1}^n \{ \uparrow \{ 864 + 64 + (464 + 96^{#1}) \times (P_i + 4) + (176 + 96^{#2} + 136 \times A \times M_i) \times U_i \} \div 4096 \uparrow \times 4096 + S_i \times \{ P_i + 4 + (U_i \times M_i \times A) \} \} \div 1024$
Formula 2	<p>32-bit mode</p> $\{ \uparrow [(s \times 1024 \div 4) \div 8 + 112] \div 4096 \uparrow \times 4096 \times \uparrow a \div (s \times 1024) \uparrow \} \div 1024$ <p>64-bit mode</p> $\{ \uparrow [(s \times 1024 \div 4) \div 8 + 144] \div 4096 \uparrow \times 4096 \times \uparrow a \div (s \times 1024) \uparrow \} \div 1024$

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 memory on the basis of the value of `PDAGGR` in the client environment definition. The size of the memory can be obtained from the following formula (in bytes).

Formula:

$$e + \uparrow d \div 4 \uparrow \times 4 + \uparrow (17 + 4 \times a + 4 \times b + c + d) \div 4 \uparrow \times 4 \times (N + 1) \quad \text{(bytes)}$$

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)

d: Length of operation area for set functions (see Table 15-4)

e: 32-bit mode: $\text{MAX}(4 \times N \times 24, 16408)$

64-bit mode: $\text{MAX}(8 \times N \times 40, 32808)$

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 (<i>p</i> , <i>s</i>)	$\uparrow(p + 1) \div 2 \uparrow$	$\uparrow(p + 7) \div 2 \uparrow^3$
FLOAT	8	10

Column's data type	Column length	Length of operation area for set function ¹
SMALLFLT	4	6
INTERVAL YEAR TO DAY	5	8
INTERVAL HOUR TO SECOND	4	6
CHAR (<i>n</i>)	<i>n</i>	<i>n</i> + 3
VARCHAR (<i>n</i>)	<i>n</i> + 2	<i>n</i> + 5
NCHAR (<i>n</i>)	2 × <i>n</i>	2 × <i>n</i> + 2
NVARCHAR (<i>n</i>)	2 × <i>n</i> + 2	2 × <i>n</i> + 4
MCHAR (<i>n</i>)	<i>n</i>	<i>n</i> + 3
MVARCHAR (<i>n</i>)	<i>n</i> + 2	<i>n</i> + 5
DATE	4	6
TIME	3	6
BLOB (<i>n</i>)	—	
BINARY (<i>n</i>)	<i>n</i> + 2	<i>n</i> + 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 operation area is 18.

(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

<p>32-bit mode</p> <p>a</p> $\sum_{i=1} (13 \times 1024 + 6 \times 1024 \times b + c)$ <p>64-bit mode</p> <p>a</p> $\sum_{i=1} (13 \times 1024 + 7 \times 1024 \times b + c)$ <p>(bytes)</p>
--

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 manual *HiRDB Version 8 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
1500 or less	Batch hash join	0.5
1500 × (packet split count ÷ 3) or less	Packet split hash join	1-level packet split
1500 × (packet split count ÷ 3) ² or less		2-level packet split
Greater than 1500 × (packet split count ÷ 3) ²		3-level packet split

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: MIN { ↓(size of hash table ÷ 2) ÷ page length of hash table ↓, 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 1012	4096
1013 to 2036	8192
2037 to 4084	16384
4085 to 16360	32768
16361 to 32720	$\lceil (\text{maximum length of hash table row} + 48) \div 2048 \rceil \times 2048$

c: Maximum length of hash table row

For details about the length of a hash table row, see the manual *HiRDB Version 8 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

$a \times 2$ (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 18. *Determining Work Table File Size* because a work table is created.

Formula

$$\{100 + (a + 2) \times (\text{number of offset rows} + \text{number of rows specified in the } \text{LIMIT} \text{ clause})\} \times b$$

(bytes)

a : Row length

The row length cannot exceed 32,720 bytes. The row length is calculated with the following formula:

$$\sum_{i=1}^m (A_i) + 2 \times m + 4 + c$$

(bytes)

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:

$$1 + \text{number of UNION [ALL] clause specifications}$$

(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

$$a \times 500 + (20 + 6) \times 800 + 16 \quad (\text{bytes})$$

a : Row length. To determine the row length, use the following formula:

$$\sum_{i=1}^m (A_i) + 4 \times (m + 2) + 12 + 4 + 8$$

(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.

A_i : 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, A_i 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

$$(16 + 16 \times m) + a$$

64-bit mode

$$(16 + 24 \times m) + a$$

(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

Table 15-5 shows the data length 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)
--

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

$$\begin{aligned} & \uparrow \{ \\ & 2586 + S_i \times 60 + P_i \times 20 + T_i \times 1424 + C_i \times T_i \times 72 + W_i \times 776 + T_i \times W_i \times 72 \\ & + K_i \times 276 + K_i \times T_i \times 72 + L_i \times 3 + L_i \times T_i + D_i \times T_i \times 134 + A_{ri} \times 108 \\ & + G_i \times 44 + S_{li} \times 40 + U_{pi} \times 110 \\ & + F_i \times 90 + T_i \times C_{wi} \times 48 \\ & + \text{MAX}(P_i, W_{pi}) \times 60 \\ & \} \times 1.2 \div 1024 \uparrow \end{aligned} \quad (\text{KB})$$

S_i : Number of items to be retrieved in SQL statements

P_i : Number of embedded variables, ? parameters, or SQL parameters in SQL statements

T_i : Number of table names in SQL statements

C_i : Number of column names in SQL statements

W_i : Number of predicates used in Boolean operators (AND and OR) in SQL statements

K_i : Number of literals in SQL statements

L_i : Total length of literals in SQL statements (bytes)

D_i : Total number of storage conditions defined in SQL statements

A_{ri} : Number of arithmetic operations and concatenation operations in SQL statements

G_i : Number of columns specified in GROUP BY clause of SQL statements

O_{ri} : Number of column specification or sort item specification numbers in ORDER BY clause of SQL statements

F_i : Total number of set functions and scalar functions in SQL statements

S_{li} : Number of queries specified in SQL statements

U_{pi} : Number of columns to be updated in SQL statements

C_{wi} : Number of WHENs in CASE expression of SQL statements

W_{pi} : 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 or not `SELECT_APSL` is applied. For details about the access path display utility (`pdvwopt`), see the manual *HiRDB Version 8 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 8 System Definition*.

15.1.8 Formula for size of memory required during BLOB data retrieval or updating (HiRDB/Single Server)

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:

$$\begin{aligned} &\uparrow \{ \\ &\quad c \\ &\quad \sum_{i=1}^c (\text{actual length of BLOB input variable } i^1 + 118) + \\ &\quad d \\ &\quad \sum_{j=1}^d (\text{specified length of BLOB output variable } j^2 + 86) \\ &\} \div 1024 \uparrow \end{aligned}$$

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:

$$\uparrow \left\{ \begin{array}{l} e \\ \sum_{i=1}^c \left\{ \begin{array}{l} d \\ \sum_{j=1}^e (\text{defined length of BLOB output variable } j + 18) \end{array} \right\} \end{array} \right\} \div 1024 \uparrow$$

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>PDBLKF</code> operand is omitted or is set to 1	—	Formula 2
	<code>PDBLKF</code> operand is set to 2 or greater	Formula 1	

Legend:

— : Not applicable

Formula 1

$$\uparrow \{864 + 16 \times a + (6 \times a + 2 \times d + b) \times c\} \div 1024 \uparrow$$

(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

$$\text{MAX}(X_1, X_2)$$

(kilobytes)

X_1 : $\uparrow (864 + 22 \times a + 2 \times c + b) \div 1024 \uparrow$

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

15.2 Estimating the memory size required for a HiRDB/Parallel Server

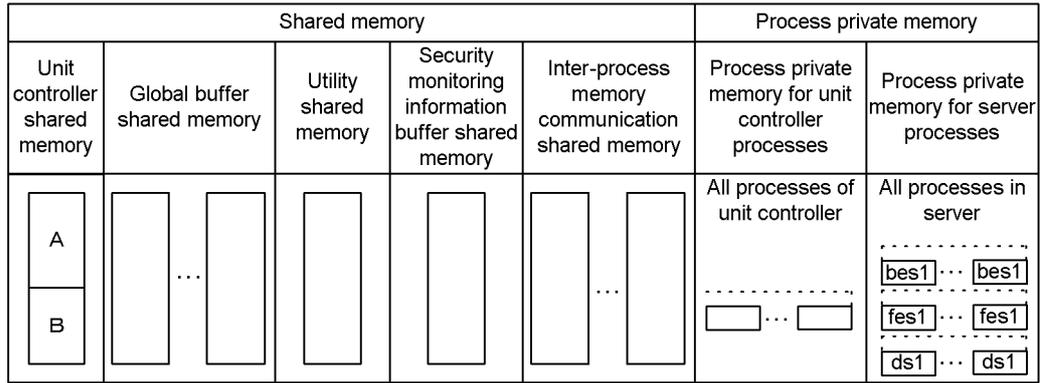
This section explains how to estimate the size of the memory required for each unit constituting a HiRDB/Parallel Server. The topics covered include:

- 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

Figure 15-2 shows the memory allocation for each unit of a HiRDB/Parallel Server.

Figure 15-2: Memory allocation for each unit of a HiRDB/Parallel Server



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

Table 15-6 provides the details of the shared memories for a unit of a HiRDB/Parallel Server.

Table 15-6: HiRDB/Parallel Server 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
Purpose	System control	Global buffers	Communication between the unit controller and utilities	Security audit information buffer	Client-server inter-process communication
Processes	All HiRDB processes	Back-end servers, dictionary servers	Utility processes	Front-end servers	Front-end servers, client processes

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
Number of segments	1	<ul style="list-style-type: none"> When the global buffer dynamic update facility is not used: 1-16^{#1} When the global buffer dynamic update facility is used: 32-bit mode: 1-516^{#1} 64-bit mode: 1-1,016^{#1} 	1	1	Number of clients connected using the PDIPC=MEMORY environment variable (0-2000) × 2
Maximum value per segment	See Table 15-7 <i>Size of memory required for each unit of a HiRDB/Parallel Server</i> . It may not be possible to allocate this size depending on the OS environment.	Divide the segment by the SHMMAX operand value. It may not be possible to allocate this size depending on the OS environment.	See Table 15-7 <i>Size of memory required for each unit of a HiRDB/Parallel Server</i> . It may not be possible to allocate this size depending on the OS environment.	See Table 15-7 <i>Size of memory required for each unit of a HiRDB/Parallel Server</i> . It may not be possible to allocate this size depending on the OS environment.	See Table 15-7 <i>Size of memory required for each unit of a HiRDB/Parallel Server</i> . It may not be possible to allocate this size depending on the OS environment.
Allocation conditions	None	There must be a global buffer definition	Specify <code>pd_utl_exec_mode=1</code>	Specify the <code>pd_aud_file_name</code> operand as the HiRDB file system area name for the audit trail.	There are clients connected using the PDIPC=MEMORY environment variable

15. Storage Requirements for HiRDB

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
Creation timing	At unit activation (including standby unit activation when the rapid system switchover facility is used)	<ul style="list-style-type: none"> At server activation (including standby unit activation when the rapid system switchover facility is used) When <code>pdbufmod -k {add upd}</code> is executed 	When utilities are executed	When a front-end server starts up	When client and server are connected
Deletion timing	At next unit activation (including standby unit activation when 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
Indication by <code>pdls -d mem</code>	Indicated	Indicated	Indicated	Indicated	Not indicated
SHM-OWNER of <code>pdls -d mem</code>	MANAGER	Server name	UTILITY	AUDEF	Not indicated

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
Related operands	<ul style="list-style-type: none"> pd_dic_shmpool_size pd_bes_shmpool_size 	<ul style="list-style-type: none"> pd_dbbuff_modify pdbuffer SHMMAX 	<ul style="list-style-type: none"> pd_utl_exec_mode 	<ul style="list-style-type: none"> Operands related to the security audit facility^{#2} 	<ul style="list-style-type: none"> PDIPC PDSENDMEMSIZE PDRECVMEMSIZE
Remarks	—	—	Can be created only when pd_utl_exec_mode=1 (when pd_utl_exec_mode=0, 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 8 System Definition*.

15.2.2 Calculation of required memory

The size of the memory required for each unit of a HiRDB/Parallel Server is the sum of the items listed in Table 15-7.

Table 15-7: Size of memory required for each unit of a HiRDB/Parallel Server

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 $J + K \times \text{Number of FESs in unit} + L \times (\text{Number of BESs in unit} + \text{Number of DSs in unit}) + \uparrow \{(64 + 48 \times (v + 1)) \times (\text{value of } pd_max_server_process - w) + z\} \div 1024 \uparrow$ 64-bit mode $J + K \times \text{Number of FESs in unit} + L \times (\text{Number of BESs in unit} + \text{Number of DSs in unit}) + \uparrow \{(64 + 64 \times (v + 1)) \times (\text{value of } pd_max_server_process - w) + z\} \div 1024 \uparrow$ If a plug-in is used, add: + 1400 If the asynchronous READ facility is used, add: + <i>s</i> If <i>fixed</i> is specified in the <i>pd_process_terminator</i> operand, add: + <i>M</i> × (value of <i>pd_process_terminator_max</i> - 1) 			
	Process private area used by each server process ^{1, 2}	Front-end server	$(N + h + m + p + q) \times (b + 3) + 100 + y$	
		Dictionary server	<i>pd_work_buff_mode</i> =each specified	$\{(P + i + m + r + t) \times (b + 3)\} + (a + 9) \times 2 + 100 + y$
			<i>pd_work_buff_mode</i> =pool specified or omitted	$\{(P + i + m + r + t) \times (b + 3)\} + a + 9 + \uparrow a \div 128 \times 0.1 \uparrow + 100 + n + y$
Back-end server		<i>pd_work_buff_mode</i> =each specified	$\{Q + g + (a + 9) \times c + i + m + r + t\} \times (b + 3) + 100 + y$	
	<i>pd_work_buff_mode</i> =pool specified or omitted	<ul style="list-style-type: none"> 32-bit mode $(Q + g + a + 9 + \uparrow a \div 128 \times 0.1 + 11 \uparrow + i + m + r + t) \times (b + 3) + 100 + n + y$ 64-bit mode $(Q + g + a + 9 + \uparrow a \div 128 \times 0.1 + 15 \uparrow + i + r + t) \times (b + 3) + 100 + n + y$ 		

Item		Required memory (KB)
Shared memory	Space used by the unit controller in the unit controller shared memory	$\uparrow d \div 1024 \uparrow$
	Space used by each server in the unit controller shared memory ¹	e
	Global buffer shared memory	f
	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): $\uparrow 0.3 + R \times 0.25 \uparrow$</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 If plug-ins are used, add 300 per 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 manual *HiRDB Version 8 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

- For a dictionary server, the value is value of the `pd_max_dic_process` operand.
- For a back-end server, the value is value of the `pd_max_bes_process` 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.

c: Maximum number of work tables

From Table 15-8, obtain the number of work tables per SQL statement. Use the

largest number of work tables obtained from Table 15-8 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 and version. For details about the memory requirement for your Java virtual machine, see the applicable manual.

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) } × 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 manual *HiRDB Version 8 UAP Development Guide*.

- Work table extended buffer size based on the increase in the number of work tables = MIN{ (number of work tables used × 128 - value of the `pd_work_buff_size` operand), (value of `pd_work_buff_expand_limit` operand - value of `pd_work_buff_size` operand) } × (number of users such that the number of work tables is greater than the value of the `pd_work_buff_size` operand ÷ 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:

$$(90 + 90 \sum_{i=1} \text{Memory used by the RDAREA for management of the HiRDB file system area}) \times \text{value of pd_max_ard_process}$$

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 following formula based on the initial settings:

$$\{(\text{Number of files}^1 + \text{number of extensions}^2) \div 64\} \times 1.5^3$$

¹ Value specified by `pdfmkfs -l`.

² Value specified by `pdfmkfs -e`.

³ Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2048.

t: HiRDB file system memory size

Determine with the following formula (in kilobytes):

$$347 + \text{Memory used by the work tables for management of the HiRDB file system area} + \text{Memory used by the system logs for management of the HiRDB file system area} + 90 \sum_{i=1} \text{memory used by the RDAREA for management of the HiRDB file system area}$$

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 following formula based on the initial settings:

$$\{(\text{Number of files}^1 + \text{number of extensions}^2) \div 64\} \times 1.5^3$$

¹ Value specified by `pdfmkfs -l`.

² Value specified by `pdfmkfs -e`.

³ Multiply when the area size (value specified in `pdfmkfs -n`) is at least 2048.

u: When value of `pd_utl_exec_mode` is 0: 0

When value of `pd_utl_exec_mode` is 1: $\uparrow \{(b \times 2000 + 136) \div 1024\} \uparrow \times 1024$

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 8 System Definition*.

y: Sum of the values obtained by the following formula for each server process in the unit:

In the 32-bit mode:

$\uparrow \{(64 + 48 \times (\text{value of } \text{pd_module_trace_max} + 1)) \times (\text{maximum number of processes that can be started} + 3)\} \div 1024 \uparrow$

In the 64-bit mode:

$\uparrow \{(64 + 64 \times (\text{value of } \text{pd_module_trace_max} + 1)) \times (\text{maximum number of processes that can be started} + 3)\} \div 1024 \uparrow$

For details about the maximum number of processes that can be started, see the manual *HiRDB Version 8 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):

$(D + E + F) \times \text{number of dictionary servers} + (D + E + F + G) \times \text{number of back-end servers} + \Sigma H$
--

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 $291722 + 4 \times \text{value of pd_max_rdarea_no}$ $+ \{48 \times (\text{value of pd_max_rdarea_no} + \text{number of tables}) + 304\} \times (\text{value of pd_max_users} \times 2 + 7)$ • 64-bit mode $370682 + 8 \times \text{value of pd_max_rdarea_no}$ $+ \{64 \times (\text{value of pd_max_rdarea_no} + \text{number of tables}) + 512\} \times (\text{value of pd_max_users} \times 2 + 7)$ <p>Number of tables: Number of data dictionary tables + MAX {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 < 4096 $\text{MAX}((\lfloor 3400 \div ((\lfloor (\text{record length} - 40) - 308) \div 20 \rfloor))$ $+ (\lfloor (\text{record length} - 40) \div 20 \rfloor) \times (\text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) - 1))$ $+ 0.7) \rfloor, 1) \times \text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) \times (\text{record length} - 40)$ When $4096 \leq \text{record length of server status file} < 12288$ $\text{MAX}(\lfloor 3400 \div (\lfloor (((\text{record length} - 40) - 308) \div 20) \rfloor) + 0.7) \rfloor, 1)$ $\times (\text{record length} - 40)$ When $12288 \leq \text{record length of server status file}$ $\text{MAX}(\lfloor 3400 \div (\lfloor (((\text{record length} - 40) - 836) \div 20) \rfloor) + 0.7) \rfloor, 1)$ $\times (\text{record length} - 40)$</p> <p><i>X</i>: When the number of RDAREAS in server ≤ 3400: 1 When $3401 \leq \text{number of RDAREAS in server} \leq 6800$: 2 When $6801 \leq \text{number of RDAREAS in server}$: 3</p> <p><i>b2</i>: When the record length of the status file for server < 4096 $(\lfloor 5662310 \div ((\lfloor (\text{record length} - 40) - 308) \div 20 \rfloor))$ $+ (\lfloor (\text{record length} - 40) \div 20 \rfloor) \times (\text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) - 1))$ $+ 0.7) \rfloor) \times \text{MAX}(\lfloor 4096 \div \text{record length} \rfloor, 2) \times (\text{record length} - 40)$ When $4096 \leq \text{record length of server status file} < 12288$ $\lfloor 5662310 \div (\lfloor (((\text{record length} - 40) - 308) \div 20) \rfloor) + 0.7) \rfloor$ $\times (\text{record length} - 40)$ When $12288 \leq \text{record length of server status file}$ $\lfloor 5662310 \div (\lfloor (((\text{record length} - 40) - 836) \div 20) \rfloor) + 0.7) \rfloor$ $\times (\text{record length} - 40)$</p> <p><i>Y</i>: When the number of RDAREAS in server ≤ 10200: 0 When $10201 \leq \text{number of RDAREAS in server} \leq 5672510$: 1 When $5672511 \leq \text{number of RDAREAS in server} \leq 11334820$: 2 When $11334821 \leq \text{number of RDAREAS in server}$: 3</p>

Variable	Value
<i>F</i>	If <code>commit</code> is specified in the <code>pd_dbsync_point</code> operand, add: $+ 112 \times (\text{value of } pd_max_users \times 2 + 7)$
<i>H</i>	For a back-end server, if the number of RDAREA storage areas created in a HiRDB file system area using the raw I/O facility is 1001 or greater in the server, add: <ul style="list-style-type: none"> 32-bit mode $12012 \times (\uparrow(\text{number of RDAREA storage areas created in a HiRDB file system area using the raw I/O facility} - 1000) \div 1000 \uparrow)$ 64-bit mode $16016 \times (\uparrow(\text{number of RDAREA storage areas created in a HiRDB file system area using the raw I/O facility} - 1000) \div 1000 \uparrow)$

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
Windows (32-bit mode)	68,100	11,200	14,400	3,100	4,500	3,900	3,800
Windows Server 2003 (IPF)	183,200	17,900	23,200	5,300	11,600	12,900	15,000
Windows (x64)	85,300	14,400	18,900	4,300	10,400	10,800	11,100

R: The number of objects specified in a narrowed search using the security audit facility audit trail

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 INSERT (-SELECT) statement	When none of 1-8 as follows are applicable: 0 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) × 2 + 1 2. When specifying the ORDER BY clause Number of additional work tables = 2 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) × 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

15.2.3 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: $\{ \lceil (432 + 304 \times n) \div 1024 \rceil + 289 + x + z \} \times 1024$</p> <p>Value of <code>pd_utl_exec_mode</code> set to 1: $\{ \lceil (432 + 304 \times n) \div 1024 \rceil + \lceil (s \times 2000 + 136) \div 1024 \rceil + y + z \} \times 1024$</p> <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): $\lceil 64 + \{(\text{Number of BESs in unit} + \text{number of acceptable BESs}) \times 48\} \div 1024 \rceil$ Other unit: 0</p> <p><i>n</i>: Number of servers in unit + $\beta + \gamma$ + number of utility servers in unit + 1 α: If the unit controller has a MGR: 2 If the unit controller contains a FES: 3 If the unit controller contains a DS: 7 If the unit controller contains BESs: $(\text{number of BESs} + \beta) \times 15$</p> <p><i>s</i>: Value of <code>pd_max_users</code> <i>t</i>: Value of <code>pd_max_dic_process</code> <i>u</i>: Value of <code>pd_max_bes_process</code></p> <p>β: Unit subject to standby-less system switchover (effects distributed): Number of acceptable BESs Other unit: 0</p> <p>γ: Unit subject to standby-less system switchover (1:1): Number of alternate BESs Other unit: 0</p>

Process item	Shared memory calculation formula (bytes)
Lock server	<p>• When there is a server (FES, BES, or DS) in the unit: 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_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 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>. In a unit subject to standby-less system switchover (1:1), <i>servers in the unit</i> means <i>all accepted BESs + all alternate BESs</i>.</p> $y \sum_{x=1} \{ 192 + 48 + c_x + d_x + 48 + 4096 + gx + 48 + i_x + 48 + 12252 + 48 + n_x + p_x + t_x + u_x + 16 \}$ <p>x: Server serial number in the unit y: Number of servers in the unit</p> <p>c_x: When <code>pd_lck_hash_entry</code> is omitted or 0 is specified: For FES with <code>pd_fes_lck_pool_size</code> omitted: $(\downarrow(8 + 4 \times \text{MAX}(\uparrow(p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4) \div 10 \uparrow, 11261)) \div 16 \downarrow + 1) \times 16$ For FES with <code>pd_fes_lck_pool_size</code> specified: $(\downarrow(8 + 4 \times \text{MAX}(\uparrow(\text{value of } \text{pd_fes_lck_pool_size} \times 6) \div 10 \uparrow, 11261)) \div 16 \downarrow + 1) \times 16$ For BES or DS: $(\downarrow(8 + 4 \times \text{MAX}(\uparrow((p + 3) \times 2 \times 5 + \text{value of } \text{pd_lck_pool_size} \times 6) \div 10 \uparrow, 11261)) \div 16 \downarrow + 1) \times 16$ When 2 or a greater non-prime number is specified for <code>pd_lck_hash_entry</code>: $(\downarrow(8 + 4 \times \text{largest prime number that is less than the value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$ When 1 or a prime number is specified for <code>pd_lck_hash_entry</code>: $(\downarrow(8 + 4 \times \text{value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$</p>

Process item	Shared memory calculation formula (bytes)
	<p>d_x: For FES with <code>pd_fes_lck_pool_size</code> omitted: $((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 96$ For FES with <code>pd_fes_lck_pool_size</code> specified: $\text{Value of } pd_fes_lck_pool_size \times 6 \times 96$ For BES or DS: $((p + 3) \times 2 \times 5 + \text{value of } pd_lck_pool_size \times 6) \times 96$</p> <p>$g_x$: For FES: $(p + 3) \times 2 \times 256$ For BES with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 32$: $((p + 3) \times 2 + s) \times 256$ For BES with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 32$: $((p + 3) \times 2 + 32) \times 256$ For DS with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 16$: $((p + 3) \times 2 + s) \times 256$ For DS with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 16$: $((p + 3) \times 2 + 16) \times 256$</p> <p>$i_x$: For FES with <code>pd_fes_lck_pool_size</code> omitted: $((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 2 \times 64$ For FES with <code>pd_fes_lck_pool_size</code> specified: $(\text{value of } pd_fes_lck_pool_size \times 8)$ rounded up to the next even value $\times 64$ For BES with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 32$: $(\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + s$ $\times (\text{value of } pd_max_rdarea_no + 1))$ rounded up to the next even value $\times 64$ For BES with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 32$: $(\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + 32$ $\times (\text{value of } pd_max_rdarea_no + 1))$ rounded up to the next even value $\times 64$ For DS with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 16$: $(\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + s + 4)$ rounded up to the next even value $\times 64$ For DS with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 16$: $(\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + 20)$ rounded up to the next even value $\times 64$</p>

Process item	Shared memory calculation formula (bytes)
	<p>n_x: For FES: $(p + 3) \times 2 \times 48$ For BES with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 32$: $((p + 3) \times 2 \times 17 + s) \times 48$ For BES with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 32$: $((p + 3) \times 2 \times 17 + 32) \times 48$ For DS with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 16$: $((p + 3) \times 2 \times 17 + s) \times 48$ For DS with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 16$: $((p + 3) \times 2 \times 17 + 16) \times 48$</p> <p>$p_x$: For FES with the number of FESs in the HiRDB system > 1: $s + 1$ For FES with the number of FESs in the HiRDB system = 1: s For BES with $s >$ value of <code>pd_max_bes_process</code>: s For BES with $s \leq$ value of <code>pd_max_bes_process</code>: Value of <code>pd_max_bes_process</code> For DS with $s >$ value of <code>pd_max_dic_process</code>: s For DS with $s \leq$ value of <code>pd_max_dic_process</code>: Value of <code>pd_max_dic_process</code></p> <p>s: value of <code>pd_max_users</code></p> <p>t_x: For FES: $32 + (p + 3) \times 2 \times \uparrow \text{value of } \text{pd_max_open_holdable_cursors} \div 16 \uparrow \times 4$ For BES with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 32$: $32 + ((p + 3) \times 2 + s) \times \uparrow \text{value of } \text{pd_max_open_holdable_cursors} \div 16 \uparrow \times 4$ For BES with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 32$: $32 + ((p + 3) \times 2 + 32) \times \uparrow \text{value of } \text{pd_max_open_holdable_cursors} \div 16 \uparrow \times 4$</p>

Process item	Shared memory calculation formula (bytes)
	<p>For DS with the value of <code>pd_utl_exec_mode = 1</code> and $s > 16$:</p> $32 + ((p + 3) \times 2 + s) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For DS with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 16$:</p> $32 + ((p + 3) \times 2 + 16) \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>u_x: For FES with <code>pd_fes_lck_pool_size</code> omitted:</p> $32 + ((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 2 \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For FES with <code>pd_fes_lck_pool_size</code> specified:</p> $32 + (\text{value of } pd_fes_lck_pool_size \times 8) \text{ rounded up to the next even value} \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For BES with the value of <code>pd_utl_exec_mode = 1</code> and $s > 32$:</p> $32 + (\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + s \times (\text{value of } pd_max_rdarea_no + 1)) \text{ rounded up to the next even value} \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For BES with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 32$:</p> $32 + (\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + 32 \times (\text{value of } pd_max_rdarea_no + 1)) \text{ rounded up to the next even value} \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For DS with the value of <code>pd_utl_exec_mode = 1</code> and $s > 16$:</p> $32 + (\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + s + 4) \text{ rounded up to the next even value} \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <p>For DS with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 16$:</p> $32 + (\text{value of } pd_lck_pool_size \times 8 + (p + 3) \times 2 \times 2 \times 5 + 20) \text{ rounded up to the next even value} \times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$ <ul style="list-style-type: none"> When there is no server (FES, BES, or DS) in the unit <p>8416</p>

Process item	Shared memory calculation formula (bytes)
Transaction manager	$288 + 32 \times B + 192 \times s \times 2$ <p>If the unit contains FES, add the following value:*</p> $+ 1028 + (420 + 528 + 256 + 384 \times 2) \times (A \times 2 + 7) + 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ C$ <p>If the unit contains BES, add the following value:*</p> $+ 1028 + (420 + 528 + 256 + 384 \times 2)$ $\times (u \times 2 + 7) + 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ D$ <p>If the unit contains DS, add the following value:*</p> $+ 1028 + (420 + 528 + 256 + 384 \times 2)$ $\times (t \times 2 + 7) + 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ E$ <p><i>s</i>: Value of <code>pd_max_users</code> <i>t</i>: Value of <code>pd_max_dic_process</code> <i>u</i>: Value of <code>pd_max_bes_process</code> <i>A</i>: For a multi-FES system: $s + 1$ If not a multi-FES system: s <i>B</i>: Unit subject to standby-less system switchover (effects distributed): Number of host BESs + <code>pd_ha_max_act_guest_servers</code> operand correction value Other unit: Number of servers in the unit <i>C</i>: If the unit meets one condition in <i>Conditions</i> below: $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ If the unit does not match both of the conditions in <i>Conditions</i> below: 0 <i>D</i>: If the unit meets one condition in <i>Conditions</i> below: $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (u \times 2 + 7)$</p>

Process item	Shared memory calculation formula (bytes)
	<p>If the unit does not meet both conditions in <i>Conditions</i> below: 0 <i>E</i>: 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 <i>Conditions</i></p> <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. <p>* Add the value obtained from the above formula as many times as the value of <i>B</i>. For units subject to standby-less system switchover (1:1): $\text{Number of accepted BESs} + \text{Number of alternate BESs}$ Other than the above, the number of the servers in the unit.</p>
Timer server	$32 \times (\text{value of } pd_max_users + 3)$ $\times (\text{number of BESs in system} + 1 + \text{number of utility servers in unit} + 1)$ $+ 1440$ 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 When there is DS in the unit: 7 When there is BES in the unit: Number of BESs $\times 15$
Statistics log server	$384 + 128 \times 16 + 32 + 288 \times 2 + 1024 + 128 \times 3$ $+ \text{value of } pd_stj_buff_size \times 1024 \times 3 + 64 + 4096 + 8192$
Process server	$160 + 512 \times \text{MAX}(c, 256) + 80 + 256 + (\text{value of } pd_max_server_process + 50) \times (256 + 144) + 16 + 1 \times 34 + 16$ $c: \lceil (45 + d + e + f + (g \times \text{number of BESs in the unit}^*) + h + i) \div 16 \rceil \times 16$ d : If the unit contains MGR, 46; if not, j . e : If the unit contains DS, 15; if not, 0. f : If the unit contains FES, 9; if not, 0. g : If the unit contains BES, 24; 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 manager is specified in the <code>pd_mlg_msg_log_unit</code> operand, 1; if <code>local</code> is specified, 2. * 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.

Process item	Shared memory calculation formula (bytes)
System manager	$640 + (44 + 4) \times (g + h + i) + (100 + 4) \times \{(p + q + 2) + u \times (15 + 1)\} + (92 + 4) \times c + 40 \times (k + m + n \times o + u) \times 14 + 256 \times m + 128 \times c + 36 \times d + 12 \times e + 96 \times o + v \times (16 \times 34 \times (k + u) + 15 + 36 \times z + 15) + w \times (48 + 15 + 4 \times z + 15 + 4 \times y + 15) + v \times (132 + 15) + 8 + 5844 + s + s \times o + 16 + 96 \times o + 1024$ <p> <i>c</i>: Number of units <i>d</i>: Number of -c options specified in <code>pdunit</code> operand <i>e</i>: Number of <code>pdcltgrp</code> operands specified <i>g</i>: Number of FESs in the system <i>h</i>: Number of BESs in the system <i>i</i>: Number of DSs in the system <i>j</i>: Number of FESs in unit <i>k</i>: Number of BESs in unit <i>m</i>: Number of DSs in the unit <i>n</i>: Number of alternate BESs in the unit <i>o</i>: If the unit is subject to standby-less system switchover (1:1), 1; if not 0 <i>p</i>: $i + k + m + n$ <i>q</i>: $24 + t + j \times 3 + k \times 15 + m \times 7$ <i>r</i>: $14 \times (k + m + u) + p + q + u \times 15 + 2 + 38 + 4 + p \times 4$ <i>s</i>: $212 + 2052 + 128 \times (r + 3) + v \times (40 \times (k + u) + 72 \times (k + u))$ <i>t</i>: If the unit contains MGR, 2; if not, 0 <i>u</i>: Number of acceptable BESs (value of the <code>pd_ha_max_act_guest_servers</code> operand) <i>v</i>: If the unit is subject to standby-less system switchover (effects distributed), 1; if not, 0 <i>w</i>: If there is a unit that is subject to standby-less system switchover (effects distributed), 1; if not, 0 <i>y</i>: Number of units in the HA group <i>z</i>: Number of servers in the HA group </p>

Process item	Shared memory calculation formula (bytes)
Name server	$X + Y + 1024 + Z$ <p> <i>X</i>: $\text{MAX}(65536, \{ \downarrow ((4096 + 5808 + (\text{total number of nodes in the system} \times 72) + a + b) \div 1024) \downarrow + 2 \} \times 1024)$ <i>Y</i>: $\text{MAX}(16384, \{ \downarrow (c + d + e) \div 1024 \downarrow + 5 \} \times 1024)$ <i>Z</i>: (total number of BESs in the unit + <i>j</i> + <i>g</i> + <i>k</i>) \times 1024 <i>a</i>: When the unit contains MGR: 1056 When the unit contains no MGR: 0 <i>b</i>: (total number of BESs in the unit + <i>j</i> + <i>g</i> + <i>k</i>) \times 528 <i>c</i>: When the unit contains MGR: {(total number of units in the system - 1) + (total number of units in the system - 1) \times 15} \times 244, When the unit contains no MGR: 488 <i>d</i>: When the unit contains FES: (number of BESs in other units + number of DSs in other units) \times 244, When the unit contains no FES: 0 <i>e</i>: When the unit contains BES and DS: Number of BESs in other units \times 244 When the unit does not contain BES nor DS: 0 <i>f</i>: When the unit contains FES: 3 When the unit contains no FES: 0 <i>g</i>: $24 + f + h + i$ <i>h</i>: When the unit contains DS: 7 When the unit contains no DS: 0 <i>i</i>: When the unit contains BES: $6 \times$ (total number of BESs in the unit + <i>j</i>) When the unit contains no BES: 0 <i>j</i>: Unit subject to standby-less system switchover (effects distributed): Number of acceptable BESs Other unit: 0 <i>k</i>: Unit subject to standby-less system switchover (1:1): Number of alternate BESs Other unit: 0 </p>

Process item	Shared memory calculation formula (bytes)
Node manager	<p>Unit contains MGR:</p> $\uparrow (1152 + 416 \times \text{total number of units in the system} + 80 \times \text{total number of servers in the system} + 1536 + 928 + 56 \times C + 240 \times A + 44 \times A + 28 \times A + 16 \times B + 16 \times \text{total number of BESs in the system} + 8 \times \text{total number of units in the system} + 32)$ $\div 1024 \uparrow \times 1024$ <p>Unit contains no MGR:</p> $\uparrow (928 + 56 \times C + 240 \times A + 44 \times A + 28 \times A + 16 \times B + 16 \times \text{total number of BESs in the system} + 8 \times \text{total number of units in the system} + 32)$ $\div 1024 \uparrow \times 1024$ <p>A: Value of <code>pd_utl_exec_mode = 0</code>: 1024 Value of <code>pd_utl_exec_mode = 1</code>: 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 1024, use 1024.</p> <p>B: <code>pdcltgrp</code> operand not specified: 0 <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 Unit subject to standby-less system switchover (effects distributed): Number of acceptable BESs Other unit: 0</p>

Process item	Shared memory calculation formula (bytes)
I/O server	<p> $\uparrow(28 + (\uparrow(32 + A) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128$ When the unit is not subject to standby-less system switchover (effects distributed): A:When <code>pd_large_file_use=N</code> is specified (or omitted) $3248 + (14 + 16) \times 808 + 1 \times 272 + (534 \times 272)^1$ $+ \{(534 \times 272 + 16 \times 272 + \text{value of } pd_max_file_no \times 808) \times \text{number of BESs}\}^2$ $+ \{534 \times 272 + 16 \times 272 + \text{value of } pd_max_file_no \times 808\}^3$ When <code>pd_large_file_use=Y</code> is specified: $3248 + (14 + 16) \times 972 + 1 \times 276 + (534 \times 276)^1$ $+ \{(534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972) \times \text{number of BESs}\}^2$ $+ \{534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972\}^3$ </p> <p> ¹ Add this value if there are FESs. ² Add this value if there are BESs. ³ Add this value if there are DSs. In a unit subject to standby-less system switchover (1:1), double the value obtained in the above formula. When the unit is subject to standby-less system switchover (effects distributed): A:When <code>pd_large_file_use=N</code> is specified (or omitted) $\uparrow 48 + 24 \times \text{number of BESs}^4 \div 16 \uparrow \times 16$ $+ \uparrow (3248 + 16 \times 808 + 534 \times 272 + 16 \times 272 + \text{value of } pd_max_file_no \times 808) \div 16 \uparrow \times 16 \times \text{number of BESs}^4$ $+ \uparrow (3248 + (14 + 16) \times 808 + 1 \times 272) \div 16 \uparrow \times 16$ When <code>pd_large_file_use=Y</code> is specified $\uparrow 48 + 24 \times \text{number of BESs}^4 \div 16 \uparrow \times 16$ $+ \uparrow (3248 + 16 \times 972 + 534 \times 276 + 16 \times 276 \text{ value of } pd_max_file_no \times 972) \div 16 \uparrow \times 16 \times \text{number of BESs}^4$ $+ \uparrow (3248 + (14 + 16) \times 972 + 1 \times 276) \div 16 \uparrow \times 16$ </p> <p> ⁴ Includes the value of <code>pd_ha_max_act_guest_servers</code>. </p>

Process item	Shared memory calculation formula (bytes)
Log server	$32 + 48 + 128 \times 37$ $+ \{$ $384 + 128 \times 7 + 1024 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of pd_log_rec_leng}^* \uparrow$ $\times \text{value of pd_log_rec_leng}^*$ $+ 64 + 4096 \times 2 + (256 + 480) \times B$ $+ \uparrow \{(512 + 256) + 128 \times B + 256 \times B\} \div (8192 - 128) \uparrow \times 8192$ $+ 128 \times \text{value of pd_log_write_buff_count}^*$ $+ (\text{value of pd_log_write_buff_count}^* + A)$ $\times \uparrow \{\text{value of pd_log_max_data_size}^* + (68 + 44 + 96 + 160)\} \div 4096 \uparrow$ $\times 4096 + C$ $\} \times \text{number of servers in the unit} + D + 128 \times \text{number of FESs in the unit}$ $+ 512 \times \text{number of HiRDB servers in the unit}$ <p><i>A</i>: 16 <i>B</i>: Number of pdlogadfg -d sys operands specified* <i>C</i>: 512 <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): 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. 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>

Process item	Shared memory calculation formula (bytes)
Synchronization point dump server	$\left\{ \begin{aligned} & \uparrow (368 + 1456 \times 2) \div 1024 \uparrow \times 1024 \\ & + \uparrow \{ (96 + 80 + 208 + 208) + 192 \times (\text{number of pdlogadfg -d spd operands specified}^*) \\ & + 416 \times (\text{number of pdlogadpf -d spd operands specified}^*) + 1023 \} \div 1024 \uparrow \\ & \times 1024 \end{aligned} \right\} \times (\text{total number of servers} + A)$ <p><i>A</i>: 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): 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>

Process item	Shared memory calculation formula (bytes)
Common to all units	$a + \{b + 64 + (s + 3) \times c + 64 + 48 + d + e\}$ $\times (\text{total number of FESSs, BESs, and DSs in unit} + i)$ $+ (g \times (\text{total number of BESs and DSs in unit} + i)) + f$ $+ (\text{value of pd_max_server_process} + 100) \times (48 + 16) + 32$ $+ (\text{value of pd_max_server_process} + 100 + 384) \times 32 + 32 + h$ $+ (\text{value of pd_max_server_process} + 127) \times 32 + 32$ <p>If the standby-less system switchover (effects distributed) facility is used, add:</p> $(\uparrow(28 + (\uparrow(56 + 72584) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128)$ <p><i>a:</i> 28864 <i>b:</i> 2988 <i>c:</i> 1952 <i>d:</i> 32 × 32 <i>e:</i> 64 + 64 × {(s + 3) × 2 + MAX(5, (↓s + 3) ÷ 10 ↓) + 7} <i>f:</i> 512 × (13 + total number of FESSs, BESs, and DSs in unit × 3) × 2 <i>g:</i> {(96 + value of pd_lck_until_disconnect_cnt × 112 + 4095) ÷ 4096} × 4096 × 2 <i>h:</i> ↑ (number of port numbers specified with pd_registered_port × 16 + 32 + 1023) ÷ 1024 ↑ × 1024 If pd_registered_port is omitted: 0 <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): pd_ha_max_guest_servers operand correction value <i>s:</i> Value of pd_max_users <i>t:</i> Value of pd_max_dic_process <i>u:</i> Value of pd_max_bes_process <i>s</i> is <i>t</i> for a DS and <i>u</i> for a BES. If pd_max_dic_process or pd_max_bes_process is omitted, use <i>s</i>.</p>

Process item	Shared memory calculation formula (bytes)
Transaction log server	$\{1024 + 512 \times A\} \times (\text{number of servers in unit} + H)$ $+ \{$ $128 \times B + 128$ $+ [F + \uparrow 256 \div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*$ $+ \uparrow (\text{value of pd_log_max_data_size}^* + 68 + 44 + 96 + 160)$ $\div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*]$ $\times D \times 2 + E + (48 + 8) \times B \times 2$ $\} \times (\text{number of BESs and DSs in unit} + H)$ $+ \{$ $584 \times B + 128 \times B + 64 \times B \times C + 128 + F$ $+ \uparrow 512 \div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*$ $+ \uparrow (\text{value of pd_log_max_data_size}^* + 68 + 44 + 96 + 160)$ $\div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*$ $+ E + (48 + 8) \times (B \times 2 + 2)$ $\} \times (\text{number of servers in unit} + H)$ <p> <i>A</i>: 2 <i>B</i>: $7 + J \times 2$ <i>C</i>: Number of BESs in the entire system $\times 4$ + number of DSs in the entire system $\times 2$ + number of FESs in the entire system <i>D</i>: Number of servers in unit <i>E</i>: 512 <i>F</i>: 512 <i>H</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): pd_ha_max_act_guest_servers operand correction value <i>J</i>: Maximum value of <i>s</i>, <i>t</i>, and <i>u</i> in the servers in the unit <i>s</i>: Value of pd_max_users <i>t</i>: Value of pd_max_dic_process <i>u</i>: Value of pd_max_bes_process * 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> <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): pd_ha_max_act_guest_servers operand correction value </p>

Process item	Shared memory calculation formula (bytes)
Security audit	$\lceil A \div 1024 \rceil \times 1024$ <p><i>A</i>: 560 if the <code>pd_aud_file_name</code> operand is omitted 456 + (304 × 200) + <i>B</i> if the <code>pd_aud_file_name</code> operand is specified <i>B</i>: 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 4096 or greater: (160 × value of <code>pd_aud_async_buff_count</code> operand) +{(\lceil value of <code>pd_aud_async_buff_size</code> operand \div 4096 \lceil × 4096) × value of <code>pd_aud_async_buff_count</code> operand} + 4096 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>

Explanation:

MGR: System manager

FES: Front-end server

DS: Dictionary server

BES: Back-end server

(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: $\{ \uparrow (432 + 304 \times n) \div 1024 \uparrow + 289 + x + z \} \times 1024$</p> <p>Value of <code>pd_utl_exec_mode</code> set to 1: $\{ \uparrow (432 + 304 \times n) \div 1024 \uparrow + \uparrow (s \times 2000 + 136) \div 1024 \uparrow + y + z \} \times 1024$</p> <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): $\uparrow 64 + \{(\text{Number of BESs in unit} + \text{number of acceptable guest BESs}) \times 48\} \div 1024 \uparrow$ Other unit: 0</p> <p><i>n</i>: Number of servers in unit + $\beta + \gamma$ + number of utility servers in unit + 1 α: If there is a MGR in the unit: 2 If there is a FES in the unit: 3 If there is a DS in the unit: 7 If there are BESs in the unit: $(\text{number of BESs} + \beta) \times 15$</p> <p><i>s</i>: Value of <code>pd_max_users</code> <i>t</i>: Value of <code>pd_max_dic_process</code> <i>u</i>: Value of <code>pd_max_bes_process</code></p> <p>β: Unit subject to standby-less system switchover (effects distributed): Number of acceptable BESs Other unit: 0</p> <p>γ: Unit subject to standby-less system switchover (1:1): Number of alternate BESs Other unit: 0</p>

Process item	Shared memory calculation formula (bytes)
Lock server	<p>• When there is a server (FES, BES, or DS) in the unit: 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_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 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>. In the unit subject to standby-less system switchover (1:1), <i>servers in the unit</i> means <i>all accepted BESs + all alternate BESs</i>.</p> $y \sum_{x=1} \{ 272 + 80 + c_x + d_x + 64 + 8192 + g_x + 80 + i_x + 64 + 16336 + 64 + n_x + p_x + t_x + u_x + 16 \}$ <p><i>x</i>: Server serial number in the unit <i>y</i>: Number of servers in the unit</p> <p><i>c_x</i>: When <code>pd_lck_hash_entry</code> is omitted or 0 is specified: For FES with <code>pd_fes_lck_pool_size</code> omitted: $(\downarrow(8 + 8 \times \text{MAX}(\uparrow(p + 3) \times (\text{value of } \text{pd_max_access_tables} + 4) \div 10 \uparrow, 11261)) \div 16 \downarrow + 1) \times 16$ For FES with <code>pd_fes_lck_pool_size</code> specified: $(\downarrow(8 + 8 \times \text{MAX}(\uparrow(\text{value of } \text{pd_fes_lck_pool_size} \times 4) \div 10 \uparrow, 11261)) \div 16 \downarrow + 1) \times 16$ For BES or DS: $(\downarrow(8 + 8 \times \text{MAX}(\uparrow((p + 3) \times 2 \times 5 + \text{value of } \text{pd_lck_pool_size} \times 4) \div 10 \uparrow, 11261)) \div 16 \downarrow + 1) \times 16$ When 2 or a greater non-prime number is specified for <code>pd_lck_hash_entry</code>: $(\downarrow(8 + 8 \times \text{largest prime number that is less than the value of } \text{pd_lck_hash_entry}) \div 16 \downarrow + 1) \times 16$ 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>

Process item	Shared memory calculation formula (bytes)
	<p>d_x: For FES with <code>pd_fes_lck_pool_size</code> omitted: $((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 128$ For FES with <code>pd_fes_lck_pool_size</code> specified: $\text{Value of } pd_fes_lck_pool_size \times 4 \times 128$ For BES or DS: $(p + 3) \times 2 \times 5 + \text{value of } pd_lck_pool_size \times 4 \times 128$</p> <p>$g_x$: For FES: $(p + 3) \times 2 \times 320$ For BES with the value of <code>pd_utl_exec_mode = 1</code> and $s > 32$: $(p + 3) \times 2 + s \times 320$ For BES with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 32$: $(p + 3) \times 2 + 32 \times 320$ For DS with the value of <code>pd_utl_exec_mode = 1</code> and $s > 16$: $(p + 3) \times 2 + s \times 320$ For DS with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 16$: $(p + 3) \times 2 + 16 \times 320$</p> <p>$i_x$: For FES with <code>pd_fes_lck_pool_size</code> omitted: $((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 2 \times 112$ For FES with <code>pd_fes_lck_pool_size</code> specified: $(\text{value of } pd_fes_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow)$ rounded up to the next even value $\times 112$ For BES with the value of <code>pd_utl_exec_mode = 1</code> and $s > 32$: $(\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow$ $+ (p + 3) \times 2 \times 2 \times 5 + s \times (\text{value of } pd_max_rdarea_no + 1))$ rounded up to the next even value $\times 112$ For BES with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 32$: $(\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow + (p$ $+ 3)$ $\times 2 \times 2 \times 5 + 32 \times (\text{value of } pd_max_rdarea_no + 1))$ rounded up to the next even value $\times 112$ For DS with the value of <code>pd_utl_exec_mode = 1</code> and $s > 16$: $(\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow$ $+ (p + 3) \times 2 \times 2 \times 5 + s + 4)$ rounded up to the next even value $\times 112$ For DS with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 16$: $(\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow$ $+ (p + 3) \times 2 \times 2 \times 5 \times 2 \times 2 \times 5 + 20)$ rounded up to the next even value $\times 112$</p>

Process item	Shared memory calculation formula (bytes)
	<p>n_x: For FES: $(p + 3) \times 2 \times 80$ For BES with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 32$: $((p + 3) \times 2 \times 17 + s) \times 80$ For BES with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 32$: $((p + 3) \times 2 \times 17 + 32) \times 80$ For DS with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 16$: $((p + 3) \times 2 \times 17 + s) \times 80$ For DS with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 16$: $((p + 3) \times 2 \times 17 + 16) \times 80$</p> <p>$p$: For FES, when the number of FESs in the HiRDB system > 1: $s + 1$ For FES, when the number of FESs in the HiRDB system = 1: s For BES with $s > \text{value of } pd_max_bes_process$: s For BES with $s \leq \text{value of } pd_max_bes_process$: Value of <code>pd_max_bes_process</code> For DS with $s > \text{value of } pd_max_dic_process$: s For DS with $s \leq \text{value of } pd_max_dic_process$: Value of <code>pd_max_dic_process</code></p> <p>s: Value of <code>pd_max_users</code></p> <p>t_x: For FES: $32 + (p + 3) \times 2 \times \lceil \text{value of } pd_max_open_holdable_cursors \div 16 \rceil \times 4$ For BES with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 32$: $32 + ((p + 3) \times 2 + s) \times \lceil \text{value of } pd_max_open_holdable_cursors \div 16 \rceil \times 4$ For BES with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 32$: $32 + ((p + 3) \times 2 + 32) \times \lceil \text{value of } pd_max_open_holdable_cursors \div 16 \rceil \times 4$ For DS with the value of <code>pd_utl_exec_mode</code> = 1 and $s > 16$: $32 + ((p + 3) \times 2 + s) \times \lceil \text{value of } pd_max_open_holdable_cursors \div 16 \rceil \times 4$ For DS with the value of <code>pd_utl_exec_mode</code> = 0 or $s \leq 16$: $32 + ((p + 3) \times 2 + 16) \times \lceil \text{value of } pd_max_open_holdable_cursors \div 16 \rceil \times 4$</p>

Process item	Shared memory calculation formula (bytes)
	<p>u_x: For FES with <code>pd_fes_lck_pool_size</code> omitted: $32 + ((p + 3) \times (\text{value of } pd_max_access_tables + 4)) \times 2$ $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <p>For FES with <code>pd_fes_lck_pool_size</code> specified: $32 + (\text{value of } pd_fes_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow)$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors$ $\div 16 \uparrow \times 4$</p> <p>For BES with the value of <code>pd_utl_exec_mode = 1</code> and $s > 32$: $32 + (\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow$ $+ (p + 3) \times 2 \times 2 \times 5 + s \times (\text{value of } pd_max_rdarea_no + 1))$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors$ $\div 16 \uparrow \times 4$</p> <p>For BES with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 32$: $32 + (\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow$ $+ (p + 3) \times 2 \times 2 \times 5 + 32 \times (\text{value of } pd_max_rdarea_no + 1))$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors$ $\div 16 \uparrow \times 4$</p> <p>For DS with the value of <code>pd_utl_exec_mode = 1</code> and $s > 16$: $32 + (\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow$ $+ (p + 3) \times 2 \times 2 \times 5 + s + 4)$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <p>For DS with the value of <code>pd_utl_exec_mode = 0</code> or $s \leq 16$: $32 + (\text{value of } pd_lck_pool_size \times 5 + \downarrow \text{value of } pd_lck_pool_size \div 3 \downarrow$ $+ (p + 3) \times 2 \times 2 \times 5 \times 2 \times 2 \times 5 + 20)$ rounded up to the next even value $\times \uparrow \text{value of } pd_max_open_holdable_cursors \div 16 \uparrow \times 4$</p> <ul style="list-style-type: none"> When there is no server (FES, BES, or DS) in the unit <p>16688</p>

Process item	Shared memory calculation formula (bytes)
Transaction manager	$304 + 32 \times B + 192 \times s \times 2$ <p>If the unit contains FES, add the following value:*</p> $+ 1048 + (416 + 688 + 256 + 392 \times 2) \times (A \times 2 + 7) + 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ C$ <p>If the unit contains BES, add the following value:*</p> $+ 1048 + (416 + 688 + 256 + 392 \times 2) \times (u \times 2 + 7)$ $+ 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ D$ <p>If the unit contains DS, add the following value:*</p> $+ 1048 + (416 + 688 + 256 + 392 \times 2) \times (t \times 2 + 7)$ $+ 256 \times 2$ $+ 128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (A \times 2 + 7)$ $+ E$ <p><i>s</i>: Value of <code>pd_max_users</code> <i>t</i>: Value of <code>pd_max_dic_process</code> <i>u</i>: Value of <code>pd_max_bes_process</code> <i>A</i>: For a multi-FES system: $s + 1$; if not a multi-FES system: s <i>B</i>: 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 <i>C</i>: 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 <i>D</i>: If the unit meets one condition in <i>Conditions</i> below: $128 \times (\text{number of BESs in the system} \times 4 + \text{number of DSs in the system} \times 2$ $+ \text{number of FESs in the system}) \times (u \times 2 + 7)$</p>

Process item	Shared memory calculation formula (bytes)
	<p>If the unit does not meet both conditions in <i>Conditions</i> below: 0 <i>E</i>: 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</p> <p>Conditions:</p> <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. <p>* Add the value obtained from the above formula as many times as the value of <i>B</i>. For units subject to standby-less system switchover (1:1): $\text{Number of accepted BESs} + \text{Number of alternate BESs}$ Other than the above, the number of the servers in the unit</p>
Timer server	$32 \times (\text{value of } pd_max_users + 3)$ $\times (\text{number of BESs in system} + 1 + \text{number of utility servers in unit} + 1)$ $+ 1440 + (48 - 32) \times 2$ <p>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 When there is DS in the unit: 7 When there is BES in the unit: number of BESs $\times 15$</p>
Statistics log server	$424 + 128 \times 16 + 32 + 288 \times 2 + 1168 + 144 \times 3$ $+ \text{value of } pd_stj_buff_size \times 1024 \times 3 + 64 + 4096 + 8192$
Process server	$176 + 512 \times \text{MAX}(c, 256) + 80 + 256 + (\text{value of } pd_max_server_process + 50) \times (256 + 160) + 16 + 1 \times 34 + 16$ $c: \uparrow (45 + d + e + f + (g \times \text{number of BESs in the unit}^*) + h + i) \div 16 \uparrow \times 16$ d : If the unit contains MGR, 46; if not, j . e : If the unit contains DS, 15; if not, 0. f : If the unit contains FES, 9; if not, 0. g : If the unit contains BES, 24; 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 manager is specified in the <code>pd_mlg_msg_log_unit</code> operand, 1; if <code>local</code> is specified, 2. * 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.

Process item	Shared memory calculation formula (bytes)
System manager	$672 + (48 + 8) \times (g + h + i) + (108 + 8) \times \{(p + q + 2) + u \times (15 + 1)\} + (104 + 8) \times c + 40 \times (k + m + n \times o + u) \times 14 + 256 \times m + 128 \times c + 40 \times d + 16 \times e + 96 \times o + v \times (16 \times 34 \times (k + u) + 15 + 44 \times z + 15) + w \times (48 + 15 + 4 \times z + 15 + 4 \times y + 15) + v \times (144 + 15) + 8 + 5864 + s + s \times o + 16 + 96 \times o + 1024$ <p> <i>c</i>: Number of units <i>d</i>: Number of -c options specified in <code>pdunit</code> operand <i>e</i>: Number of <code>pdcltgrp</code> operands specified <i>g</i>: Number of FESs in the system <i>h</i>: Number of BESs in the system <i>i</i>: Number of DSs in the system <i>j</i>: Number of FESs in unit <i>k</i>: Number of BESs in unit <i>m</i>: Number of DSs in the unit <i>n</i>: Number of alternate BESs in the unit <i>o</i>: If the unit is subject to standby-less system switchover, 1; if not, 0 <i>p</i>: $i + k + m + n$ <i>q</i>: $24 + t + j \times 3 + k \times 15 + m \times 7$ <i>r</i>: $14 \times (k + m + u) + p + q + u \times 15 + 2 + 38 + 4 + p \times 4$ <i>s</i>: $236 + 2052 + 148 \times (r + 3) + v \times (40 \times (k + u) + 72 \times (k + u))$ <i>t</i>: If the unit contains MGR, 2; if not, 0 <i>u</i>: Number of acceptable BESs (value of the <code>pd_ha_max_act_guest_servers</code> operand) <i>v</i>: If the unit is subject to standby-less system switchover (effects distributed), 1; if not, 0 <i>w</i>: If there is a unit that is subject to standby-less system switchover (effects distributed), 1; if not, 0 <i>y</i>: Number of units in the HA group <i>z</i>: Number of servers in the HA group </p>

Process item	Shared memory calculation formula (bytes)
Name server	$X + Y + 1024 + Z$ <p> <i>X</i>: $\text{MAX}(65536, \{ \downarrow ((4096 + 5808 + (\text{total number of nodes in the system} \times 72) + a + b) \div 1024) \downarrow + 2 \} \times 1024)$ <i>Y</i>: $\text{MAX}(16384, \{ \downarrow (c + d + e) \div 1024 \downarrow + 5 \} \times 1024)$ <i>Z</i>: (total number of BESs in the unit + <i>j</i> + <i>g</i> + <i>k</i>) \times 1024 <i>a</i>: When the unit contains MGR: 1056 When the unit contains no MGR: 0 <i>b</i>: (total number of BESs in the unit + <i>j</i> + <i>g</i> + <i>k</i>) \times 528 <i>c</i>: When the unit contains MGR: {(total number of units in the system - 1) + (total number of units in the system - 1) \times 15} \times 244, When the unit contains no MGR: 488 <i>d</i>: When the unit contains FES: (number of BESs in other units + number of DSs in other units) \times 244, When the unit contains no FES: 0 <i>e</i>: When the unit contains BES and DS: Number of BESs in other units \times 244 When the unit does not contain BES nor DS: 0 <i>f</i>: When the unit contains FES: 3 When the unit contains no FES: 0 <i>g</i>: $24 + f + h + i$ <i>h</i>: When the unit contains DS: 7 When the unit contains no DS: 0 <i>i</i>: When the unit contains BES: $6 \times$ (total number of BESs in the unit + <i>j</i>) When the unit contains no BES: 0 <i>j</i>: Unit subject to standby-less system switchover (effects distributed): Number of acceptable BESs Other unit: 0 <i>k</i>: Unit subject to standby-less system switchover (1:1): Number of alternate BESs Other unit: 0 </p>

Process item	Shared memory calculation formula (bytes)
Node manager	<p>Unit contains MGR:</p> $\begin{aligned} &\uparrow (1312 + 448 \times \text{total number of units in the system} + 96 \times \text{total number of servers} \\ &\text{in the system} \\ &+ 2048 + 1104 + 72 \times C + 240 \times A + 44 \times A + 28 \times A \\ &+ 16 \times B + 16 \times \text{total number of BESs in the system} + 8 \times \text{total number of units in} \\ &\text{the system} + 48) \\ &\div 1024 \uparrow \times 1024 \end{aligned}$ <p>Unit contains no MGR:</p> $\begin{aligned} &\uparrow (1104 + 72 \times C + 240 \times A + 44 \times A + 28 \times A \\ &+ 16 \times B + 16 \times \text{total number of BESs in the system} + 8 \times \text{total number of units in} \\ &\text{the system} + 48) \\ &\div 1024 \uparrow \times 1024 \end{aligned}$ <p><i>A</i>: Value of <code>pd_utl_exec_mode = 0</code>: 1024 Value of <code>pd_utl_exec_mode = 1</code>: 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 BES, add: Value of <code>pd_max_users</code> \times D If the value of <i>A</i> obtained from the previous formula is not greater than 1024, use 1024.</p> <p><i>B</i>: <code>pdcltgrp</code> operand not specified: 0 <code>pdcltgrp</code> operand specified: Number of <code>pdcltgrp</code> operands specified + 1</p> <p><i>C</i>: Number of servers in the unit + <i>E</i></p> <p><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>

Process item	Shared memory calculation formula (bytes)
I/O server	<p>$\uparrow(56 + (\uparrow(56 + A) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128$ When the unit is not subject to standby-less system switchover (effects distributed): A: When <code>pd_large_file_use=N</code> is specified (or omitted) $3248 + (14 + 16) \times 808 + 1 \times 272 + (534 \times 272)^1$ $+ \{(534 \times 272 + 16 \times 272 + \text{value of } pd_max_file_no \times 808) \times \text{number of BESs}\}^2$ $+ \{534 \times 272 + 16 \times 272 + \text{value of } pd_max_file_no \times 808\}^3$ $+ (48 - 32) \times 3$</p> <p>When <code>pd_large_file_use=Y</code> is specified $3248 + (14 + 16) \times 972 + 1 \times 276 + (534 \times 276)^1$ $+ \{(534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972) \times \text{number of BESs}\}^2$ $+ \{534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972\}^3$ $+ (48 - 32) \times 3$</p> <p>¹ Add this value if there are FESs. ² Add this value if there are BESs. ³ Add this value if there are DSs. For a standby-less system switchover configuration (1:1) target unit, double the value obtained in the above formula. When the unit is subject to standby-less system switchover (effects distributed): A: When <code>pd_large_file_use=N</code> is specified (or omitted) $\uparrow 64 + 24 \times \text{number of BESs}^4 \div 16 \uparrow \times 16$ $+ \uparrow 3296 + 16 \times 808 + 534 \times 272 + 16 \times 272 + \text{value of } pd_max_file_no \times 808) \div 16 \uparrow \times 16 \times \text{number of BESs}^4$ $+ \uparrow(3296 + (14 + 16) \times 808 + 1 \times 272) \div 16 \uparrow \times 16$</p> <p>When <code>pd_large_file_use=Y</code> is specified $\uparrow 64 + 24 \times \text{number of BESs}^4 \div 16 \uparrow \times 16$ $+ \uparrow(3296 + 16 \times 972 + 534 \times 276 + 16 \times 276 + \text{value of } pd_max_file_no \times 972) \div 16 \uparrow \times 16 \times \text{number of BESs}^4$ $+ \uparrow(3296 + (14 + 16) \times 972 + 1 \times 276) \div 16 \uparrow \times 16$</p> <p>⁴ Includes the value of <code>pd_ha_max_act_guest_servers</code>.</p>

Process item	Shared memory calculation formula (bytes)
Log server	$32 + 48 + 128 \times 37$ $+ \{$ $432 + 128 \times 7 + 1168 + 512$ $+ \uparrow (128 + 256 + 160 + 8 + 64) \div \text{value of pd_log_rec_leng}^* \uparrow$ $\times \text{value of pd_log_rec_leng}^*$ $+ 64 + 4096 \times 2 + (256 + 480) \times B$ $+ \uparrow \{(512 + 256) + 128 \times B + 464 \times B\} \div (8192 - 128) \uparrow \times 8192$ $+ 144 \times \text{value of pd_log_write_buff_count}^*$ $+ (\text{value of pd_log_write_buff_count}^* + A)$ $\times \uparrow \{\text{value of pd_log_max_data_size}^* + (68 + 44 + 96 + 160)\} \div 4096 \uparrow \times$ 4096 $+ C$ $\} \times \text{number of servers in the unit} + D + 128 \times \text{number of FESs in the unit}$ $+ 512 \times \text{number of HiRDB servers in the unit}$ <p>A: 16 B: Number of pdlogadfg -d sys operands specified* C: 512 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): 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. 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>

Process item	Shared memory calculation formula (bytes)
Synchronization point dump server	$\left\{ \begin{aligned} & \uparrow (384 + 1536 \times 2) \div 1024 \uparrow \times 1024 \\ & + \uparrow \{(128 + 80 + 240 + 240) + 192 \times (\text{number of pdlogadfg -d spd operands specified}^*) \\ & + 416 \times (\text{number of pdlogadpf -d spd operands specified}^*) + 1023\} \div 1024 \uparrow \\ & \times 1024 \end{aligned} \right\} \times (\text{total number of servers} + A)$ <p><i>A</i>: 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): 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>

Process item	Shared memory calculation formula (bytes)
Common to all units	$a + \{b + 80 + (s + 3) \times c + 64 + 48 + d + e\}$ $\times (\text{total number of FESSs, BESs, and DSs in the unit } + i)$ $+ (g \times (\text{total number of BESs and DSs in the unit } + i)) + f$ $+ (\text{value of } pd_max_server_process + 100) \times (64 + 16) + 32$ $+ (\text{value of } pd_max_server_process + 100 + 384) \times 32 + 32 + h$ $+ (\text{value of } pd_max_server_process + 127) \times 48 + 32$ <p>If the standby-less system switchover (effects distributed) facility is used, add:</p> $(\uparrow(56 + (\uparrow(56 + 88560) \div 32 \uparrow \times 32)) \div 128 \uparrow \times 128)$ <p><i>a</i>: 38272 <i>b</i>: 3480 <i>c</i>: 2760 <i>d</i>: 48 × 32 <i>e</i>: 80 + 96 × {(s + 3) × 2 + MAX(5, ↓ [s + 3] ÷ 10 ↓) + 7} <i>f</i>: 512 × (13 + (total number of FESSs, BESs and DSs in unit + i) × 3) × 2 <i>g</i>: {(128 + value of pd_lck_until_disconnect_cnt × 112 + 4095) ÷ 4096} × 4096 × 2 <i>h</i>: ↑ (number of port numbers specified by pd_registered_port × 16 + 32 + 1023) ÷ 1024 ↑ × 1024 0 if pd_registered_port is not specified <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): pd_ha_max_guest_servers operand correction value <i>s</i>: Value of pd_max_users <i>t</i>: Value of pd_max_dic_process <i>u</i>: Value of pd_max_bes_process <i>s</i> is <i>t</i> for a DS and <i>u</i> for a BES. If pd_max_dic_process or pd_max_bes_process is omitted, use <i>s</i>.</p>

Process item	Shared memory calculation formula (bytes)
Transaction log server	$\{1168 + 688 \times A\} \times (\text{number of servers in unit} + H)$ $+ \{$ $128 \times B + 144$ $+ [G + \uparrow 256 \div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*$ $+ \uparrow (\text{value of pd_log_max_data_size}^* + 68 + 44 + 96 + 160)$ $\div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*]$ $\times D \times 2 + E + (48 + 8) \times B \times 2$ $\} \times (\text{number of BESs and DSs in unit} + H)$ $+ \{$ $600 \times B + 128 \times B + 64 \times B \times C + 144 + G$ $+ \uparrow 512 \div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*$ $+ \uparrow (\text{value of pd_log_max_data_size}^* + 68 + 44 + 96 + 160)$ $\div \text{value of pd_log_rec_leng}^* \uparrow \times \text{value of pd_log_rec_leng}^*$ $+ E + (48 + 8) \times (B \times 2 + 2)$ $\} \times (\text{number of servers in unit} + H)$ <p>A: 2 B: $7 + J \times 2$ C: Number of BESs in the entire system $\times 4$ + number of DSs in the entire system $\times 2$ + number of FESs in the entire system D: Number of servers in unit E: 512 G: 512 H: 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): pd_ha_max_act_guest_servers operand correction value J: Maximum value of <i>s</i>, <i>t</i>, and <i>u</i> in the servers in the unit s: Value of pd_max_users t: Value of pd_max_dic_process u: Value of pd_max_bes_process</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): pd_ha_max_act_guest_servers operand correction value</p>

Process item	Shared memory calculation formula (bytes)
Security audit	$\uparrow A \div 1024 \uparrow \times 1024$ <p><i>A</i>: 624 if the <code>pd_aud_file_name</code> operand is omitted 520 + (320 × 200) + <i>B</i> if the <code>pd_aud_file_name</code> operand is specified <i>B</i>: 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 4096 or greater: (176 × value of <code>pd_aud_async_buff_count</code> operand) +{(\uparrow value of <code>pd_aud_async_buff_size</code> operand \div 4096 \uparrow × 4096) × value of <code>pd_aud_async_buff_count</code> operand}</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>

Explanation:

MGR: System manager

FES: Front-end server

DS: Dictionary server

BES: Back-end server

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

$$\begin{aligned}
& 40 + b \times 1.3 + c + d + k + 1.6 + x + y + 4 \\
& + \{[(a + 12) \div 13] \times 1.1 + [(a + 62) \div 63] + 3.7\} \times (e + 3) \\
& + \{ \\
& \quad \uparrow \uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4 \\
& \quad + 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\} \\
& \quad + 4 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} \\
& \quad + 32 + \{16 \times (f + 1) \times g\} + 112 \times B \\
& \quad + \uparrow \{(c \div 8) + 7\} \div 64 \uparrow \times 8 + \uparrow \{(k \div 8) + 7\} \div 64 \uparrow \times 8 \\
& \quad + \text{MAX}\{a \times (e + 3), c \div 8\} \times 88 + \text{MAX}\{a \times (e + 3), k \div 8\} \\
& \quad \times 24 \\
& \quad + \uparrow \{(x \div 4) + 7\} \div 64 \uparrow \times 8 \\
& \quad + \uparrow \{(y - (s \times 592 + t \times 916 + u \times 172)) \div 2\} + 7\} \div 64 \uparrow \\
& \quad \times 8 \\
& \quad + \text{MAX}\{13 \times (e + 3), x \div 4\} \times 88 \\
& \quad + 60 \times \text{MAX}\{21 \times (e + 3), (y - (s \times 592 + t \times 916 + u \\
& \quad \times 172)) \div 2\} \\
& \quad + 44 + 256 + 1024 \\
& \quad \} \div 1024 + A + 7 \\
& I \\
& + \sum_{i=1} (J_i) \\
& i=1
\end{aligned}$$

- Add this value if you specified INITIAL in the pd_def_buf_control_area_assign operand or omitted this operand.

$$+ \{[(a + 12) \div 13] \times 1.1 + [(a + 62) \div 63] + 3.7\} \times (e + 7)$$
- Add this when the HiRDB External Data Access facility is used.

$$+ \uparrow \{ \uparrow (40 + (32 + 144 \times C + D)) \div 512 \uparrow \times 512 \} \div 1024 \uparrow + \uparrow (B \times 3616) \div 1024 \uparrow$$
 (KB)

■ 64-bit mode

$$\begin{aligned}
&40 + b \times 1.3 + c + d + k + 1.6 + x + y + 5 \\
&+ \{[(a + 12) \div 13] \times 1.2 + [(a + 62) \div 63] \times 1.5 + 4.1\} \times (e + 3) \\
&+ \{ \\
&\quad \uparrow \uparrow b \div 64 \uparrow \times (8 \div 16) \uparrow \times 4 \times 4 \\
&\quad + 12 \times \{(b \div 3) + 1 - \text{mod}(b \div 3, 2)\} \\
&\quad + 4 \times a \times \{(e + 3) \times 2 + 1 + \text{MAX}(e \div 10, 5)\} \\
&\quad + 48 + \{16 \times (f + 1) \times g\} \\
&\quad + \uparrow \{(c \div 8) + 7\} \div 64 \uparrow \times 8 + \uparrow \{(k \div 8) + 7\} \div 64 \uparrow \times 8 \\
&\quad + \text{MAX}(a \times (e + 3), c \div 8) \times 104 + \text{MAX}\{a \times (e + 3), k \div 8\} \times 40 \\
&\quad + \uparrow \{(x \div 4) + 7\} \div 64 \uparrow \times 8 \\
&\quad + \uparrow \{[(y - (s \times 600 + t \times 936 + u \times 184)) \div 2] + 7\} \div 64 \uparrow \times 8 \\
&\quad + \text{MAX}\{13 \times (e + 3), x \div 4\} \times 104 \\
&\quad + 72 \times \text{MAX}\{21 \times (e + 3), (y - (s \times 600 + t \times 936 + u \times 184)) \div 2\} \\
&\quad + 72 + 256 + 1536 \\
&\quad \} \div 1024 + A + 7 \\
&I \\
&+ \sum_{i=1} (J_i) \\
&\bullet \text{ Add this value when INITIAL is specified in the pd_def_buf_control_area_assign operand or the} \\
&\quad \text{operand is omitted.} \\
&+ \{[(a + 12) \div 13] \times 1.2 + [(a + 62) \div 63] \times 1.5 + 4.1\} \times (e + 7) \\
&\quad \text{(KB)}
\end{aligned}$$

(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):

$$\text{Formula 1} + \uparrow \{(\uparrow (40 + (\text{value obtained by adding Formulas 2 through 4})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

For 64-bit mode (KB):

$$\text{Formula 1} + \uparrow \{(\uparrow (72 + (\text{value obtained by adding Formulas 2 through 4})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

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.

- Add Formula 4 if the `pd_dfw_awt_process` operand is specified.
- If you omit the `pd_max_commit_write_no` operand (except cases in which `v6compatible` or `v7compatible` is specified in the `pd_sysdef_default_option` operand), or you specify a value other than 0 in the `pd_sysdef_default_option` operand, add Formula 5. However, if you have already added Formula 3, you do not need to add this formula.
- If you omit the `pd_sds_shmpool_size` operand, the following value is set:

For 32-bit mode:

$$\uparrow \{(\uparrow (40 + (\text{total of Formulas 2 through 4})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

For 64-bit mode:

$$\uparrow \{(\uparrow (72 + (\text{total of Formulas 2 through 4})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

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 \\ & \quad \} \div 1024 \\ & + 3.5 + \uparrow (224 \times v \times w) \div 1024 \uparrow + 0.5 \\ & + \uparrow \{ \\ & \quad \uparrow (28 + (\uparrow (32 + ((\uparrow g \div 127 \uparrow + 1) \times 2048 + 128)) \div 32 \uparrow \times 32)) \\ & \quad \div 128 \uparrow \times 128 \\ & \quad \} \div 1024 \uparrow \\ & K \\ & + \sum_{i=1} (L_i) \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 + 1024 \\ & \quad \} \div 1024 \\ & + 3.5 + \uparrow (224 \times v \times w) \div 1024 \uparrow + 0.5 \\ & + \uparrow \{ \\ & \quad \uparrow (56 + (\uparrow (56 + ((\uparrow g \div 127 \uparrow + 1) \times 2048 + 128)) \div 32 \uparrow \times 32)) \\ & \quad \div 128 \uparrow \times 128 \\ & \quad \} \div 1024 \uparrow \\ & K \\ & + \sum_{i=1} (L_i) \end{aligned} $

Condition	Shared memory calculation formula (KB)
Formula 2 (bytes)	<p>32-bit mode 500×1024 $+ (\uparrow 372 \times g \div 16 \uparrow \times 16) + 328 \times h + 112 \times 240^3$ $+ 5072 \times (e + 15) + 96 \times z$ $+ 32 \times m + 172 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16$ $+ 48 \times p + 36 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\}$ $+ 68 \times G + 64^1 + 368^2$ $+ ((\downarrow (\uparrow (g \div 8) \uparrow + 3) \div 4 \downarrow) \times 4) \times m$ $\} \div 1024 \uparrow$</p> <p>64-bit mode 500×1024 $+ (\uparrow 472 \times g \div 16 \uparrow \times 16) + 344 \times h$ $+ (\uparrow 136 \times 240 \div 16 \uparrow \times 16)^4$ $+ 9424 \times (e + 15) + 144 \times z$ $+ 48 \times m + 336 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16$ $+ 32 \times p + 72 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\}$ $+ 68 \times G + 64^1 + 448^2$ $+ ((\downarrow (\uparrow (g \div 8) \uparrow + 7) \div 8 \downarrow) \times 8) \times m$</p>
Formula 3 (bytes)	<p>32-bit mode $(32 + 16 \times z) \times (G + 1) + 16$</p> <p>64-bit mode $(48 + 32 \times z) \times (G + 1) + 16$</p>
Formula 4 (bytes)	<p>32-bit mode $88 + 52 \times H + 3400$</p> <p>64-bit mode $112 + 56 \times H + 7200$</p>
Formula 5 (bytes)	<p>32-bit mode $(32 + 16 \times z) \times P + 16$</p> <p>64-bit mode $(48 + 32 \times z) \times P + 16$</p>

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 When you specify `v6compatible` or `v7compatible` in the `pd_sysdef_default_option` operand, use $112 \times \text{MAX}(r, i \times 1.2)$ for calculating instead of 112×240 .

⁴ When you specify `v6compatible` or `v7compatible` in the

pd_sysdef_default_option operand, use $(\uparrow 136 \times \text{MAX}(r, (i \times 1.2)) \div 16 \uparrow \times 16)$ for calculating instead of $(\uparrow 136 \times 240 \div 16 \uparrow \times 16)$.

(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):

$$\text{Formula 1} + \uparrow \{(\uparrow (40 + (\text{value obtained by adding Formulas 2 through 4})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

For 64-bit mode (KB):

$$\text{Formula 1} + \uparrow \{(\uparrow (72 + (\text{value obtained by adding Formulas 2 through 5})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

For details about the variables used in these formulas, see (4) below.

Notes on Formulas 1 through 7

- If any of the following conditions is satisfied, add Formula 3:
 - Y is specified in the pd_rdarea_open_attribute_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
- If HiRDB External Data Access is installed, add Formula 5.
- If you omit the pd_max_commit_write_no operand (except cases in which v6compatible or v7compatible is specified in the pd_sysdef_default_option operand), or you specify a value other than 0 in the pd_sysdef_default_option operand, add Formula 7. However, if you have already added Formula 4, you do not need to add this formula.
- If you omit the pd_bes_shmpool_size operand, the following value is set:

For 32-bit mode:

$$\uparrow \{(\uparrow (40 + (\text{total of Formulas 2 through 5, and 7})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow + \uparrow \{(\uparrow (40 + \text{Formula 5}) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

For 64-bit mode:

$$\uparrow \{(\uparrow (72 + (\text{total of Formulas 2 through 5, and 7})) \div 512 \uparrow \times 512)\} \div 1024 \uparrow + \uparrow \{(\uparrow (72 + \text{Formula 5}) \div 512 \uparrow \times 512)\} \div 1024 \uparrow$$

Condition	Shared memory calculation formula
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)\} + 512 + 512^1 \\ & \quad \} \div 1024 \\ & + \uparrow \{72 + 8 \times v \times (8 + 3 \times w)\} \div 1024 \uparrow \\ & + \uparrow \{(\uparrow g \div 127 \uparrow + 1) \times 2048 + 128\} \div 1024 \uparrow \\ & + \uparrow \{ \\ & \quad \uparrow (28 + (\uparrow (32 + ((\uparrow g \div 127 \uparrow + 1) \times 2048 + 128)) \div 32 \uparrow \times 32)) \\ & \quad \div 128 \uparrow \times 128\} \\ & \quad \} \div 1024 \uparrow \\ & \quad M \\ & + \sum_{i=1} (N_i) \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)\} + 1024 + 512^1 \\ & \quad \} \div 1024 \\ & + \uparrow \{72 + 24 \times v \times (2 + w)\} \div 1024 \uparrow \\ & + \uparrow \{ \\ & \quad \uparrow (56 + (\uparrow (56 + ((\uparrow g \div 127 \uparrow + 1) \times 2048 + 128)) \div 32 \uparrow \times 32)) \\ & \quad \div 128 \uparrow \times 128 \\ & \quad \} \div 1024 \uparrow \\ & \quad M \\ & + \sum_{i=1} (N_i) \end{aligned} $

Condition	Shared memory calculation formula
Formula 2 (bytes)	<p>32-bit mode 500×1024 $+ (308 + 48^1) \times g + 328 \times h + 112 \times r^3$ $+ 5072 \times (e + 15) + 96 \times z$ $+ 32 \times m + 172 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16$ $+ 48 \times p + 48 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\}$ $+ 68 \times G + 144 \times F + 80 + 32 \times g + 64^2$ $+ ((\downarrow(\uparrow(g \div 8) \uparrow + 3) \div 4 \downarrow) \times 4) \times m$</p> <p>64-bit mode 500×1024 $+ (400 + 56^1) \times g + 344 \times h + 136 \times r^4$ $+ 9424 \times (e + 15) + 144 \times z$ $+ 48 \times m + 336 \times \{a \times (e + 3) + (e + 3) \times 2 + 22\} + 16$ $+ 64 \times p + 96 \times \{(e + 3) \times 2 + 1 + \text{MAX}(5, [e + 3] \div 10)\}$ $+ 68 \times G + 160 \times F + 96 + 48 \times g + 64^2$ $+ ((\downarrow(\uparrow(g \div 8) \uparrow + 7) \div 8 \downarrow) \times 8) \times m$</p>
Formula 3 (bytes)	<p>32-bit mode $\{[(\uparrow \uparrow g \div 8 \uparrow \div 4 \uparrow) \times 4] + 8\} \times (a \times [[e + 3] + e + 15])$</p> <p>64-bit mode $\{[(\uparrow \uparrow g \div 8 \uparrow \div 8 \uparrow) \times 8] + 8\} \times (a \times [e + 3] + e + 15)$</p>
Formula 4 (bytes)	<p>32-bit mode $(32 + 16 \times z) \times (e \times 2 + 7 + 1) + 16$</p> <p>64-bit mode $(48 + 32 \times z) \times (e \times 2 + 7 + 1) + 16$</p>
Formula 5 (bytes)	<p>32-bit mode $32 + 144 \times C + (6160 + 112 + 48 \times e) \times B$</p>
Formula 6 (bytes)	<p>32-bit mode $88 + 52 \times H + 3400$</p> <p>64-bit mode $112 + 56 \times H + 7200$</p>
Formula 7	<p>32-bit mode $(32 + 16 \times z) \times P + 16$</p> <p>64-bit mode $(48 + 32 \times z) \times P + 16$</p>

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 When you specify `v6compatible` or `v7compatible` in the `pd_sysdef_default_option` operand, use $112 \times \text{MAX}((i \times 1.2), r)$ for calculating instead of $112 \times r$.

⁴ When you specify `v6compatible` or `v7compatible` in the `pd_sysdef_default_option` operand, use $136 \times \text{MAX}((i \times 1.2), r)$ for calculating instead of $136 \times r$.

(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` operand¹

f: Total number of back-end servers

g: Value of `pd_max_rdarea_no` operand

h: Value of `pd_max_file_no` operand

i: Total number of indexes in the server (for a partitioning key index, this is the number of partitions in the server)

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 DML²

u: Total number of parameters for the plug-in functions used with DML²

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

B: Value of the `pd_max_foreign_server` operand

C: Total number of DBMSs supported by HiRDB External Data Access Adapters

- HiRDB External Data Access Adapters for HiRDB or XDM/RD E2 connection: 3
- HiRDB External Data Access Adapters for ORACLE connection: 1

If all HiRDB External Data Access Adapters are installed, the total number of supported DBMSs is 5.

D: Total number of HiRDB External Data Access Adapters for SQL exchanges

- HiRDB External Data Access Adapters for HiRDB or XDM/RD E2 connection: 512
- HiRDB External Data Access Adapters for ORACLE connection: 2048

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

P: Value of the `pd_max_commit_write_reclaim_no` operand

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 Table 15-9.

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.
-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 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> $\sum_{i=1}^n \{ \uparrow \{ 736 + 64 + (280 + 56^1) \times (P_i + 4) + (124 + 80^2 + 96 \times A \times M_i) \times U_i \} \div 4096 \uparrow \times 4096 + S_i \times \{ P_i + 4 + (U_i \times M_i \times A) \} \} \div 1024$ <p>64-bit mode</p> $\sum_{i=1}^n \{ \uparrow \{ 928 + 64 + (464 + 96^1) \times (P_i + 4) + (176 + 96^2 + 136 \times A \times M_i) \times U_i \} \div 4096 \uparrow \times 4096 + S_i \times \{ P_i + 4 + (U_i \times M_i \times A) \} \} \div 1024$

Formulas	Shared memory calculation formula (KB)
Formula 2	<p>32-bit mode</p> $\{ \lceil [(s \times 1024 \div 4) \div 8] + 112 \rceil \div 4096 \uparrow \times 4096 \times \lceil a \div (s \times 1024) \rceil \} \div 1024$ <p>64-bit mode</p> $\{ \lceil [(s \times 1024 \div 4) \div 8] + 144 \rceil \div 4096 \uparrow \times 4096 \times \lceil a \times (s \times 1024) \rceil \} \div 1024$

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 Table 15-10.

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
-r, -b	If the unit has an RDAREA for which -r is specified and which belongs to the same HA group, that RDAREA is used in the calculation.

pdbuffer statement option	Calculation condition
-i	If an RDAREA that contains the index specified with -i belongs to the same HA group, that RDAREA is used in the calculation.
-o	If there are any RDAREAs in the same HA group that are not specified with pddbuffer -r, they are used in the calculation.

If the pddbuffer operand is omitted, HiRDB calculates the shared memory area automatically, so it need not be estimated.

Shared memory calculation formula (KB)
<p>32-bit mode</p> $n \sum_{i=1} \{ (96 + ((736 \times (A + B)) + (272 \times (F + (8 \times (A + B)))) + 8 \times F \times (A + B) + 16) + H + D) + 2048 + G + (E \times F + (8 \times (A + B))) \} \div 1024$ <p>64-bit mode</p> $n \sum_{i=1} \{ ((144 + ((928 \times (A + B)) + (448 \times (F + (8 \times (A + B)))) + (16 \times F \times (A + B)))) + 16 + H + D) + 2048 + G + (E \times F + (8 \times (A + B))) \} \div 1024$

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 pddbuffer -p)

D: Add this value if the prefetch facility is used (pddbuffer -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 -</code>), MAX (page size of the RDAREA that stores the index specified with <code>-i</code> 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:

$$56 \times (F + (8 \times (A + B)))$$

In the 64-bit mode:

$$96 \times (F + (8 \times (A + B)))$$

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 memory on the basis of the value of `PDAGGR` 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

$$e + \uparrow d \div 4 \uparrow \times 4 + \uparrow (17 + 4 \times a + 4 \times b + c + d) \div 4 \uparrow \times 4 \times (N + 1)$$

(bytes)

15. Storage Requirements for HiRDB

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)

d: Length of operation area for set functions (see Table 15-11)

e: 32-bit mode: MAX ($4 \times N \times 24$, 16408)

64-bit mode: MAX ($8 \times N \times 40$, 32808)

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 (<i>p</i> , <i>s</i>)	$\uparrow (p + 1) \div 2 \uparrow$	$\uparrow (p + 7) \div 2 \uparrow^3$
FLOAT	8	10
SMALLFLT	4	6
INTERVAL YEAR TO DAY	5	8
INTERVAL HOUR TO SECOND	4	6
CHAR (<i>n</i>)	<i>n</i>	<i>n</i> + 3
VARCHAR (<i>n</i>)	<i>n</i> + 2	<i>n</i> + 5
NCHAR (<i>n</i>)	$2 \times n$	$2 \times n + 2$
NVARCHAR (<i>n</i>)	$2 \times n + 2$	$2 \times n + 4$
MCHAR (<i>n</i>)	<i>n</i>	<i>n</i> + 3
MVARCHAR (<i>n</i>)	<i>n</i> + 2	<i>n</i> + 5
DATE	4	6
TIME	3	6
BLOB (<i>n</i>)	—	

Column's data type	Column length	Length of operation area for set function ¹
BINARY (<i>n</i>)	$n + 2$	$n + 5$

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 operation area is 18.

(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

$a + 128$ (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

<p>32-bit mode</p> <p>a</p> $\sum_{i=1} (13 \times 1024 + 6 \times 1024 \times b + c)$ <p>64-bit mode</p> <p>a</p> $\sum_{i=1} (13 \times 1024 + 7 \times 1024 \times b + c)$ <p>(bytes)</p>
--

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 manual *HiRDB Version 8 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
1500 or less	Batch hash join		0.5
$1500 \times (\text{packet split count} \div 3)$ or less	Packet split Hash join	1-level packet split	1
$1500 \times (\text{packet split count} \div 3)^2$ or less		2-level packet split	2
Greater than $1500 \times (\text{packet split count} \div 3)^2$		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: $\text{MIN} \{ \lfloor (\text{size of hash table} \div 2) \div \text{page length of hash table} \rfloor, 64 \}$

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 1012	4096
1013 to 2036	8192

Maximum length of hash table row	Page length of hash table (bytes)
2037 to 4084	16384
4085 to 16360	32768
16361 to 32720	\uparrow (maximum length of hash table row + 48) \div 2048 \uparrow \times 2048

c: Maximum length of hash table row

For details about the length of a hash table row, see the manual *HiRDB Version 8 UAP Development Guide*.

(4) Procedure for obtaining the size of the memory required for foreign table retrieval

The following formula is used to determine the size of the memory required for foreign table retrieval (when the HiRDB External Data Access facility is used). You can check the SQL statements that retrieve from foreign tables with the access path display facility (`pdvwopt` command). When an SQL statement is executed on a foreign table, the memory requirement is 2,147,483,647 bytes.

Formula

$$a + 16 + 16 \times b + c$$

This is calculated when there are embedded variables or ? parameters in the SQL statement used for foreign table retrieval.

$$+ 16 + 16 \times e + d$$

(bytes)

a: Length of the SQL statement retrieving from the foreign table

b: Number of columns in the foreign table to be retrieved

c: If the foreign server's information definition contains the `pd_hb_ary_fec_num` operand:

- When the `pd_hb_ary_fec_num` operand is specified

Total column length of the foreign table (see Table 15-5) \times value of the `pd_hb_ary_fec_num` operand

- When the `pd_hb_ary_fec_num` operand is not specified

$\text{MAX}\{\text{total column length of the foreign table (see Table 15-5), value of the } \text{pd_sql_send_buff_size} \text{ operand} \times 2048\}$

If the foreign server's information definition does not contain the `pd_hb_ary_fec_num` operand:

Total column length of the foreign table (see Table 15-5)

d: Total array length of embedded variables or ? parameters in the SQL statement that performs the foreign table retrieval (see Table 15-5)

e: Number of embedded variables or ? parameters in the SQL statement that performs the foreign table retrieval

(5) 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.

(6) 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 18. *Determining Work Table File Size* because a work table is created.

Formula

$$\{100 + (a + 2) \times (\text{number of offset rows} + \text{number of rows specified in the } \text{LIMIT} \text{ clause})\} \times b$$

(bytes)

a: Row length

The row length cannot exceed 32,720 bytes. The row length is calculated with the

following formula:

$$\sum_{i=1}^m (A_i) + 2 \times m + 4 + c$$

(bytes)

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.

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 .

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 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 maintenance areas for the first n records

The number of maintenance areas for the first n records is calculated with the following formula:

1 + number of UNION [ALL] clause specifications

(7) 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

$$a \times 500 + (20 + 6) \times 800 + 16 \quad (\text{bytes})$$

a : Row length. To determine the row length, use the following formula:

$$\sum_{i=1}^m (A_i) + 4 \times (m + 2) + 12 + 4 + 8$$

(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.

A_i : 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, A_i corresponding to 1 that was added to m is 12 bytes.

(8) 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

<p>32-bit mode $(16 + 16 \times m) + a$ 64-bit mode $(16 + 24 \times m) + a$ (bytes)</p>
--

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

For details about the data length, see Table 15-5.

(b) When the PREPARE statement of the definition SQL is executed

Formula

<p>$SQL\ statement\ length + 20$ (bytes)</p>
--

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 or not `SELECT_APSL` is applied. For details about the access path display utility (`pdvwopt`), see the manual *HiRDB Version 8 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 8 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.

$$\begin{aligned} &\uparrow \{ \\ &\quad c \\ &\quad \sum_{i=1}^c (\text{actual length of BLOB input variable } i^1 \times 2 + 58) + \\ &\quad d \\ &\quad \sum_{j=1}^d (\text{specified length of BLOB output variable } j^2 + 26) \\ &\} \div 1024 \uparrow \end{aligned}$$

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:

$$256 \times \text{number of concurrently open cursors}$$

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

$$a + b \quad (\text{KB})$$

a: Maximum value from the following formula for BLOB input variables or output variables specified in one SQL statement:

$$\left\lceil \left\{ \sum_{i=1}^c (\text{actual length of BLOB input variable } i^* + 118 + 70 \times \text{number of output variables}) \right\} \div 1024 \right\rceil$$

* This is the actual length of BLOB data passed as embedded variables from the UAP to the HiRDB server.

This is the actual length of BLOB data passed as embedded variables from the UAP to the HiRDB server.

$$d \sum_{i=1} \{280 + 184 \times (\text{number of tables specified in } SQLi + 1)\}$$

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 PDBLKBUFSIZE operand	
		Omitted or 0	1 or greater
An array-type embedded variable is specified in the INTO clause of FETCH statement		Formula 1	
An array-type embedded variable is not specified in the INTO clause of FETCH statement	PDBLK operand is omitted or set to 1	—	Formula 2
	PDBLK operand is set to 2 or greater	Formula 1	

Legend:

— : Not applicable

Formula 1

$$\uparrow \{864 + 16 \times a + (6 \times a + 2 \times d + b) \times c\} \div 1024 \uparrow$$

(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

$$\text{MAX}(X_1, X_2)$$

(kilobytes)

X_1 : $\uparrow (864 + 22 \times a + 2 \times c + b) \div 1024 \uparrow$

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

Chapter

16. Determining RDAREA Size

This chapter explains how to determine the size of each type of RDAREA.

This chapter contains the following sections:

- 16.1 Determining the size of a user RDAREA
- 16.2 Determining the size of a data dictionary RDAREA
- 16.3 Determining the size of the master directory RDAREA
- 16.4 Determining the size of the data directory RDAREA
- 16.5 Determining the size of a data dictionary LOB RDAREA
- 16.6 Determining the size of a user LOB RDAREA
- 16.7 Determining the size of the registry RDAREA
- 16.8 Determining the size of the registry LOB RDAREA
- 16.9 Determining the size of the list 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

$$\begin{aligned} & \text{Size of user RDAREA (bytes)} \\ & = \text{Page length of the user RDAREA}^1 \times \text{total number of pages in the user RDAREA}^2 \end{aligned}$$

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.

$$\begin{aligned} & \text{Total number of pages in a user RDAREA} = \\ & a \\ & \sum_{i=1}^a \{ \uparrow Si \div f \uparrow + \uparrow Si \div g \uparrow \} \\ & c \\ & + \sum_{i=1}^c \{ \uparrow Ti \div e \uparrow \times e \} \\ & d \\ & + \sum_{i=1}^d \{ \uparrow (li + l) \div e \uparrow \times e \} + 6 \times (a + 1) + 2 \times \uparrow 20480 \div b \uparrow \end{aligned}$$

a: Number of HiRDB files constituting the user RDAREA

b: Page length of the user RDAREA (bytes)

c: Total number of tables in the user RDAREA

d: Total number of indexes to be stored in the user RDAREA

e: Segment size specified in the `create rdarea` statement for the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`)

f: $\downarrow \{b - 20\} \div \{(\uparrow e \div 32 \uparrow \times 8) + 56\} \downarrow$

$$g: \uparrow (125 \times b) \div (16 \times f) \uparrow \times f$$

S_i : Number of segments in each HiRDB file that is specified in the `create rdarea` statement for the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`)

T_i : Number of pages required to store each table (see 16.1.2 *Calculating the number of table storage pages*)

I_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 or not `FIX` is specified for the table in `CREATE TABLE`. The procedures are explained in (1) and (2) as follows. The variables used in the formulas 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

$$\text{Number of stored pages in table} = \frac{(P + \sum_{i=1}^n P S_i) \times g}{g - \sqrt{\frac{g \times h}{100}}} \quad (\text{Unit: pages})$$

(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

$$P = \left(\text{MIN} \left(255, \frac{b \times (100 - c)}{100} - 48, \frac{\sum_{i=1}^f d_i}{2} \times 2 + 8 + 2 \times f \right) \right)$$

Determine the value of b and c so that the following condition is satisfied:

$$\frac{b \times (100 - c)}{100} - 48 \geq \frac{\sum_{i=1}^f d_i}{2} \times 2 + 8 + 2 \times f$$

(b) How to obtain the value of PSi

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.

$$PS_i = a \times e_i / (b - 62)$$

(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.

$$Q = \left(\text{MIN}(255, \frac{b \times (100 - c)}{100} - 48, \frac{\sum_{i=1}^f d_i}{2} \times 2 + 6) \right)$$

Determine the value of b and c so that the following condition is satisfied:

$$\frac{b \times (100 - c)}{100} - 48 \geq \frac{\sum_{i=1}^f d_i}{2} \times 2 + 6$$

When you determine the value of b, make sure that the restrictions on the total length of columns that are described in the common rules for CREATE TABLE in the manual *HiRDB Version 8 SQL Reference* are also satisfied.

(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.

Table 16-1: List of data lengths

Classification	Data type			Data length (bytes)		
Numeric data	INTEGER			4		
	SMALLINT			2		
	LARGE DECIMAL(<i>m,n</i>) ¹			$\downarrow m/2 \downarrow + 1^2$		
	FLOAT or DOUBLE PRECISION			8		
	SMALLFLT or REAL			4		
Character data	CHARACTER(<i>n</i>)			n^3		
	VARCHAR(<i>n</i>) (variable-length character string)	$d \leq 255$	Element of repetition column	$d + 2$		
			Other	$d + 1$		
		$d \geq 256$		6		
	VARCHAR (<i>n</i>) 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			$d + 3$		

Classification	Data type			Data length (bytes)	
National character data	NCHAR(<i>n</i>) or NATIONAL CHARACTER(<i>n</i>)			$2 \times n^4$	
	NVARCHAR(<i>n</i>)	$d \leq 127$	Element of repetition column	$2 \times d + 2$	
			Other	$2 \times d + 1$	
		$d \geq 128$		6	
	NVARCHAR(<i>n</i>) 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		$2 \times d + 3$	

16. Determining RDAREA Size

Classification	Data type			Data length (bytes)	
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	
Time interval data	INTERVAL HOUR TO SECOND			4	
Timestamp data	TIMESTAMP(n)			$7 + (n \div 2)$	
Large object data	BLOB			9	
Binary data	BINARY(n)	$n \leq 255$		$d + 3$	
		$n \geq 256$	Branching ⁵		15
			Not branching ⁵		$d + 3$

d : Actual data length (in characters)

m, n : Positive integer

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 " $\lfloor k/2 \rfloor + 2$ ", 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):

$$32717 < (a + \text{number of columns in table} \times 2 + 8)$$

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):

$$32717 < (a + \text{number of columns in table} \times 2 + 8)$$

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):

$$32717 < (a + \text{number of columns in table} \times 2 + 8)$$

⁵ Normally the calculation assumes that there is no branching. Branch only when the following condition is satisfied:

$BL > \text{page length} - 50$ $BL \text{ (bytes)} = \sum_{i=1}^f d_i + 2 \times f + 6$

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. Furthermore, add the value obtained from the following formula to the value of P :

$$SPN = SPN1 + SPN2 + SPN3 + SPN4$$

Columns other than BINARY type
 $SPN1 =$
 f
 $\sum_{i=1}^f \uparrow \text{value of branch } di \div (b - 61) \uparrow \times a \times SF$

BINARY type columns
 • Number of page branches due to INSERT
 $SPN2 =$
 f
 $\{ \sum_{i=1}^f \downarrow \text{value of branch } di \div (b - 59) \downarrow \times a + A \} \times SF$

• Number of page branches due to pdload or pdrorg
 When value of branch $di > (b - 2853) \div 255$
 $SPN3 =$
 f
 $\uparrow \{ \sum_{i=1}^f (\text{value of branch } di + 11) \times a \} \div (b - 48) \uparrow \times SF$

When value of branch $di \leq (b - 2853) \div 255$
 $SPN4 =$
 $\uparrow a \div 255 \uparrow \times SF$

The formula for A is as follows:

$$A = \left(\begin{array}{c} \uparrow \\ \text{MIN} \left(\begin{array}{c} 255, \\ \downarrow \sum_{i=1}^f \{ d_i - (b-59) \times \frac{\text{Value of branch } di}{b-59} + 11 \} \downarrow \end{array} \right) \\ \uparrow \end{array} \right)$$

f : Number of branch columns

SF : 1.3

Use a value greater than 1.3 in the following cases:

- A large number of columns with the abstract data type will be updated.

- A large number of repetition columns will be updated in such a manner that the elements' data lengths increase or the number of elements increases.
- A large number of VARCHAR, NVARCHAR, MVARCHAR, or BINARY columns will be updated in such a manner that the updates will increase the data length.
- A large number of BINARY columns will be updated in such a manner that the updates will increase the data length.
- A large number of columns with data suppressed will be updated in such a manner that their data lengths increase.
- A large number of columns other than those above will be updated in such a manner that the NULL values are changed to non-NULL values.

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 (<i>n</i>)	$d \geq 256$	$d + 2$
	No-split option specified	0
NVARCHAR (<i>n</i>)	$d \geq 128$	$2 \times d + 2$
	No-split option specified	0
MVARCHAR (<i>n</i>)	$d \geq 256$	$d + 2$
	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 d_i of an abstract data type column.

Formula

$$d_i = \sum_{k=1}^h ADT_k + 5$$

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:

$$ADT_k = \sum_{i=1}^m att_j + 10 + 2 \times m$$

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.

Assign the value of the corresponding $atte_j$ to the following formula and add branch row storage pages $ADTLS$ to P :

$$ADTLS = \sum_{I=1}^h \uparrow atte_j \div (b - 62) \uparrow \times a$$

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

Data type	Condition	Data length att _j (bytes)	Data length of branch section atte _j (bytes)
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, add the value obtained from the following formula to *P*:

$$\uparrow \sum_{i=1}^m \{ es_i \times en_i + 14 \times (en_i - 1) \} \div (b - 62) \uparrow \times a$$

m: Number of repetition columns with the variable-length character string type that satisfy the condition shown in Table 16-4.

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 (1024 ÷ 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.

$$S_{sn} = \uparrow S_n \div S_{rn} \uparrow \times S_{rn}$$

$$S_{rn}: \uparrow 1024 \div D_{vn} \uparrow$$

D_{vn} : Number of RDAREA partitions

3. Estimate the number of segments S used per RDAREA, making provision for a surplus.

$$S = \uparrow (S_{sn} \times K) \div S_{rn} \uparrow \times S_{rn}$$

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

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

$$\frac{10000 - \left(\frac{8192 \times (100 - 30)}{100} - 48 \right)}{14 \times 2 + 8 + 2 \times 6} = 85$$

3. Calculation of the number of table storage pages

$$\frac{85 \times 100}{100 - \frac{100 \times 40}{100}} = 142 \text{ pages}$$

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

$$\frac{10000 - \left(\frac{8192 \times (100 - 30)}{100} - 48 \right)}{14 \times 2 + 6} = 60$$

3. Calculation of the number of table storage pages

$$\begin{array}{c} \uparrow \\ \frac{60 \times 100}{100} \\ \downarrow \\ \frac{100 \times 40}{100} \\ \downarrow \\ \uparrow \end{array} = \underline{100\text{pages}}$$

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 manual *HiRDB Version 8 System Operation Guide*.

(1) Calculation procedure

The following formula is used to calculate the number of pages needed to store an index.

Formula

$$\text{Number-of-index-storage-pages} = \sum_{i=1}^n P_i + P_d$$

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

$$\begin{aligned}
 P_1 = & \frac{c - h}{\text{MAX} \left(1, \frac{\frac{a \times (100 - b)}{100} - g - 68}{18 + g + 4 \times d} \right)} \\
 + & \frac{e}{\text{MAX} \left(1, \frac{\frac{a \times (100 - b)}{100} - g - 68}{18 + g} \right)} \\
 + & \frac{h}{\text{MAX} \left(1, \frac{\frac{a \times (100 - b)}{100} - g - 68}{14 + g} \right)} \\
 P_{i+1} = & \frac{P_i}{\text{MAX} \left(1, \frac{\frac{a \times (100 - b)}{100} - g - 68}{14 + g} + 1 \right)}
 \end{aligned}$$

Formula 2

$$P d = \left(\left(\frac{f}{\frac{a \times 95}{100} - 70} + 1 \right) \times e \right)$$

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:

$$\text{Number of duplicated elements} = \lfloor (\lfloor a \times 0.95 \rfloor - 82) \div 4 \rfloor - 1$$

(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

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
Numeric data	INTEGER	4	5	6	N	5	7
	SMALLINT	2	3	4	N	3	5
	LARGE DECIMAL(<i>m,n</i>) ³	$\downarrow m \div 2$ $\downarrow + 1$	$\downarrow m \div 2$ $2 \downarrow + 2$	$\downarrow m \div 2$ $2 \downarrow + 3$	N	$\downarrow m \div 2$ $2 \downarrow + 2$	$\downarrow m \div 2$ $2 \downarrow + 4$
	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
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>)	$7 + (p \div 2)$	$8 + (p \div 2)$	$9 + (p \div 2)$	—	$8 + (p \div 2)$	$10 + (p \div 2)$

Legend:

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.

(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
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

$$\text{DB storage key length}(g) = \lceil 5 \div 4 \rceil \times 4 = 8$$

$$\begin{aligned}
 P_1 = & \left(\frac{10000 - 10000}{\text{MAX} \left(1, \frac{\frac{8192 \times (100 - 30)}{100} - 8 - 68}{18 + 8 + 4 \times 1} \right)} \right) \\
 + & \left(\frac{0}{\text{MAX} \left(1, \frac{\frac{8192 \times (100 - 30)}{100} - 8 - 68}{18 + 8} \right)} \right) \\
 + & \left(\frac{10000}{\text{MAX} \left(1, \frac{\frac{8192 \times (100 - 30)}{100} - 8 - 68}{14 + 8} \right)} \right) \\
 = & 0 + 0 + 39 = 39 \\
 P_2 = & \left(\frac{39}{\text{MAX} \left(1, \frac{\frac{8192 \times (100 - 30)}{100} - 8 - 68}{14 + 8} + 1 \right)} \right) = 1
 \end{aligned}$$

$P_d=0$ (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

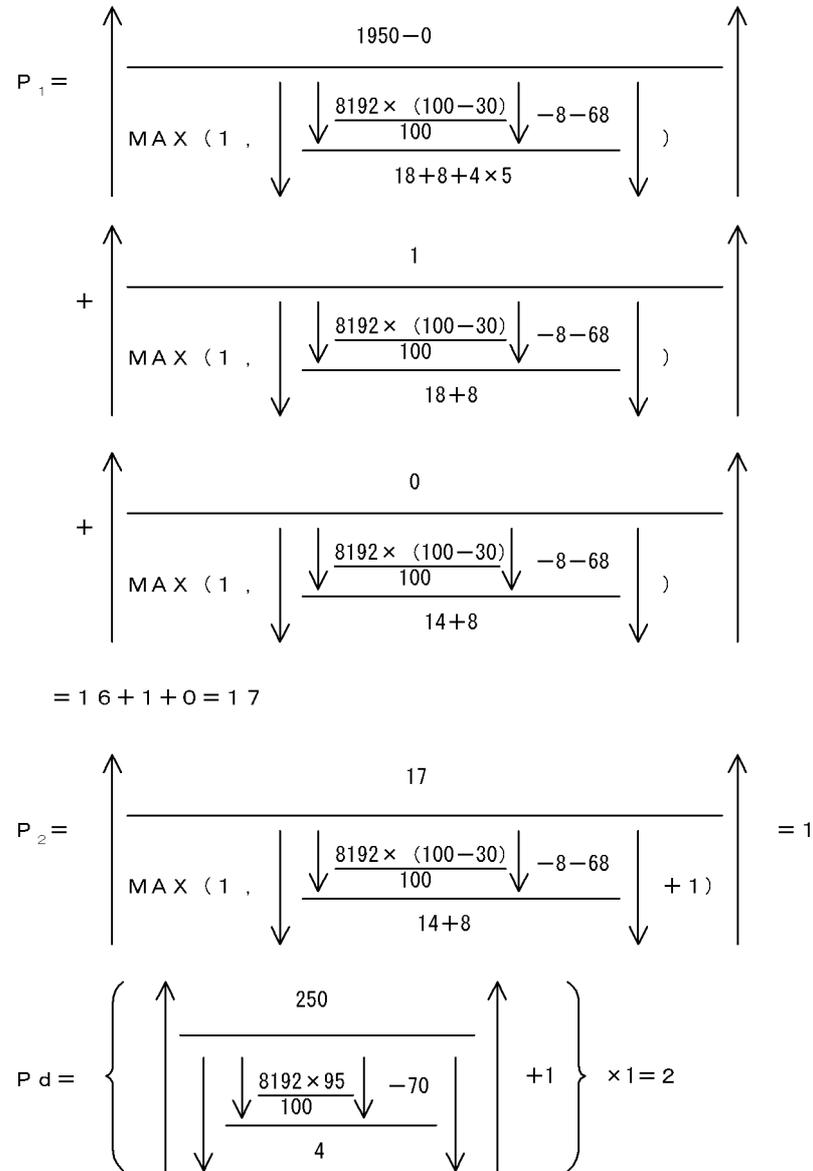
Obtain the number of index storage pages for the STOCK table shown in Example 1 when the PNAME column is used as the index (with duplicated key values).

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: $(10000 - 250)/5 = 1950$ (Average number of duplicates: 5)

Formula

$$\text{DB storage key length}(g) = \lceil (2 \times 4) \div 4 \rceil \times 4 = 8$$



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
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
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: $d = (10000 - 1000) \div 62 = 146$.

16. Determining RDAREA Size

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): $f = 1000/2 = 500$
5. DB storage key length (g) of the `SEX` and `YEAR_JOINED` columns:
 $g = \lceil (4 + 1 + 5)/4 \rceil \times 4 = 12$

$$\begin{aligned}
 P_1 &= \left(\text{MAX} \left(1, \frac{\frac{8192 \times (100-30)}{100} - 12 - 68}{18+12+4 \times 146} \right) \right) \times 62 - 0 \\
 &+ \left(\text{MAX} \left(1, \frac{\frac{8192 \times (100-30)}{100} - 12 - 68}{18+12} \right) \right) \times 2 \\
 &+ \left(\text{MAX} \left(1, \frac{\frac{8192 \times (100-30)}{100} - 12 - 68}{14+12} \right) \right) \times 0 \\
 &= 7 + 1 + 0 = 8 \\
 P_2 &= \left(\text{MAX} \left(1, \frac{\frac{8192 \times (100-30)}{100} - 12 - 68}{14+12} + 1 \right) \right) \times 8 = 1 \\
 P_d &= \left\{ \left(\text{MAX} \left(1, \frac{\frac{8192 \times 95}{100} - 70}{4} \right) + 1 \right) \times 2 \right\} = 4
 \end{aligned}$$

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 a data dictionary RDAREA (bytes)
 $= a \times b \times 1.3 + c \times 125 + 1600000$
 Size of data dictionary RDAREA = $a \times b \times 1.3 + c \times 125 + 1600000$
 (bytes)

a: Page length of the data dictionary RDAREA¹

b: Total number of pages in the data dictionary RDAREA²

c: Segment size of the data dictionary RDAREA³

¹ This is the page length specified in the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`).

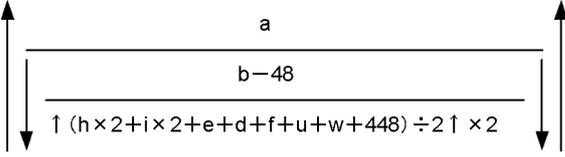
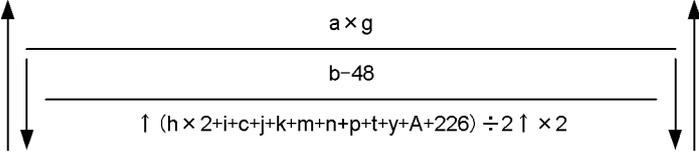
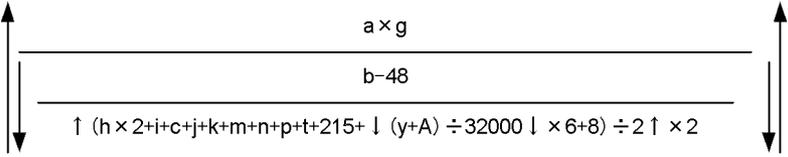
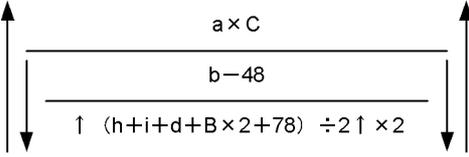
² This is the number of table storage pages + number of index storage pages (see (2) and (3) as follows).

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 21.

(a) Formula 1

Dictionary table name	Formula
SQL_TABLES	
SQL_COLUMNS	<p>• DEFAULT clause is not specified, and the default data length specified in the DEFAULT clause is 255 bytes or less</p>  <p>• Default data length specified in the DEFAULT clause is 256 bytes or greater</p>  <p>+ a × g × ↑ (w + y) ÷ (b - 61) ↑</p>
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)
- u*: Average length of foreign server name (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 on how to calculate the base length, see the section on data codes and data lengths set in the SQL descriptor area in the manual *HiRDB Version 8 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 RDB': 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

(b) Formula 2

Dictionary table name	Formula
SQL_INDEXES	$\frac{c}{a-48} \uparrow (f \times 2 + b + d + 2 \times k + j + m + n + 127) \div 2 \uparrow \times 2$
SQL_INDEX_COLINF	$\frac{c \times k}{a-48} \uparrow (f + b + d + g + 35) \div 2 \uparrow \times 2$
SQL_DIV_INDEX	$\frac{c \times i}{a-48} \uparrow (f + b + d + j + 44) \div 2 \uparrow \times 2$
SQL_EXCEPT	$\frac{c \times e}{a-48} \uparrow \{f + b + d + (k \times 7 - 2) + 33\} \div 2 \uparrow \times 2$

Dictionary table name	Formula
SQL_INDEX_DATATYPE	$\uparrow (f \times 2 + b + d + g + q + r + 43) \div 2 \uparrow \times 2$
SQL_INDEX_FUNCTION	$\uparrow (f \times 2 + b + d + g + p + 38) \div 2 \uparrow \times 2$

- 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

(c) Formula 3

Dictionary table name	Formula
SQL_TABLE_PRIVILEGES	$\begin{array}{c} d \\ \hline a-48 \\ \hline \uparrow (3 \times b + c + 54) \div 2 \uparrow \times 2 \end{array}$
SQL_RDAREA_PRIVILEGES	$\begin{array}{c} e \\ \hline a-48 \\ \hline \uparrow (b + f + 36) \div 2 \uparrow \times 2 \end{array}$
SQL_VIEW_TABLE_USAGE	$\begin{array}{c} h \\ \hline a-48 \\ \hline \uparrow (2 \times b + 2 \times c + 41) \div 2 \uparrow \times 2 \end{array}$
SQL_VIEWS	$\begin{array}{c} h \\ \hline a-48 \\ \hline \uparrow (b + c + 43) \div 2 \uparrow \times 2 \end{array} + h \times \begin{array}{c} g \\ \hline a-48 \end{array}$
SQL_VIEW_DEF*	$\begin{array}{c} h \\ \hline a-48 \\ \hline \uparrow (b + c + 28) \div 2 \uparrow \times 2 \end{array} + h \times \begin{array}{c} 150 \times i + g + 1000 \\ \hline a-48 \end{array}$

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

i: Average number of columns for a view table

* These tables are used by the system.

(d) Formula 4

Data dictionary table name	Formula
SQL_REFERENTIAL_CONSTRAINTS	

$$E: \{e \times h + 2 \times h + (h - 1)\} + 1$$

$$F: \{e \times i + 2 \times i + (i - 1)\} + 1$$

$$G: \{2 \times h + (h - 1)\} + 1$$

$$H: \{2 \times i + (i - 1)\} + 1$$

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	$\frac{f}{a-48} \times \frac{1(c+g+36) \div 2 \uparrow \times 2}{1}$
SQL_RDAREAS	$\frac{b}{a-48} \times \frac{1(c \times 2 + 99) \div 2 \uparrow \times 2}{1}$
SQL_USERS	$\frac{e}{a-48} \times \frac{1(d+h+47) \div 2 \uparrow \times 2}{1}$

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)

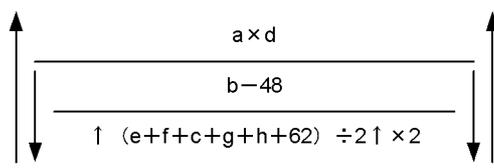
e: Total number of schemas

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*	 $a \times d$ $b - 48$ $\uparrow (e+f+c+g+h+62) \div 2 \uparrow \times 2$

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)

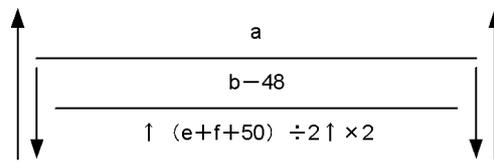
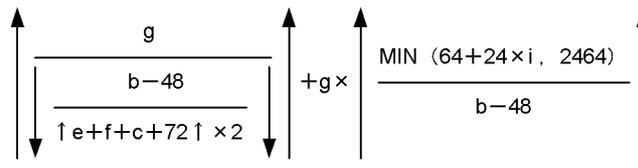
f: Average length of table identifiers (bytes)

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	 a $b - 48$ $\uparrow (e+f+50) \div 2 \uparrow \times 2$
SQL_COLUMN_STATISTICS ¹	 g $b - 48$ $\uparrow e+f+c+72 \uparrow \times 2$ $+g \times \frac{\text{MIN}(64+24 \times i, 2464)}{b-48}$

Dictionary table name	Formula
SQL_INDEX_STATISTICS ¹	

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 collected²

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	

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	<p> a $b-48$ $\uparrow (c+d \times 2 + e + k + m + n + p + q + r + 314) \div 2 \uparrow \times 2$ </p> <p> 174 $b-48$ 310 </p>
SQL_ROUTINE_RESOURCES	<p> $a \times i$ $b-48$ $\uparrow (c+d \times 2 + e + g + h + k + 49) \div 2 \uparrow \times 2$ </p>
SQL_ROUTINE_PARAMS	<p> $a \times j$ $b-48$ $\uparrow (c+d \times 2 + e + f + k + s + t + 140) \div 2 \uparrow \times 2$ </p> <p> 1044 $b-48$ 161 </p>

- 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 names¹ (bytes)
- f*: Average length of parameter names (bytes)
- g*: Average length of authorization identifiers of resource² owners (bytes)
- h*: Average length of resource² names (bytes)
- i*: Average number of resources² 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*.

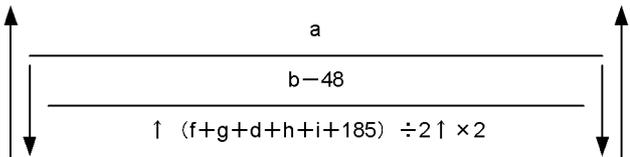
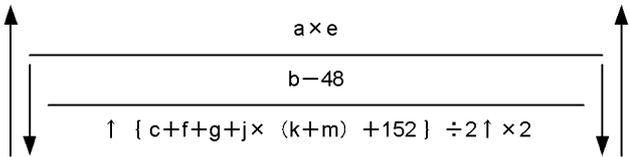
² Resources include tables and indexes.

(j) Formula 10

Dictionary table name	Formula
SQL_ALIASES	

- a*: Total number of tables for which aliases are defined
- b*: Page length of the data dictionary RDAREA (bytes)
- c*: Average length of authorization identifiers of tables for which aliases are defined (bytes)
- d*: Average length of table aliases (bytes)
- e*: Average length of RD-node names (bytes)
- f*: Average length of base table names (bytes)

(k) Formula 11

Dictionary table name	Formula
SQL_DATATYPES	 a $b-48$ $\uparrow (f+g+d+h+i+185) \div 2 \uparrow \times 2$
SQL_DATATYPE_DESCRIPTOR	 $a \times e$ $b-48$ $\uparrow \{ c+f+g+j \times (k+m) + 152 \} \div 2 \uparrow \times 2$

- 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)

(I) Formula 12

Dictionary table name	Formula
SQL_PLUGINS	$\frac{k}{b-48} \uparrow (m+d \times 2+n+o+p+66) \div 2 \uparrow \times 2$
SQL_PLUGIN_ROUTINES	$\frac{a \times k}{b-48} \uparrow (c+d+e+m+q+r+s+47) \div 2 \uparrow \times 2$
SQL_PLUGIN_ROUTINE_PARAMS	$\frac{a \times i \times k}{b-48} \uparrow (d \times 2+e+m+f+n+t+u+v+140) \div 2 \uparrow \times 2$

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*.

(m) Formula 13

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

(n) Formula 14

Dictionary table name	Formula
SQL_INDEX_RESOURCES	$\frac{a}{b-48} \uparrow (c \times 2 + d + e + 35) \div 2 \uparrow \times 2$
SQL_TYPE_RESOURCES	$\frac{g+h}{b-48} \uparrow (c \times 2 + e \times 2 + 35) \div 2 \uparrow \times 2$
SQL_TABLE_RESOURCES	$\frac{i}{b-48} \uparrow (c \times 2 + e + j + 35) \div 2 \uparrow \times 2$

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)

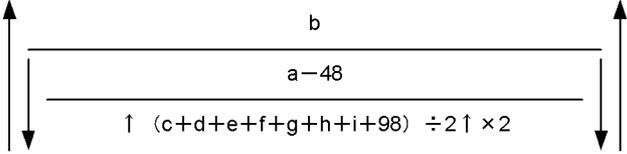
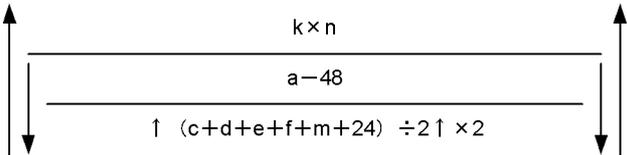
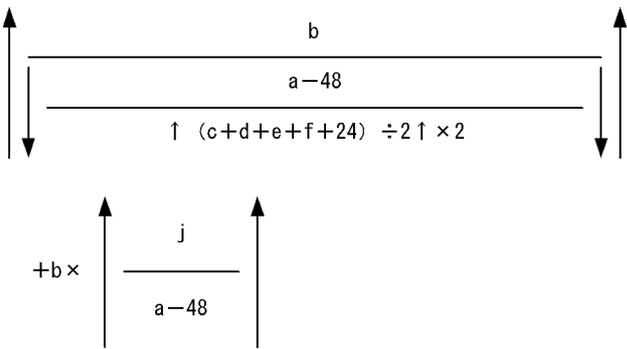
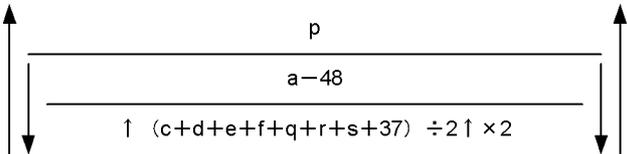
g: Total number of attributes defined for abstract data types

h: Total number of abstract data types defined as subtypes

i: Total number of abstract data types

j: Average length of table identifier (bytes)

(o) Formula 16

Dictionary table name	Formula
SQL_TRIGGERS	 $\frac{b}{a-48} \uparrow (c+d+e+f+g+h+i+98) \div 2 \uparrow \times 2$
SQL_TRIGGER_COLUMNS	 $\frac{k \times n}{a-48} \uparrow (c+d+e+f+m+24) \div 2 \uparrow \times 2$
SQL_TRIGGER_DEF_SOURCE	 $\frac{b}{a-48} \uparrow (c+d+e+f+24) \div 2 \uparrow \times 2$ $+b \times \frac{j}{a-48}$
SQL_TRIGGER_USAGE	 $\frac{p}{a-48} \uparrow (c+d+e+f+q+r+s+37) \div 2 \uparrow \times 2$

a: Data dictionary RDAREA page length (bytes)

b: Total number of trigger definitions

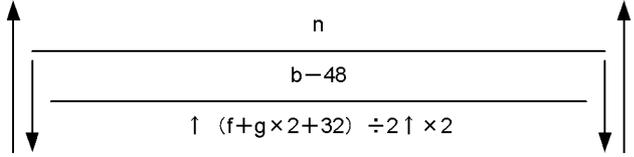
c: Average length of trigger authorization identifiers (bytes)

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)

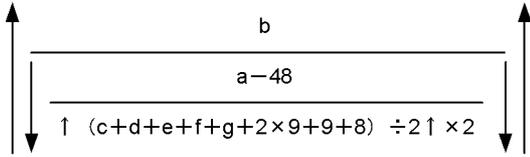
(p) Formula 17

Dictionary table name	Formula
SQL_PARTKEY	
SQL_PARTKEY_DIVISION	

Dictionary table name	Formula
SQL_DIV_TYPE	 $\frac{n}{b-48} \times \left(\frac{f+g \times 2+32}{2} \right) \times 2$

- 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)
- n*: Number of matrix partitioning tables that combine hash partitioning and key range partitioning with boundary values specified

(q) Formula 18

Dictionary table name	Formula
SQL_AUDITS	 $\frac{b}{a-48} \times \left(\frac{c+d+e+f+g+2 \times 9+9+8}{2} \right) \times 2$

- 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)

(r) Formula 19

Dictionary table name	Formula
SQL_AUDITS_REGULARIZE	$\frac{(c+d+e+2 \times 5+104+8) \div 2 \uparrow \times 2}{a-48} \times b$

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)

(s) Formula 20

Data dictionary table name	Formula
SQL_KEYCOLUMN_USAGE	$\frac{(c+d+e+f+g+22) \div 2 \uparrow \times 2}{a-48} \times b$

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 21

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: Value obtained from the formula shown below, where all the variables are specified in the check constraint search conditions:

In 32-bit mode: $S \times 36 + T + (U1 + U2 + U3) \times 48 + V \times 128 + 1000$

In 64-bit mode: $S \times 72 + T + (U1 + U2 + U3) \times 96 + V \times 184 + 1400$

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 22

Data dictionary table name	Formula
SQL_SYSPARAMS	2

(v) Formula 23

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 24

Dictionary table name	Formula
SQL_FOREIGN_SERVERS	
SQL_USER_MAPPINGS	

Dictionary table name	Formula
SQL_USAGE_PRIVILEGES*	

a: Number of foreign servers

b: Data dictionary RDAREA page length (bytes)

c: Average length of authorization identifiers (bytes)

d: Average length of foreign server names (bytes)

e: Average length of DBMS names used in foreign servers (bytes)

f: Average length of version identifiers of DBMSs used in foreign servers (bytes)

g: Total number of user mappings

h: Average length of user passwords used by foreign servers (bytes)

k: Number of USAGE privilege definitions

If *n* people are granted this privilege for a foreign server, this would be the number of foreign servers $\times n$.

* These are tables used by the system.

(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).

16. Determining RDAREA Size

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 g^1)	Number of key types (Variable c^1)	Average number of key duplicates (Variable d^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		b
SQL_INDEXES	10	$d + e + 2$	a	$\uparrow h \div a \uparrow$
	11	$d + i + 2$	h	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 USER USED BY operand of the database initialization utility (pdinit)	Average number of RDAREA access privileges used per user
	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

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_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		

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_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
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_ALIASES	42	$d + v + 4$	Total number of table aliases to be created	1

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_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		

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_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

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_INDEX_FUNCTIONS	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_USER_MAPPINGS	89	$d + FSRV$	Number of user mappings	1
SQL_TRIGGERS	90	$d + e + 2 + 16$	Number of triggers	1
	91	$d + TRIG + 2$		
	92	$d + t + 2$		
	93	4		
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_DEFINITION_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

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_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

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_KEYCOLUMN_US AGE	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
SQL_TABLE_CONSTR AINTS	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_COLUMN S	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_S AME_USERS	124	$d + e + 2$	Number of public view tables \times number of authorization identifiers	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 names² (bytes)

u: Total number of specific names²

v: Average length of table aliases (bytes)

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)

FSRV: Average length of foreign server names (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)

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:

$$\bullet \lceil \text{key-length} \div 4 \rceil \times 4$$

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

$\text{Size of data dictionary RDAREA} = (\lceil (a \times 1.3) \div b \rceil) \times b \times 4096$ (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	$\frac{((a+c+e+g) \times (b+1) + 120) \times 30}{96}$
Formula 2	$\frac{((a+c+e+g) \times (b+1) + 120) \times 30}{9}$

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:

$$\text{Number of pages for index storage for the data dictionary tables}^* \times 2$$

* See 16.1.3 *Calculating the number of index storage pages* 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 CREATE INDEX 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

* 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) $= \left\{ \begin{aligned} &\uparrow(a+2) \div 800 \uparrow \times 51 + \uparrow(b+120) \div 6000 \uparrow \times 51 + \uparrow(c+240) \div 6000 \uparrow \times 51 \\ &+ \uparrow(d+240) \div 6000 \uparrow \times 51 + \uparrow e \div 6400 \uparrow \times 51 + 2 + 6 \times n \\ &\}^1 \times 4096^2 \end{aligned} \right.$
--

a: Total number of data dictionary RDAREAs + total number of user 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

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)

$$= \left\{ \sum_{i=1}^e (g_i + 86) + \sum_{j=1}^f (p_j + 86) \right\} \div 3000 \uparrow \times 51 + 6 \times n + 1 \}^1 \times 4096^2$$

$$g_i = \left\{ (5 \times a_i + 2 \times b_i + 2 \times c_i + 48) \div 32 \right\} \uparrow$$

$$p_j = \left\{ (d_j + 12) \div 16 \right\} \uparrow$$

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)

$$= \left\{ \begin{array}{l} a \\ \left[\sum_{i=1}^a \left(S_i \div 64000 \uparrow \times 96 + 7 + 3 \times (a - 1) \right) \right]^1 \\ + \\ b \\ \left[\sum_{j=1}^b (C_j + 1024) \div 8192 \uparrow \right]^2 \\ \}^3 \times 8192^4 \end{array} \right.$$

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

$$\begin{aligned} &\text{Size of data dictionary LOB RDAREA for object storage (bytes)} \\ &= \{ \\ &\quad a \\ &\quad [\sum_{i=1}^a \lceil S_i \div 64000 \rceil \times 96 + 7 + 3 \times (a - 1)]^1 \\ &\quad + \\ &\quad b \\ &\quad [\sum_{j=1}^b \lceil (C_j + 1024) \div 8192 \rceil]^2 \\ &\quad \}^3 \times 8192^4 + 500000^5 \end{aligned}$$

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

$$\begin{aligned}
 &QO_i \text{ (bytes)} = \\
 &a \\
 &\sum_{i=1} \{ \\
 &1840 + 46 \times RCN + 298 \times Si + 20 \times Pi + 1138 \times Ti + 76 \times Ti \times Di + 80 \times Ci + 40 \times Li + 534 \times Wi \\
 &+ 20 \times Ki + Li + 8 \times TCi + 656 \times Di + 48 \times nFF + 100 \times nFP + 148 \times nFC + 696 \times nPFF \\
 &+ 16 \times (nAT + nPAT) + 20 \times nCAT + 28 \times (nAF + nCAF) + 20 \times (nAA + nPAA + nCAA) \\
 &+ 1057 \times nSPA + 120 \times nSPP + 287 \times nSFF + 8 \times nSFP + 813 \times nJFC + 20 \times nJFP \\
 &[+ 60 \times nFT + nFTS + 32 \times nFTQ]^1 \\
 &[+ 1057 \times nTR + 120 \times (nTSN + nTSO) + 20 \times (nTCN + nTCO)]^2 \\
 &[+ 760 + 376 \times RCC + 1880 \times RCT]^3 \\
 &[+ 32 \times Si + 16]^4 \\
 &[+ \uparrow (42 \times SiT) + \{ 52 + 152 \times (SiTA + SiSA + SiNA) \times (SiT + SiS + SiN) \} \uparrow]^5 \\
 &\}
 \end{aligned}$$

a : Number of SQL statements in stored procedures

¹ Add this formula if you specify foreign tables.

² Add this formula if you use triggers.

³ Add this formula if you use referential constraints.

⁴ This is the formula for determining the length of the Column Name Descriptor Area (SQLCND A). Add it for dynamic SQL statements.

⁵ This is the formula for determining the length of the Type Name Descriptor Area (SQLTND A). 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:

$$\begin{aligned}
 PR \text{ (bytes)} = & \\
 & a \\
 & \sum_{i=1} \{ \\
 & 600 + 28 \times sRi + 32 \times (sRUi + sDi) + 56 \times sSXi + sCUi + sSi + sPi + sLA \\
 & + sKi + sL + 80 \times sWi + 24 \times sCM + 32 \times sCCR + 2 \times sDCR + 60 \times sCHD + 72 \times sDHD + 64 \times sHCN \\
 & + 8 \times sCHD \times sHCN + 48 \times nRFF + 100 \times nRFP + 148 \times nRFC + 200 \times nPRFF + 8 \times nPRFP \\
 & + 196 \times nPA + 64 \times nPP + 36 \times nPPI + 20 \times nPPO + 200 \times nPPA + 8 \times nPPP + 20 \times nAR + 48 \times nARA \\
 & + 16 \times nRPAT + 20 \times nCAT + 28 \times (nRPAF + nRCAF) + 20 \times (nRPAA + nRCAA) + 287 \times nRSFF \\
 & + 8 \times nRSFP + 813 \times nPJA + 20 \times nPJP + 813 \times nRJFC + 20 \times nRJFP \\
 & [+ 28 \times (nTSN \times 2 + nTSO)]^* \\
 & \}
 \end{aligned}$$

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:

$$\begin{aligned}
 PR \text{ (bytes)} = & \\
 & a \\
 & \sum_{i=1} \{ 240 + 608 \times RCC + (5120 + 100 \times RD_i + 256 \times RI_i) \times RCP \times RCT \}
 \end{aligned}$$

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>RCN</i>	Total number of tables and indexes used by SQL objects
<i>Si</i>	Number of retrieval items in SQL statements (if the columns specified by SQL statements are index columns, the number of those columns)
<i>Pi</i>	Number of embedded variables or parameters in SQL statements
<i>Ti</i>	Number of table names in SQL statements
<i>Ci</i>	Number of column names in SQL statements
<i>TCi</i>	Number of table composition columns in SQL statements
<i>Wi</i>	Number of logical operators in SQL statements ¹
<i>Ki</i>	Number of literals in SQL statements ¹
<i>Li</i>	Total length of literals in SQL statements ¹ (bytes)
<i>Ii</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 ¹
<i>nFP</i>	Number of function call parameters in SQL statements ¹

Variable name	Explanation
<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 ¹
<i>nSFP</i>	Total number of system definition scalar function arguments in SQL statements ¹
<i>nJFC</i>	Number of Java function calls in SQL statements
<i>nJFP</i>	Total number of Java function arguments 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>nFT</i>	Number of foreign tables in SQL statements
<i>nFTS</i>	Length of SQL statements that retrieve from foreign tables ²
<i>nFTQ</i>	Number of embedded variables or ? parameters in SQL statements that retrieve from foreign tables ²
<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

16. Determining RDAREA Size

Variable name	Explanation
<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
<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>RD_i</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>sR_i</i>	Number of SQL parameters in procedures and functions (count SQL parameters specified with INOUT twice)
<i>sRU_i</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>sD_i</i>	Total number of SQL variables (<code>declare</code>) in procedures, functions, and triggered SQL statements
<i>sSX_i</i>	Total number of <code>SQLCODE</code> and <code>SQLCOUNT</code> variables in procedures, functions, and triggered SQL statements
<i>sCU_i</i>	Total number of <code>CURRENT_TIME</code> and <code>CURRENT_DATE</code> variables in procedures, functions, and triggered SQL statements
<i>sS_i</i>	Number of data manipulation SQLs in procedures and triggered SQL statements (excluding cursor declarations: <code>OPEN</code> , <code>FETCH</code> , <code>CLOSE</code> , <code>UPDATE</code> , <code>DELETE</code> , <code>INSERT</code> , etc.)
<i>sP_i</i>	Number of routine control SQL statements in procedures, functions, and triggered SQL statements (<code>BEGIN</code> , <code>SET</code> , <code>IF</code> , <code>ELSEIF</code> , <code>WHILE</code> , etc.)
<i>sL_A</i>	Number of labels in procedures, functions, and triggered SQL statements
<i>sK_i</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>sW_i</i>	Number of conditional predicates in procedures, functions, and triggered SQL statements

Variable name	Explanation
<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>nFF</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
<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 Java procedure calls in routines
<i>nPJP</i>	Total number of Java procedure call arguments in routines
<i>nRJFC</i>	Number of Java function calls in routines
<i>nRJFP</i>	Total number of Java function call arguments in routines

Variable name	Explanation
<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

¹ When triggers are used, all of the trigger activation conditions of the triggers activated by execution of SQL statements must be counted.

² SQL statements that retrieve from foreign tables can be checked with the access path display utility (`pdvwopt` command).

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

$$\begin{aligned} &\text{Size of user LOB RDAREA (bytes)} \\ &= \{ \\ &\quad a \\ &\quad [\sum_{i=1}^a \uparrow Si \div 64000 \uparrow \times 96 + 7 + 3 \times (a - 1)]^1 \\ &\quad b \\ &\quad + [\sum_{j=1}^b \uparrow (Cj + 1024) \div 8192 \uparrow]^2 \\ &\quad \}^3 \times 8192^4 \end{aligned}$$

a: Number of HiRDB files that constitute the user LOB RDAREA

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.

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`)

Cj: Length of each BLOB data (bytes)

¹ Total number of pages in the directory pages part.

² Total number of pages in the data pages part.

³ Total number of pages in the user LOB RDAREA.

⁴ Page length of the user LOB RDAREA.

a: Length of registry contexts

b: 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)

$$= \left\{ \begin{array}{l} a \\ \left[\sum_{i=1}^a \left(S_i \div 64000 \uparrow \times 96 + 7 + 3 \times (a - 1) \right)^1 \right. \\ b \\ \left. + \left[\sum_{j=1}^b \left(C_j + 1024 \right) \div 8192 \uparrow \right]^2 \right. \\ \left. \right\}^3 \times 8192^4$$

a: Number of HiRDB files that constitute the registry LOB RDAREA

b: Number of registry key values that exceed 32000 bytes

S_i: Number of segments for the registry LOB RDAREA

C_j: Length of registry key values that exceed 32000 bytes

¹ Total number of pages in the directory pages part.

² Total number of pages in the data pages part.

³ Total number of pages in the registry RDAREA

⁴ Page length of the registry LOB RDAREA

16.9 Determining the size of the list RDAREA

Use the following formula to determine the size of the list RDAREA:

Formula

$$\begin{aligned} &\text{Size of list RDAREA (bytes)} \\ &= \{ \\ &\quad a \\ &\quad \Sigma (\uparrow Si \div f \uparrow + \uparrow Si \div g \uparrow) \\ &\quad \quad i=1 \\ &\quad + 6 \times (a + 1) + \uparrow (1024 \times n) \div (25 \times b) \uparrow + \uparrow 20480 \div b \uparrow + c \times e \\ &\quad \} \times b \end{aligned}$$

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: $\downarrow \{b - 20\} \div \{\uparrow e \div 32 \uparrow \times 8\} + 56 \downarrow$

g: $\uparrow (125 \times b) \div (16 \times f) \uparrow \times f$

n: Maximum number of lists that can be created in one list RDAREA*

Si: Number of HiRDB file segments that constitute the list RDAREA*

* This value is specified with the `create rdarea` statement of the database initialization utility (`pdinit`) or database structure modification utility (`pdmod`).

Chapter

17. Determining System File Size

This chapter explains how to determine the sizes of system files, including system log files, synchronization point dump files, and status files.

This chapter contains the following sections:

- 17.1 Determining the size of system log files
- 17.2 Determining the size of synchronization point dump files
- 17.3 Determining the size of status 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 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

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

$\text{Total size of all system log files (bytes)} = (\uparrow a \div c \uparrow \times 3)^* \times b$
--

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.
- 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) = $\Sigma \{b + e + \lceil b \div (a - 256) \rceil \times 256 + d\} + c$
 ■ For a recovery-unnecessary front-end server only, add:
 $+f$

Σ 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: 1328 + 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

17. Determining System File Size

- For HiRDB/Single Server
 $2 \times \text{pd_log_rec_leng operand value}$
- For HiRDB/Parallel Server
 - 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)	$(\uparrow 1328 \div \text{pd_log_rec_leng operand value} \uparrow + 3) \times \text{pd_log_rec_leng operand value}$
Not used	Used	0
Not used	Not used	$2 \times \text{pd_log_rec_leng operand value}$

- 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	$2 \times \text{pd_log_rec_leng operand value} + \uparrow 128 \times (\text{all BESs} + \text{all FESs}) \div \text{pd_log_rec_leng operand value} \uparrow \times \text{pd_log_rec_leng operand value}$
	server	
No	uap	$2 \times \text{pd_log_rec_leng operand value}$
	server	

e:

- For a HiRDB/Parallel Server
 For a front-end server: $56 \times \text{total number of back-end servers}$
 Other than a front-end server: 0
- For a HiRDB/Single Server: 0

f : $10 \times$ value of `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.

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

Condition	Amount of system log information (bytes)
Table not partitioned	$(794 \times b + 2331) \times a + (632 \times a \times d)^*$
Table partitioned	$(794 \times b + 2447 \times c + 3087) \times a + (632 \times a \times d)^*$

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

Condition	Amount of system log information (bytes)
Amount of system log information output by dictionary server	$(794 \times b + 1535 \times c + 3087) \times a$
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 are presented as follows.

(1) HiRDB/Single Server

Condition	Amount of system log information (bytes)
Indexes not partitioned	n $\sum_{i=1} \{ 600 \times b_i + W_i \times (X_i \times \uparrow (100 - f) \div 100 \uparrow + 132) + 272 \times W_i + 1940 \times \uparrow W_i \div V_i \uparrow \}$
Indexes partitioned	N $\sum_{i=1} \{ 611 \times a_i + 600 \times b_i + 5016 + a \sum_{j=1} \{ W_{ij} \times (X_{ij} \times \uparrow (100 - f) \div 100 \uparrow + 132) + 272 \times W_{ij} + 1940 \times \uparrow W_{ij} \div V_{ij} \uparrow \} \}$

- a*: Number of partitions for table for which index is defined
- f*: Value (%) of the PCTFREE operand specified during index definition
This is the percentage of unused space in a page.
- n*: Total number of indexes to be defined
- b*: Number of index component columns
- 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.*

X : Page length of user RDAREA used to store index (bytes)

(2) HiRDB/Parallel Server

Condition	Amount of system log information (bytes)
Amount of system log information output by dictionary server	$n \sum_{i=1} \{ 8 \times a_i + 600 \times b_i + 611 \times c_i + 5016 \}$
Amount of system log information output by back-end server	$n \sum_{i=1} \{ W_i \times (X_i \times \uparrow (100 - f) \div 100 \uparrow + 132) + 272 \times W_i + 1940 \times \uparrow W_i \div V_i \uparrow \}$

a_i : Number of partitions for table for which index is defined

b_i : Number of index component columns

c_i : Number of RDAREAs used to store index

n : Total number of indexes to be defined

f : Value (%) of the PCTFREE operand specified during index definition

This is the percentage of unused space in a page.

V_i : Segment size of user RDAREA used to store index (pages)

W_i : Number of index storage pages

For details, see 16.1.3 *Calculating the number of index storage pages.*

X_i : Page length of user RDAREA used to store index (bytes)

17.1.4 Amount of system log information output during table data updating

If you manipulate rows in a table, the system outputs the system log information listed in Table 17-1.

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	This log information is output when row data is manipulated in columns with the following data types: <ul style="list-style-type: none"> • VARCHAR¹ • NVARCHAR² • MVARCHAR¹ • Repetition columns • Abstract data type • BINARY type³
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 (INSERT, DELETE, or UPDATE statement), as shown in Table 17-2.
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 (INSERT statement)	Amount of addition log information
Key deletion (DELETE statement)	Amount of deletion log information
Key updating (UPDATE 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

Table 17-3 shows 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, 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

If a **BLOB** column is defined for the table, 9 bytes must be used as the row length shown in Table 17-3 for a **BLOB** column, and the amount of log information shown in Table 17-4 must be added.

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 (INSERT statement)	Not specified or partial specified	1176
	All specified	$2348 + p1 + 8340 \times p2 + (148 + lt) \times p3$
	Not specified	300
Data deletion (DELETE statement)	Not specified or partial specified	1052
	All specified	
	Not specified	468
Data updating (UPDATE statement)	Not specified or partial specified	1324
	All specified	$2496 + p1 + 8334 \times p2 + (148 + lt) \times p3$
	Not specified	312
Data concatenation operation (UPDATE statement) $Bb \leq 7168$	Not specified or partial specified	2344
	All specified	$8340 \times a + 1600 + d + 8340 \times p4 + (148 + lt2) \times p3$
	Not specified	428

Type of data manipulation (SQL statement)	Specification of recovery	Amount of log information output (bytes)
Data concatenation operation (UPDATE statement) $Bb > 7168$	Not specified or partial specified	2772
	All specified	$2772 + Ba + 8340 \times p4 + (148 + lt2) \times p3$
	Not specified	428

B_i : BLOB data length (bytes)

B_a : One of the following values:

- $B_b > 7168$: $8192 - \{(B_b - 7168) - \downarrow(B_b - 7168) \div 8192 \downarrow \} \times 8192$
- $B_b \leq 7168$: 0

B_b : BLOB data length before update (bytes)

B_c : BLOB added data length (bytes)

l_t : One of the following values:

- $B_i > 7168$: $B_i - 7168 - \downarrow(B_i - 7168) \div 8192 \downarrow \times 8192$
- $B_i \leq 7168$: 0

l_{t2} : One of the following values:

- $B_c + B_b > 7168$: $(B_c + B_b - B_a - 7168) - \downarrow(B_c + B_b - B_a - 7168) \div 8192 \downarrow \times 8192$
- $B_c + B_b \leq 7168$: 0

p_1 : One of the following values:

- $B_i > 7168$: 7168
- $B_i \leq 7168$: B_i

p_2 : One of the following values:

- $B_i > 7168$: $\downarrow(B_i - 7168) \div 8192 \downarrow$
- $B_i \leq 7168$: 0

p_3 : One of the following values:

- $l_t = 0$ or $l_{t2} = 0$: 0
- $l_t > 0$ or $l_{t2} > 0$: 1

p_4 : One of the following values:

- $Bc + Bb > 7168$: $\downarrow(Bc + Bb - Ba - 7168) \div 8192 \downarrow$
- $Bc + Bb \leq 7168$: 0

a: One of the following values:

- $Bb > 0$: 1
- $Bb = 0$: 0

d: One of the following values:

- $Bc + Bb \leq 7168$: $Bc + Bb$
- $Bc + Bb > 7168$: 7168

(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. Table 17-5 shows 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. Table 17-6 shows the formulas for obtaining the amount of log information for one 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) *	$(b + 160) + (SPN - 1) \times (b + 152)$ ■ If the <code>pd_rpl_hdepath</code> operand is specified, add: 160

* For BINARY type columns only.

b: Page length of an RDAREA

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 *di* as the length of data to be added.

$$SPN = SPN1 + SPN2 + SPN3 + SPN4$$

■ Other than BINARY columns

$$SPN1 =$$

$$f$$

$$\sum_{i=1}^f \uparrow \text{Value of branch } d_i \div (b - 61) \uparrow \times SF$$

$$SPN1 =$$

■ BINARY columns

■ Number of branch pages due to INSERT

$$SPN2 =$$

$$f$$

$$\{ \sum_{i=1}^f \downarrow \text{Value of branch } d_i \div (b - 59) \downarrow + 1 \} \times SF$$

■ Number of branch pages due to pdload or pdrorg

When value of branch $d_i > (b - 2853) \div 255$

$$SPN3 =$$

$$f$$

$$\uparrow \{ \sum_{i=1}^f (\text{branch } d_i + 11) \} \times (b - 48) \uparrow \times SF$$

When value of branch $d_i \leq (b - 2853) \div 255$

$$SPN4 = SF$$

f : Number of branch columns

SF : 1.3

Use a value greater than 1.3 in the following cases:

- A large number of columns with the abstract data type will be updated.
- A large number of repetition columns will be updated in such a manner that the elements' data lengths increase or the number of elements increases.
- A large number of columns with VARCHAR, NVARCHAR, MVARCHAR, or BINARY data type will be updated in such a manner that the elements' data lengths increase
- A large number of columns with BINARY data type will be updated in such a manner that the elements' data lengths increase
- A large number of columns with data suppressed will be updated in such a manner that their data lengths increase.
- A large number of columns other than the above will be updated in such a manner that the NULL values are changed to non-NULL values.

(3) Determining the amount of index log information

Table 17-7 shows the formulas for determining the amount of index log information that is output per index when manipulating a row.

Table 17-7: 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	$k_1 + 156$ or $(k_1 + 156) \times 2^*$	
	Adding the same key as for existing row	$d \leq 200$	$k_1 + 156$
		$d > 200$	$k_1 + 292$
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)

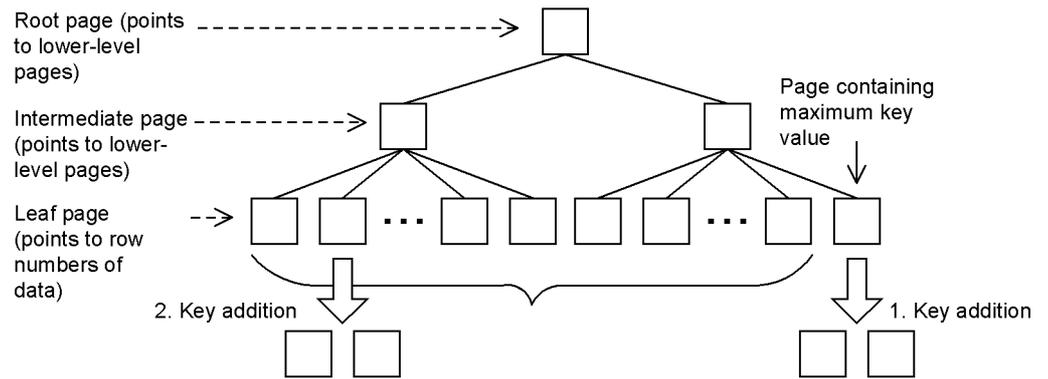
k_2 : Length of key to be deleted (bytes)

* 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

Figure 17-1 shows 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.

Table 17-8 shows the formulas for determining by split type the amount of index log information that is output at any one time.

Table 17-8: 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	$2 \times k_I + a + 8 \times (m + 1) \leq 31516$	$2 \times k_I + 472 + a + 8 \times (m + 1)$
			$2 \times k_I + a + 8 \times (m + 1) > 31516$	$2 \times k_I + 632 + a + 8 \times (m + 1)$
		There is not enough unused area to add the key in upper-level page	$2 \times k_I + a + 8 \times (m + 1) \leq 31516$	$n-1 \sum_{i=2} (288 + a) + 2 \times k_I + 472 + a + 8 \times (m + 1)$
			$2 \times k_I + a + 8 \times (m + 1) > 31516$	$n-1 \sum_{i=2} (288 + a) + 2 \times k_I + 628 + a + 8 \times (m + 1)$
	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	$n-1 \sum_{i=2} (288 + a) + 2 \times k_I + 472 + a + 8 \times (m + 1)$

Split type	Condition		Amount of log information output (bytes)	
		$d_l > 200$	There is enough unused area to add the key in lower-level page	$k_l + 472 + a$
			There is not enough unused area to add the key in lower-level page	$k_l + 462 + 2 \times a$
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	$2 \times k_l + a + 8 \times (m + 1) \leq 31516$	$2 \times k_l + 332 + a + 8 \times (m + 1)$
			$2 \times k_l + a + 8 \times (m + 1) > 31516$	$2 \times k_l + 492 + a + 8 \times (m + 1)$
		There is not enough unused area to add the key in upper-level page	$2 \times k_l + a + 8 \times (m + 1) \leq 31516$	$n-1 \sum_{i=2} (288 + a) + 2 \times k_l + 332 + a + 8 \times (m + 1)$
			$2 \times k_l + a + 8 \times (m + 1) > 31516$	$n-1 \sum_{i=2} (288 + a) + 2 \times k_l + 492 + a + 8 \times (m + 1)$

a : Page length of RDAREA storing the index (bytes)

d_l : Number of duplicated key values

k_l : Length of key value to be added (bytes)

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 a large amount of data is handled, the amount of log information shown in Table 17-9 must be added.

Table 17-9: 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)	1940

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. Table 17-10 shows the amount of event log information that is output when a single row is manipulated.

Table 17-10: 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 17-11.

- Database load utility (pdload command)
- Database reorganization utility (pdroorg command)
- Rebalancing facility (pdrbal command)

The system log size is calculated by adding the value shown in Table 17-12 to the value obtained in Table 17-11. 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 HiRDB/Single Server:
Obtain the total amount of system log information calculated for each RDAREA.
- For HiRDB/Parallel Server:
Obtain the total amount of system log information calculated for each RDAREA for each server.

Table 17-11: 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 or n specified in -i option)	$\sum_{i=1}^n \{ [132 + \uparrow (100 - f) \div 100 \uparrow \times Xi] \times Wi + 280 \times Wi + 1940 \times \uparrow Wi \div Vi \uparrow \}$ $+ 280 \times m + 1940 \times \uparrow m \div s \uparrow + a \times r$	$\sum_{i=1}^n \{ 280 \times Wi + 1940 \times \uparrow Wi \div Vi \uparrow \}$ $+ 280 \times m + 1940 \times \uparrow m \div s \uparrow + c \times r$

Number	Condition	Amount of system log information output (bytes)	
		a specified in -l option	p specified in -l option
2	Batch index creation not performed (s specified in -i option)	$n \sum_{i=1} \{ 280 \times W_i + 1940 \times \uparrow W_i \div V_i \uparrow + b \times r + e \times d \}$ $+ 280 \times m + 1940 \times \uparrow m \div s \uparrow + a \times r$	$n \sum_{i=1} \{ 280 \times W_i + 1940 \times \uparrow W_i \div V_i \uparrow + b \times r + e \times d \}$ $+ 280 \times m + 1940 \times \uparrow m \div s \uparrow + c \times r$
3	Index creation or regeneration performed (nothing loaded with pdload, or ixrc or ixor specified in pdrorg -k option)	Add the following amount to the system log: <ul style="list-style-type: none"> • 17.1.3 Amount of system log information output during index definition • Table 17-9 Amount of log information for page allocation and segment allocation • Table 17-14 Amount of system log information output during execution of the PURGE TABLE statement 	Add the following amount to the system log: <ul style="list-style-type: none"> • Table 17-9 Amount of log information for page allocation and segment allocation • Table 17-14 Amount of system log information output during execution of the PURGE TABLE statement

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 Tables 17-3 Formulas for determining the amount of base row log information per data item and 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 17-7 Amount of index log information per index.

c: Amount of log information output when one data item is added, as obtained from Table 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 Tables 17-8 Amount of index log information by split type and 17-9 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 17-11, 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 17-12: 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	Add the following amount of log information for (the number of LOB columns \times the number of rows): <ul style="list-style-type: none"> • See <i>Data addition (INSERT statement)</i> in Table 17-4 <i>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> $3544 \times (\text{LOB data length} \div 31744)$ (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 Table 17-14 <i>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:	Add the following log amount of information for each LOB RDAREA configuration file: $\Sigma \{17000 \times \uparrow (\text{the number of HiRDB file segments} \div 64000) \uparrow \times 95\}$

Conditions for addition	Values added to the amount of system log information
<ul style="list-style-type: none"> The table is reorganized without the -j option specified The LOB column does not specify <code>recovery no</code> and <code>n</code> is not specified for the -1 option 	<p>Add the following amount of log information for each item (the number of LOB columns or LOB attributes \times the number of rows).</p> <ul style="list-style-type: none"> ■ LOB column is <code>recovery all</code> specification and also -1 a specification $\uparrow 17600 \div sr \uparrow \times sr$ ■ Other than the above $\uparrow 3200 \div sr \uparrow \times sr$ <p><i>sr</i>: Record length of the system log file specified in the <code>pd_log_rec_leng</code> operand.</p>
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 Table 17-10 <i>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. Following is the formula for obtaining the amount of system log information that is output depending on the SQL manipulation.

Formula

<p>Amount of system log information (bytes) $= 568 \times (\text{CONNECTS count} + \text{"set session authorization" executions count})$</p>
--

17.1.7 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. Table 17-13 shows the formula for obtaining the amount of system log information that is output:

Table 17-13: 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	1956
Other than LOB RDAREA	$1372 + (144 + p) \times 2$

p: Page size of RDAREA subject to automatic extension

17.1.8 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. Table 17-14 shows the formulas for calculating the amount of system log information when the `PURGE TABLE` statement is executed. In the formulas, calculate the variables *A*, *B*, and *C* for each RDAREA configuration file.

Table 17-14: Amount of system log information output during execution of the `PURGE TABLE` statement

Type	Amount of system log information (bytes)
Tables	$1000 + 1100 \times \text{number of allocated segments}$
Rebalancing tables	$1000 + 1100 \times \text{number of allocated segments} + 400 \times \uparrow 1024 \div \text{number of partitioning RDAREAS} \uparrow$
Indexes*	$1000 + 1100 \times \text{number of allocated segments}$
LOB columns or LOB attributes*	$1000 + 17000 \times A + 160 \times B + 2 \times C$

A: $\uparrow (\text{number of segments used by HiRDB files} \div 64000) \uparrow \times 95$

B: $\uparrow (\text{number of segments used by HiRDB files} \div 64000) \uparrow$

C: $\uparrow (\text{number of segments used by HiRDB files} \div 64000) \uparrow \times 8150$

* For plug-ins, the initialization log for each plug-in is output. For details, see the documentation for each plug-in.

17.1.9 Amount of system log information output when the free page release utility (pdreclaim) is executed

Information is output to the system log when the free page release utility (`pdreclaim`) is used to release the free pages of tables or indexes. Table 17-15 shows the formulas for determining the amount of log output. If row partitioning is used with the tables or indexes, this calculation must be performed for each partitioned RDAREA. For a HiRDB/Single Server, determine the total amount of log information calculated from each row partitioned RDAREA. For a HiRDB/Parallel Server, determine the total amount of log information calculated from each row partitioned RDAREA for each server.

Table 17-15: Amount of system log information output when the free page release utility is executed

Type	Amount of system log information (bytes)
Tables	$140 \times n$ ■ Calculate when the -o option is not specified. $504 \times m$
Indexes	$\{(k + 14) \div (j - 68) \times 1408 + k + 12 \times h + 906\} \times n$ ■ When the -x option is specified, the amount of system log information is added when the free page release index is a unique index, a primary key, or a cluster key that specifies UNIQUE or PRIMARY. $\{m - n - ((k + 14) \times n) \div ((j - 68) \times 2)\} \times (j - 56) \times 0.7$
Segments	$2380 \times p$

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

17.1.10 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 (or that is output by the single server for a HiRDB/Single Server):

Formula

Amount of system log information (bytes) = $n \times \{1604 \times (A + B + C + D) \times (\uparrow E \uparrow + 1)\}$ $+ m \times \{872 \times (a + b + c + 1)\}$ $+ 11680 \times \uparrow \{(A + B + C + D) \times (\uparrow E \uparrow + 1)\} \times 30 \div 540 \uparrow$ $+ 332 \times \{(A + B + C + D) \times (\uparrow E \uparrow + 1)\} \times 30$ $+ 7760$

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 manual *HiRDB Version 8 System Operation Guide*.

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) = Number of records in synchronization point dump file ¹ × 4096 ²
--

¹ See (2) as follows; the obtained value is specified in 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		$\uparrow \{1520 \times (2 \times a + 7)\} \div 4096 \uparrow$ $+ \uparrow (96 + 112 \times c) \div 4096 \uparrow + 3$
HiRDB/Parallel Server	Front-end server	$\uparrow (320 \times b + 840) \times (2 \times a + 7) \div 4096 \uparrow + 4$
	Back-end server	$\uparrow \{1456 \times (2 \times d + 7) + 64 \times (2 \times d + 7) \times (b \times 4 + f + 2)\} \div 4096 \uparrow$ $+ \uparrow (96 + 112 \times c) \div 4096 \uparrow + 3$
	Dictionary server	$\uparrow \{1456 \times (2 \times e + 7) + 64 \times (2 \times e + 7) \times (b \times 4 + f + 2)\} \div 4096 \uparrow$ $+ \uparrow (96 + 112 \times c) \div 4096 \uparrow + 3$

a: Value of the `pd_max_users` operand

For multi-front-end servers, value of the `pd_max_users` operand + 1

b: Number of back-end servers

c: Value of the `pd_lck_until_disconnect_cnt` operand

d: Value of the `pd_max_bes_process` operand

e: Value of the `pd_max_dic_process` operand

f: Number of front-end servers

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

$$\text{Size of status file (bytes)} = a \times b$$

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

$$\begin{aligned} &\text{Number of records} \\ &= \lceil \lceil S \div (\text{record length} - 40) \rceil + \lceil S \div 100 \rceil + S + 2 \rceil \times 1.2 \end{aligned}$$

See (3) as follows for the value of S .

(3) Determining the value of S

(a) HiRDB/Single Server

Type	Formula for S
Unit	$\begin{aligned} &\lceil (2056 + 128 \times d) \div g \rceil + \lceil 2512 \div g \rceil + \lceil 40 \div g \rceil \\ &+ \lceil 308 \div g \rceil + \lceil 32 \div g \rceil + \lceil 4000 \div g \rceil \end{aligned}$

Type	Formula for S
Server	$\begin{aligned} & \uparrow 128 \div g \uparrow + \uparrow 3280^1 \div g \uparrow + \uparrow (592^2 + j \times b) \div g \uparrow \\ & + \uparrow 8192 \div g \uparrow + \uparrow (8192 - 128) \div g \uparrow \times \{((a + m) \div k) - 1\} \\ & + \uparrow [\uparrow \{(\uparrow c \div 127 \uparrow + 1) \times 2048\} \div 8192 \uparrow \times 8192] \div g \uparrow \\ & + \uparrow 2048 \div g \uparrow + \uparrow \{244 + 64 \times (2 \times h + 7)\} \div g \uparrow \\ & + \uparrow 96 \div g \uparrow + \uparrow \{48 + 68 \times (2 \times h + 7)\} \div g \uparrow \\ & + \uparrow \{8192 + \uparrow (e \times 98 \div 8192) \uparrow \times 8192 + \uparrow (f \times 80 \div 8192) \uparrow \times 8192\} \div g \uparrow \\ & + \uparrow (8 \times c + 4) \div g \uparrow + \uparrow 256 \div g \uparrow \\ & + w \\ & \blacksquare \text{ If the status file record length} < 4096, \text{ add:} \\ & + \downarrow \text{MAX}\{3400 \div r + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4096 \div s \downarrow, 2) \times n \\ & + \downarrow 5662310 \div r + 0.7 \downarrow \times \text{MAX}(\downarrow 4096 \div s \downarrow, 2) \times p \\ & + \downarrow \text{MAX}\{\text{MAX}(\downarrow 51200 \div t \downarrow, 1) \div 8 + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4096 \div s \downarrow, 2) \times 8 \times 2 \\ & \blacksquare \text{ If } 4096 \leq \text{status file record length} < 12288, \text{ add:} \\ & + \downarrow \text{MAX}\{(3400 \div \downarrow (s - 348) \div 20 \downarrow) + 0.7, 1\} \downarrow \times n \\ & + \downarrow (5662310 \div \downarrow (s - 348) \div 20 \downarrow) + 0.7 \downarrow \times p \\ & + \downarrow \text{MAX}\{(\text{MAX}(u, 1) \div 8) + 0.7, 1\} \downarrow \times 8 \times 2 \\ & \blacksquare \text{ If } 12288 \leq \text{status file record length, add:} \\ & + \downarrow \text{MAX}\{(3400 \div \downarrow (s - 876) \div 20 \downarrow) + 0.7, 1\} \downarrow \times n \\ & + \downarrow (5662310 \div \downarrow (s - 876) \div 20 \downarrow) + 0.7 \downarrow \times p \\ & + \downarrow \text{MAX}\{(\text{MAX}(u, 1) \div 8) + 0.7, 1\} \downarrow \times 8 \times 2 \\ & \blacksquare \text{ Add when the } \text{pd_log_auto_unload_path} \text{ operand is specified.} \\ & + \uparrow 20848 \div g \uparrow \\ & \blacksquare \text{ Add when the rapid system switchover facility and the standby-less system switchover facility} \\ & \text{are used.} \\ & + \uparrow (256 \times i + 4096) \div g \uparrow \\ & \blacksquare \text{ Add when } v \text{ is specified in the } \text{pd_dbbuff_modify} \text{ operand.} \\ & + v \end{aligned}$

a: Number of specified `pdlogadfg` operands

b: Number of specified `pdlogadfg -d spd` operands

c: Value of `pd_max_rdarea_no` operand

d: 90

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`

i: Value of `pd_max_rdarea_no` operand

j: 736

k: 11

m: 1

n: If the number of RDAREAs in the server ≤ 3400 , the value is 1.

If $3401 \leq$ number of RDAREAs in the server ≤ 6800 , the value is 2.

If $6801 \leq$ number of RDAREAs in the server, the value is 3.

p: If the number of RDAREAs in the server ≤ 10200 , the value is 0.

If $10201 \leq$ number of RDAREAs in the server ≤ 5672510 , the value is 1.

If $5672511 \leq$ number of RDAREAs in the server ≤ 11334820 , the value is 2.

If $11334821 \leq$ number of RDAREAs in the server, the value is 3.

r: $\lfloor (s - 348) \div 20 \rfloor + \lfloor g \div 20 \rfloor \times (\text{MAX}(\lfloor 4096 \div s \rfloor, 2) - 1)$

s: Status file record length

t: $\lfloor (s - 68) \div 20 \rfloor + \lfloor (s - 44) \div 20 \rfloor \times (\text{MAX}(\lfloor 4096 \div s \rfloor, 2) - 1)$

u: $\lfloor 51200 \div \{ \lfloor (s - 72) \div 20 \rfloor \} \rfloor$

v: In 32-bit mode: $\lceil (24 + 28 \times x + 32 + 112 \times y) \div g \rceil$

In 64-bit mode: $\lceil (32 + 32 \times x + 32 + 128 \times y) \div g \rceil$

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)

¹ For the 64-bit mode, use a value of 3456.

² For the 64-bit mode, use a value of 688.

(b) HiRDB/Parallel Server

Type	Formula for S
Unit	$\begin{aligned} & \uparrow [2056 + 128 \times \{14 \times (p+q) + (p+q+r) \\ & + (24 + s \times 2 + q \times 7 + r \times 3 + p \times 15 + 2) \\ & + (38 + (p+q+r) \times 4 + z + d) + 3\}] \div j \uparrow \\ & + \uparrow 944 \div j \uparrow + \uparrow 4816 \div j \uparrow + \uparrow 40 \div j \uparrow + \uparrow 372 \div j \uparrow \\ & + \uparrow 32 \div j \uparrow + \uparrow 4000 \div j \uparrow + \uparrow 43456 \div j \uparrow \\ & \blacksquare \text{ If the unit contains the system manager, add:} \\ & + \uparrow (32 + 16 \times \text{number of units}) \div j \uparrow \\ & + \uparrow (72 + 24 \times (\text{the total number of units in the system} - 1)) \div j \uparrow \\ & \blacksquare \text{ If the standby-less system switchover (effects distributed) facility is used, add:} \\ & + \uparrow \{8192 + \uparrow (f \times 98 \div 8192) \uparrow \times 8192 + \uparrow (g \times 80 \div 8192) \uparrow \times 8192\} \div j \uparrow \end{aligned}$
Front-end server	$\begin{aligned} & \uparrow 128 \div j \uparrow + \uparrow 3280^1 \div j \uparrow + \uparrow (592^2 + A \times b) \div j \uparrow \\ & + \uparrow 8192 \div j \uparrow + \uparrow (8192 - 128) \div j \uparrow \times \{\uparrow (a + C) \div B \uparrow - 1\} \\ & + \uparrow 8192 \div j \uparrow + \uparrow 2048 \div j \uparrow \\ & + \uparrow \{244 + 64 \times (2 \times e + 7)\} \div j \uparrow + \uparrow 96 \div j \uparrow \\ & \blacksquare \text{ Add when the } \text{pd_log_auto_unload_path} \text{ operand is specified.} \\ & + \uparrow 20848 \div j \uparrow \end{aligned}$

Type	Formula for S
Dictionary server	$\begin{aligned} & \uparrow 128 \div j \uparrow + \uparrow 3280^1 \div j \uparrow + \uparrow (592^2 + A \times b) \div j \uparrow \\ & + \uparrow 8192 \div j \uparrow + \uparrow 8192 \div j \uparrow \times \{ \uparrow (a + C) \div B \uparrow - 1 \} \\ & + \uparrow [\uparrow \{ (\uparrow c \div 127 \uparrow + 1) \times 2048 \} \div 8192 \uparrow \times 8192] \div j \uparrow \\ & + \uparrow 2048 \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 \{ 8192 + \uparrow (f \times 98 \div 8192) \uparrow \times 8192 + \uparrow (g \times 80 \div 8192) \uparrow \times 8192 \} \div j \uparrow \\ & + \uparrow (8 \times c + 4) \div j \uparrow \\ & + H \\ & \blacksquare \text{ If the status file record length } < 4096, \text{ add:} \\ & + \downarrow \text{MAX}\{3400 \div D + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4096 \div n \downarrow, 2) \\ & + \downarrow \text{MAX}\{\text{MAX}(\downarrow 51200 \div E \downarrow, 1) \div 8 + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4096 \div n \downarrow, 2) \times 8 \times 2 \\ & \blacksquare \text{ If } 4096 \leq \text{ status file record length } < 12288, \text{ add:} \\ & + \downarrow \text{MAX}\{(3400 \div \downarrow (n - 348) \div 20 \downarrow) + 0.7, 1\} \downarrow \\ & + \downarrow \text{MAX}\{(\text{MAX}(F, 1) \div 8) + 0.7, 1\} \downarrow \times 8 \times 2 \\ & \blacksquare \text{ If } 12288 \leq \text{ status file record length, add:} \\ & + \downarrow \text{MAX}\{(3400 \div \downarrow (n - 876) \div 20 \downarrow) + 0.7, 1\} \downarrow \\ & + \downarrow \text{MAX}\{(\text{MAX}(F, 1) \div 8) + 0.7, 1\} \downarrow \times 8 \times 2 \\ & \blacksquare \text{ Add when the system switchover facility is used.} \\ & + \uparrow (256 \times c + 4096) \div j \uparrow \\ & \blacksquare \text{ Add when the } \text{pd_log_auto_unload_path} \text{ operand is specified.} \\ & + \uparrow 20848 \div j \uparrow \\ & \blacksquare \text{ Add when HiRDB External Data Access is used.} \\ & + \uparrow \{ 128 \times (v + w + 1) \} \div j \uparrow \\ & \blacksquare \text{ Add when } \bar{Y} \text{ is specified in the } \text{pd_dbbuff_modify} \text{ operand.} \\ & + G \end{aligned}$

Type	Formula for S
Back-end server	$\begin{aligned} & \uparrow 128 \div j \uparrow + \uparrow 3280^1 \div j \uparrow + \uparrow (592^2 + A \times b) \div j \uparrow \\ & + \uparrow 8192 \div j \uparrow + \uparrow 8192 \div j \uparrow \times \{ \uparrow (a + C) \div B \uparrow - 1 \} \\ & + \uparrow [\uparrow \{ (\uparrow c \div 127 \uparrow + 1) \times 2048 \} \div 8192 \uparrow \times 8192] \div j \uparrow \\ & + \uparrow 2048 \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 (8 \times c + 4) \div j \uparrow + \uparrow 256 \div j \uparrow \\ & + \uparrow \{ 8192 + \uparrow (f \times 98 \div 8192) \uparrow \times 8192 + \uparrow (g \times 80 \div 8192) \uparrow \times 8192 \} \div j \uparrow^4 \\ & + H \\ & \blacksquare \text{ If the status file record length} < 4096, \text{ add:} \\ & + \downarrow \text{MAX}\{3400 \div D + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4096 \div n \downarrow, 2) \times k \\ & + \downarrow 5662310 \div D + 0.7 \downarrow \times \text{MAX}(\downarrow 4096 \div n \downarrow, 2) \times m \\ & + \downarrow \text{MAX}\{\text{MAX}(\downarrow 51200 \div E \downarrow, 1) \div 8 + 0.7, 1\} \downarrow \times \text{MAX}(\downarrow 4096 \div n \downarrow, 2) \times 8 \\ & \times 2 \\ & \blacksquare \text{ If } 4096 \leq \text{status file record length} < 12288, \text{ add:} \\ & + \downarrow \text{MAX}\{(3400 \div \downarrow (n - 348) \div 20 \downarrow) + 0.7, 1\} \downarrow \times k \\ & + \downarrow (5662310 \div \downarrow (n - 348) \div 20 \downarrow) + 0.7 \downarrow \times m \\ & + \downarrow \text{MAX}\{\text{MAX}(F, 1) \div 8 + 0.7, 1\} \downarrow \times 8 \times 2 \\ & \blacksquare \text{ If } 12288 \leq \text{status file record length}, \text{ add:} \\ & + \downarrow \text{MAX}\{(3400 \div \downarrow (n - 876) \div 20 \downarrow) + 0.7, 1\} \downarrow \times k \\ & + \downarrow (5662310 \div \downarrow (n - 876) \div 20 \downarrow) + 0.7 \downarrow \times m \\ & + \downarrow \text{MAX}\{\text{MAX}(F, 1) \div 8 + 0.7, 1\} \downarrow \times 8 \times 2 \\ & \blacksquare \text{ Add when the system switchover facility is used.} \\ & + \uparrow (256 \times y + 4096) \div j \uparrow \\ & \blacksquare \text{ Add when the } \text{pd_log_auto_unload_path} \text{ operand is specified.} \\ & + \uparrow 10608 \div j \uparrow \\ & \blacksquare \text{ Add when HiRDB External Data Access is used.} \\ & + \uparrow \{128 \times (t + u + 1)\} \div j \uparrow \\ & \blacksquare \text{ Add when } Y \text{ is specified in the } \text{pd_dbbuff_modify} \text{ operand.} \\ & + G \end{aligned}$

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 on a single server machine, the value is 2.

e: value of `pd_max_users` operand + 1

However, if a HiRDB/Parallel Server is used with a single server machine, value of `pd_max_users` 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

i: Value of `pd_max_bes_process` operand

j: Record length of status file minus 40

k: If the number of RDAREAs in the server ≤ 3400 , the value is 1.

If $3401 \leq$ number of RDAREAs in the server ≤ 6800 , the value is 2.

If $6801 \leq$ number of RDAREAs in the server, the value is 3.

m: If the number of RDAREAs in the server ≤ 10200 , the value is 0.

If $10201 \leq$ number of RDAREAs in the server ≤ 5672510 , the value is 1.

If $5672511 \leq$ number of RDAREAs in the server ≤ 11334820 , the value is 2.

If $11334821 \leq$ 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

t: Number of foreign servers defined to link to this back-end server

u: Number of user mappings defined for foreign servers to link to this back-end server

v: Total number of foreign server definitions

w: Total number of user mapping definitions

y: Value of `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: $\lfloor (n - 348) \div 20 \rfloor + \lfloor j \div 20 \rfloor \times (\text{MAX}(\lfloor 4096 \div n \rfloor, 2) - 1)$

E: $\lfloor (n - 68) \div 20 \rfloor + \lfloor (n - 44) \div 20 \rfloor \times (\text{MAX}(\lfloor 4096 \div n \rfloor, 2) - 1)$

F: $\lfloor 51200 \div \{ \lfloor (n - 72) \div 20 \rfloor \} \rfloor$

G: In 32-bit mode: $\lceil (24 + 28 \times I + 32 + 112 \times J) \div g \rceil$

In 64-bit mode: $\lceil (32 + 32 \times I + 32 + 128 \times J) \div g \rceil$

H: In 32-bit mode: $\lceil ((\lfloor (\lceil c \div 8 \rceil + 3) \div 4 \rfloor) \times 4) \times K \rceil \div g \lceil$

In 64-bit mode: $\lceil ((\lfloor (\lceil c \div 8 \rceil + 7) \div 8 \rfloor) \times 8) \times K \rceil \div g \lceil$

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)

¹ For the 64-bit mode, use a value of 3456.

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.

Chapter

18. Determining Work Table File Size

This chapter explains how to determine the size of a work table file.

This chapter contains the following sections:

- 18.1 Overview of determining the size of a work table file
- 18.2 Determining the size of a HiRDB file system area (pdfmkfs -n command)
- 18.3 Determining the maximum number of files (pdfmkfs -l command)
- 18.4 Determining the maximum number of extensions (pdfmkfs -e command)

18.1 Overview of determining the size of a work table file

This section describes the estimation of the size of a work table file that is used to temporarily store information needed to execute SQL statements.

When you perform any of the following operations, the system creates a work table file to store temporary information:

- Execution of SQL statements
- Batch index creation
- Index re-creation
- Index reorganization
- Execution of the rebalancing utility

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; these options are listed in Table 18-1.

Table 18-1: Options for which values need to be specified

Option	Description
-n	Size of HiRDB file system area in which work table files are to be created
-l	Maximum number of HiRDB files (work table files) that can be created in the HiRDB file system area
-e	Maximum number of secondary allocations for the HiRDB file system area

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 (`pdreorg`), and rebalancing utility (`pdrebal`). 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, specify `Y` in the `pd_large_file_use` operand before creating the HiRDB file system area. 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**:

$$\text{Size of a work table file used by an SQL statement (bytes)} = \text{MAX}(a, b) \times c$$

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

However, when multiple front-end servers are being used, the back-end servers are value of the `pd_max_bes_process` 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

$$\text{Size of work table file to be used by one SQL statement (bytes)} = a \times b + c \times d$$

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

$$\begin{aligned} &\text{Size of a column information work table (bytes)} \\ &= \uparrow a \div \text{MIN}\{\downarrow (b - 48) \div c \downarrow, 255\} \uparrow \times b \times 2 \end{aligned}$$

a: Number of rows in the column information work table (see Table 18-2)

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

$$\text{Page length for a work table}^{\#} = \text{MAX} \{ \uparrow (\text{row length for work table} + 48) \div 2048 \uparrow \times 2048, 4096 \}$$

#: The page length of a work table must be no greater than 32,768 bytes.

Formula 2

$$\text{Row length for a work table}^* = \sum_{i=1}^n A_i + 2 \times n + 6$$

A_i:

Data length for each column in work table (see Table 18-3 for the calculation procedure)

n:

Number of columns in work table (see Table 18-3 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.

SQL statement	Number of rows in column information work table
CREATE INDEX statement	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	Number of columns specified in selection expression + number of columns specified in GROUP BY clause + number of columns specified in ORDER BY clause + number of columns specified in HAVING clause + 1 if the FOR UPDATE clause is specified If ROW is specified in the selection expression, specify the total number of rows in the table.	Data length for each column However, in the case of 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 <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
CREATE INDEX statement	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)*

(b) Obtaining the maximum number of column information work tables

Table 18-4 shows the procedure for obtaining 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
SELECT statement	<p>When none of 1-10 as follows is applicable: 0 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"> 1. When multiple tables are joined for retrieval Number of additional work tables (HiRDB/Single Server) = number of joined tables + 1 Number of additional work tables (HiRDB/Parallel Server) = number of joined tables × 2 If the join key column has an index and there is a limitation condition, the number of work tables is 0. 2. When specifying the ORDER BY clause Number of additional work tables = 2 For a non-partitioned table, if the column with the ORDER BY clause specified has an index, the number of work tables is 0. 3. When specifying a value expression containing a set function in the selection expression without specifying the GROUP BY clause* Number of additional work tables = 1 4. When specifying the GROUP BY clause Number of additional work tables = number of GROUP BY clauses specified × 2 5. When specifying the DISTINCT clause Number of additional work tables = number of DISTINCT clauses specified × 2 6. When specifying the UNION [ALL] or EXCEPT [ALL] clause Number of additional work tables (HiRDB/Single Server) = number of UNION [ALL] or EXCEPT [ALL] clauses specified + 2 Number of additional work tables (HiRDB/Parallel Server) = (number of UNION [ALL] or EXCEPT [ALL] clauses specified + 1) × 2 7. 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* Number of additional work tables = 2 8. When specifying the FOR READ ONLY clause Number of additional work tables = 1 9. 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) <ol style="list-style-type: none"> 1. 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
CREATE INDEX statement	2

* Applicable only to a HiRDB/Parallel Server.

(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) $= \uparrow a \div 184^* \uparrow \times 4096 \times 2$
--

* 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	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: <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

Table 18-5 shows the procedure for obtaining 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	Number of indexes to be used during search + 1 in either of the following cases: <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. Facility for batch acquisition from functions provided by plug-ins 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. In all these cases, the value is the number of indexes used during a search + 1.

SQL statement	Maximum number of location information work tables
UPDATE statement DELETE statement	If the search condition contains a column with an index defined, use the number of indexes used during search processing + 1.

* This is applicable to a HiRDB/Single Server 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:

$$B_i = \uparrow \text{ number of rows for which predicate } i \text{ is true in the base table for the list} \div 504 \uparrow \times 4096 \times 1.5 \text{ (bytes)}$$

* 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 $= \uparrow c \div \{ \downarrow \uparrow a \times (100 - b) \times 0.01 \uparrow \div (d + 22) \downarrow \} \uparrow$

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 $\uparrow \text{key length} \div 4 \uparrow \times 4$.

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 $= \uparrow c \div \{ \downarrow \uparrow a \times (100 - b) \times 0.01 \uparrow \div (d + 14) \downarrow \} \uparrow$

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 $\lceil \text{key length} \div 4 \rceil \times 4$.

For multiple indexes, add the key lengths of all component columns on the basis of Table 16-5 *List of index key lengths*.

(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 $\lceil \text{key length} \div 4 \rceil \times 4$.

For multiple indexes, add the key lengths of all component columns on the basis of Table 16-5 *List of index key lengths*.

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

$$\text{Page length of work table} = \text{MIN}\{ \lceil (a + 4) \div 2048 \rceil \times 2048, 2048 \}$$

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 $\lceil \text{key length} \div 4 \rceil \times 4$.

For multiple indexes, add the key lengths of all component columns on the basis of Table 16-5 *List of index key lengths*.

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

$$\text{Maximum number of files} = \text{MAX}(a, b) \times c + 20 + 2^*$$

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

However, when multiple front-end servers are being used, the back-end servers are value of the `pd_max_bes_process` 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

$$\text{Number of work table files to be used by one SQL statement} = \text{maximum number of column information work tables} + \text{maximum number of location information work tables}$$

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 × 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 4096, specify a value of 4096 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, 60000)
--

Note

For details about how to obtain the maximum number of files, see *18.3 Determining the maximum number of files (pdfmkfs -l command)*.

Chapter

19. Storage Requirements for Utility Execution

This chapter explains how to determine the file sizes and storage requirements for execution of utilities.

This chapter contains the following sections:

- 19.1 Determining the file sizes required for utility execution
- 19.2 Determining the memory size required for utility execution

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	<p>B-tree index: $(d + y) \times (b + e) + 512$</p> <p>Plug-in index: $(12 + q) \times p + 1024$</p> <p>These formulas are for the size of one index. If there are multiple indexes, determine the size of each index.</p>
Error information file	$k \times f + s \times 200$
Temporary file for creating error information file	<p>If the following condition is true, the size obtained from the formula <i>number of key duplicate errors</i> $\times 8$ + <i>number of errors detected by plug-in function</i> $\times 200$ is also required for the temporary file storage area (location specified in the TEMP environment variable).</p> <p>Condition: For each server that contains table storage RDAREAs:</p> <ul style="list-style-type: none"> For a HiRDB/Parallel Server, the server that contains the input files is different from the server containing the table storage RDAREAs.
LOB input file	b $\sum_{i=1} (LOB\ data\ length + 4)i$
LOB middle file	B $\sum_{i=1} \{$ c $\sum_{j=1} (LOB\ file\ name\ length - ij + 36) + 24\} + 1024 + c \times 84$ $j = 1$
Error data file	$\text{MIN}(f, g) \times h$
Process results file	$1500 + \text{number of servers storing table} \times 500$

File type	Formula (bytes)
Work file for sorting	<p>Condition 1: Size of index information file + $4 \times (b + e)$</p> <p>Condition 2: {Size of index information file + $4 \times (b + e)$} $\times 2$</p> <p>● Condition 1: Work buffer size $\geq \sqrt{b + e} \times (2 \times d + y + 22)$</p> <p>● Condition 1: Work buffer size $< \sqrt{b + e} \times (2 \times d + y + 22)$</p> <ul style="list-style-type: none"> • Condition 1 When the work buffer size specified in the <code>sort</code> statement $\geq E$ • Condition 2 When the work buffer size specified in the <code>sort</code> statement $< E$ <p><i>E</i>: Buffer size The buffer size obtained according to <i>buffer-size-for-sorting</i> in <i>Database Load Utility (pload)</i> of the manual <i>HiRDB Version 8 Command Reference</i>.</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 `pdrowg`, 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

- For abstract data type
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
- 27 for the SGMLTEXT type
- 2 for FREEWORD and GEOMETRY types

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.

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 × number of elements).

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)	$n \sum_{i=1} (L_i + P_i) + 1200 + A + B + c \times 96 + D + I + F$

File type	Formula (bytes)
Unload data file ¹ (-w option specified)	<p>DAT or extended DAT format: c $\sum_{i=1} (\text{maximum length of converted character string in column } i^2 + 1) \times n$</p> <p>FIX table in binary format: c $\sum_{i=1} (\text{column data length } i^3) \times n$</p> <p>Non-FIX table in binary format: $\{$ c $\sum_{i=1} (\text{column data length } i^3 + G) + 4 \times (c + 1)$ $\} \times n$</p> <p>Fixed-length character format: c $\sum_{i=1} (\text{maximum length of converted character string for column } i^4 + crlf) \times n$</p>
Unload data file ¹ (-j option specified or during reorganization in units of schemas) ⁵	n $\sum_{i=1} (L_i + P_i) +$ n $\sum_{i=1} \{$ m $\sum_{j=1} (O_{ij} + 44)$ $\} + 1200 + A + B + c \times 96 + D + I + F$
LOB data unload file ¹	n $\sum_{i=1} \{$ m $\sum_{j=1} (O_{ij} + 44)$ $\} + 1200 + A + B + c \times 96 + D + I + F$
Index information file	<p>B-tree index: $(K + p) \times n + 512$</p> <p>Plug-in index: $(12 + X) \times n + 1024$</p> <p>These formulas are for the size of one index. If there are multiple indexes, determine the size of each index.</p>

File type	Formula (bytes)
Process results file	1700 + number of servers storing table × 500 + number of tables in schema × 1000 + total number of storage RDAREAs in schema × 100
Work file for sorting	<p>Condition 1: Size of index information file + 4 × <i>n</i></p> <p>Condition 2: {Size of index information file + 4 × <i>n</i>} × 2</p> <ul style="list-style-type: none"> • Condition 1 When the work buffer size specified in the <code>sort</code> statement ≥ <i>E</i> • Condition 2 When the work buffer size specified in the <code>sort</code> statement < <i>E</i> <p><i>E</i>: Buffer size The buffer size obtained according to <i>buffer-size-for-sorting</i> for Database Reorganization Utility (<i>pdorg</i>) in the manual <i>HiRDB Version 8 Command Reference</i>.</p>

A: For key range partitioning: 48 + number of partitioning conditions × 284

For hash partitioning: 40 + *a* × 60

For matrix partitioning (combination of key range partitioning of the boundary value specification and hash partitioning): 48 + (*number of partitioning conditions* × 284) + (40 + *a* × 60)

B: *n* × 36 (for FIX table) or (44 + *c* × 4) × *n* (for non-FIX table)

D: 16 + (*number of LOB columns* × *a* × 80)

Add the value of *D* only if there are LOB columns.

F: Use the following value:

$d \sum_{i=1} \{(\text{number of abstract-data-type attributes provided by plug-in in column } i \times 84) + (\text{number of abstract-data-type LOB attributes provided by plug-in in column } i \times a \times 72)\}$ <p>+ 64 +</p> $d \sum_{i=1} (84 + \text{number of reverse generation functions } i \times 60)$
--

G: Number of attributes for which the return value of the reverse generation function on column *I* is BLOB × 4

I: 136 + number of index partitions × 60

Add this value when including the index.

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

Add the data length of each column to obtain the row length. If the data type of a column is BLOB, use 16 as the column length for calculating. If the data type of a column is the abstract data type provided by a plug-in, use 2 as the column length for calculating.

P_i: Data length of the abstract data type provided by a plug-in

O_{ij}: LOB data length

X: Value is as follows

- For abstract data type
 - 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
- 27 for the SGMLTEXT type
- 2 for FREEWORD and GEOMETRY 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

m: Number of LOB columns

n: Number of rows (for a repetition column, number of rows × 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 × number of elements).

¹ If the file is larger than 2 GB, take the following action:

- Create multiple files, each of which is no larger than 2 GB.

² Table 19-1 lists the 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	31
	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
	MVARCHAR	Actual length + 2 ²

Data type		Maximum length of converted character string (bytes)
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 (`pdreorg`) in the manual *HiRDB Version 8 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 Table 19-2 shows the maximum lengths of converted character strings for columns of the 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	
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 SECOND	8	
Time stamp data	TIMESTAMP	Decimal part 0:19 2:22 4:24 6:26	
Large object data	BLOB	0	
Binary data	BINARY	0	

Data type		Maximum lengths of converted character string (bytes)
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.

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)
Work file	Statistical information about system activity	$4096 \times \text{collection count}^* \times 2$
	Statistical information about system activity per server	$4096 \times \text{collection count}^* \times \text{number of servers} \times 2$
	Statistical information about UAPs	$800 \times \text{number of UAPs to be executed or number of transactions to be executed} \times 2$
	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	$8192 \times \text{number of SELECT statements issued by OPEN or EXECUTE (including INSERT SELECT)}$
	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	$256 \times \text{number of SQLs to be executed} \times \text{number of servers}$
	Statistical information about the history of SQL statements	$(1024 + \text{average SQL length}) \times \text{number of SQLs to be executed}$
	Statistical information about CONNECT/DISCONNECT	$256 \times \text{number of CONNECTS and DISCONNECTS}$
	Statistical information about global buffer	$512 \times \text{number of synchronization points} \times 2$

19. Storage Requirements for Utility Execution

File type		Formula (bytes)
	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 foreign server activity	$512 \times \text{number of transactions that accessed foreign servers}$
	Statistical information about foreign server utilization status	$512 \times \text{number of SQL statements executed on foreign servers}$
Work file for sorting	Work area for sorting the above work files for analysis	Maximum size of the above work files for analysis
DAT-format file	Statistical information about system activity	$3262 \times \text{collection count}^*$
	Statistical information about system activity per server	$3262 \times \text{collection count}^* \times \text{number of servers}$
	Statistical information about UAPs	$1410 \times \text{number of UAPs to be executed or number of transactions to be executed}$
	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	$(240 + \text{average SQL length}) \times \text{number of SQLs to be executed}$

	File type	Formula (bytes)
	Statistical information about CONNECT/DISCONNECT	$278 \times \text{number of CONNECTS and DISCONNECTS}$
	Statistical information about global buffer	$567 \times \text{number of synchronization points}$
	Statistical information about database manipulation for HiRDB files	$356 \times \text{number of synchronization points}$
	Statistical information about deferred write processing	$300 \times \text{number of deferred write operations}$
	Statistical information about foreign server activity	$374 \times \text{number of transactions that accessed foreign servers}$
	Statistical information about foreign server utilization status	$389 \times \text{number of SQL statements executed on foreign servers}$

* Collection count = $\downarrow(\text{pdstend command input time} - \text{pdstbegin command input time}) \div \text{interval specified with the -m option} \downarrow$

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 in units of RDAREAs	60
	Logical analysis in units of RDAREAs	$1 + 3.6 \times \{(\text{number of tables and indexes in the RDAREA}) + \text{number of LOB RDAREAs}\}$
	Accumulating condition analysis result or reorganization time prediction*	$1 + 3.6 \times \{(\text{number of tables and indexes in the RDAREA}) + \text{number of LOB RDAREAs}\}$
	Status analysis in units of tables	$1 + 3.6 \times (\text{number of storage RDAREAs})$
	Status analysis in units of indexes	$1 + 3.6 \times (\text{number of storage RDAREAs})$
	Cluster key status analysis	$1 + 3.6 \times (\text{number of storage RDAREAs})$

File type		Formula (bytes)
Work file for sorting	Work area for sorting the above work files	Value obtained from the above formula × 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.

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* Full backup file*	a $\sum_{i=1} \{28 \times c_i + (d_i + 28) \times (e_i + q_i)\}$ $+ 88 \times a + 220 \times b$
Differential backup file*	$w \times \text{size of full backup file}$
Differential backup management file	$(l + j + m) \times 32768$
Log point information file	1024

* If the backup file is larger than 2 GB, take the following action:

- 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 : $\uparrow(512 + 128 \times a) \div 32700 \uparrow$ k : Number of consecutive differential backup operations

m : $\uparrow \{ \uparrow(256 + 128 \times a + g + 8 \times h) \div 256 \uparrow \times k \} \div 100 \uparrow$

q_i : Number of directory pages in the RDAREAs being backed up

- User RDAREAs

$6 \times (t_i + 1) + 2 \times \uparrow(20480 \div d_i) \uparrow + \{ \uparrow(s_i \div u_i) \uparrow + \uparrow(s_i \div v_i) \uparrow + 2 \times t_i \}$

- User LOB RDAREA

$7 + 3 \times (t_i - 1) + \{ \uparrow(s_i \div 64000) \uparrow + t_i \} \times 96$

r_i : Segment size of RDAREAs being backed up

s_i : 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):

$$w = \left\{ \frac{a \sum_{i=1}^a Xi \div a \sum_{i=1}^a e_i}{\times 1.2} \right\}$$

Xi : 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 Xi . Calculate `PCTFREE` as 0.

- DELETE

Increase the value of Xi based on the following conditions:

Condition			Value added to X_i
Tables	Row length $< d_i$	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 $> d_i$		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 $< d_i$	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)
	Update column length $> d_i$		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.

$$\beta = \text{number of used pages of regenerated tables or indexes} + \text{number of used segments of regenerated tables or indexes} \div ui + \text{number of used segments of regenerated tables or indexes} \div vi$$

Calculate β only for regenerated tables and indexes, and add the value to Xi .

- PURGE

Calculate γ as the tables or indexes on which PURGE was performed that are stored in RDAREAs being backed up.

$$\gamma = \text{number of used segments of purged tables or indexes} \div ui + \text{number of used segments of purged tables or indexes} \div vi$$

Calculate γ only for regenerated tables and indexes, and add the value to Xi .

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	$0.8 + 0.4 \times (a + b) + (0.8 + 0.1 \times c) \times d + 0.1 \times f$
	View table	$0.5 + 0.4 \times a + e$
	Procedure	$1.0 + 0.1 \times g + h$

a: Number of columns in table to be exported

b: Number of partitioning conditions for table to be exported

c: Number of index storage RDAREAs defined for table to be exported

d: Number of indexes defined for table to be exported

e: Length of the source during view table definition (KB)

f: If a LOB column is defined in the table to be exported, this is the number of user LOB RDAREAs storing the LOB data.

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	$162 + 405 \times a + 567 \times b$
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: $370 + 235 \times c + 387 \times f$

a: Number of specified indexes

b: Number of specified columns

c: Number of indexes defined in table

d: Number of intervals

e: Number of tables

f: Number of columns in table

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</p> $240 + 2 \times \sum_{i=1}^a \left\{ \begin{array}{l} 60 + \text{length of SQL} \\ + \sum_{j=1}^{b_i} \left\{ \begin{array}{l} 100 + \sum_{k=1}^{c_{ij}} (160 + e_{ijk} \times 20 + f_{ijk} \times 70) \\ + d_{ij} \times 180 \end{array} \right\} \end{array} \right\}$ <p>● HiRDB/Parallel Server</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: $(12 + Y) \times N + 1024$</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 for sorting*	<p>Condition 1: Size of index information file + $4 \times N$</p> <p>Condition 2: $\{\text{Size of index information file} + 4 \times N\} \times 2$</p> <ul style="list-style-type: none"> • Condition 1 When the work buffer size specified in the sort statement $\geq E$ • Condition 2 When the work buffer size specified in the sort statement $< E$ <p><i>E</i>: Buffer size The buffer size obtained according to <i>buffer-size-for-sorting</i> for <i>Rebalancing Utility (pdrbal)</i> in the manual <i>HiRDB Version 8 Command Reference</i>.</p>
Execution results output file	$1000 + \text{number of table storage RDAREAs} \times 200$

d: If all key component columns are fixed-length, the value is 10; if they include a variable-length column, the value is 12.

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)

Y: Value as follows

- For abstract data type
 - 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
- 27 for the SGMLTEXT type
- 2 for FREEWORD and GEOMETRY types

* 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	$ \begin{aligned} & \text{-k set/release} \\ & 560 + (REF + 1) \times 70 + (CHK + 1) \times 70 \\ & \text{-k check} \\ & 700 + (REF + 1) \times 70 + (CHK + 1) \times 70 \\ & REF \\ & + \sum_{n=1} (490 + (RC_n \times 70)) \\ & GEN ROW \\ & + \sum_{m=1} (\sum_{l=1} (RC_{nml} \times 70)) \\ & CHK \\ & + \sum_{n=1} (490 + (CC_n \times 70)) \\ & GEN ROW \\ & + \sum_{m=1} (\sum_{l=1} (CC_{nml} \times 70)) \end{aligned} $

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

ROW: Upper limit of the number of outputs of the key value that caused a constraint error (value specified by the -w option)

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

Condition	Formula for determining memory size (KB)
32-bit mode	$\uparrow \{$ $61440 \times (140 \times a + 20 \times b + c) \div 61432 + 6004 \times d \div 500$ $+ 36008 \times a \div 1000 + 468 \times e + 404399$ $\} \uparrow \div 1024 + 267$
64-bit mode	$\uparrow \{$ $61448 \times (144 \times a + 24 \times b + c) \div 61436 + 8004 \times d \div 500$ $+ 36016 \times a \div 1000 + 468 \times e + 406399$ $\} \uparrow \div 1024 + 267$

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

Condition	Formula for determining memory size (KB)
32-bit mode	$\uparrow \{$ $\{61448 \times (144 \times a + 24 \times b + c)\} \div 61436 + (8004 \times d) \div 500$ $+ (36016 \times a) \div 1000 + 468 \times e + 406399 + 348 \times f + 344 \times g$ $\} \uparrow \div 1024 + 268$
	$\uparrow (4 \times b + 246255) \div 1024 \uparrow + 268$
	10

Condition		Formula for determining memory size (KB)
64-bit mode	DS	$\uparrow \{ \{ 61448 \times (144 \times a + 24 \times b + c) \} \div 61436 + (8004 \times d) \div 500 + (36016 \times a) \div 1000 + 468 \times e + 406399 + 348 \times f + 344 \times g \} \uparrow \div 1024 + 268$
	BES	$\uparrow (4 \times b + 245744) \div 1024 \uparrow + 268$
	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 $(144 \times a + 24 \times b + c) \div 7780$ 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	1956 (or 1957 in the 64-bit mode)
HiRDB/Parallel Server	1956 (or 1957 in the 64-bit mode)

19.2.3 Memory size required for the execution of the database load utility (pdload)

The following tables show the formulas for determining the memory size required for the execution of the database load utility (pdload). For details about the variables, see (3) as follows.

(1) HiRDB/Single Server

Condition	Formula for determining memory size (KB)
32-bit mode	$4352 + \uparrow \{ (\alpha + \gamma) \div 1024 \} \uparrow + 1482 + \uparrow \{ (\beta + \delta) \div 1024 \} \uparrow$

Condition	Formula for determining memory size (KB)
64-bit mode	$18050 + \lceil \{(\alpha + \gamma) \div 1024\} \rceil + 1482 + \lceil \{(\beta + \delta) \div 1024\} \rceil$

(2) HiRDB/Parallel Server

Condition		Formula for determining memory size (KB)
32-bit mode	MGR	$1530 + \lceil \{\alpha \div 1024\} \rceil$
	Server machine containing input files	$1577 + \lceil \{(\beta + \delta) \div 1024\} \rceil$
	BES*	$2305 + \lceil \{\gamma \div 1024\} \rceil$
64-bit mode	MGR	$4732 + \lceil \{\alpha \div 1024\} \rceil$
	Server machine containing input files	$4663 + \lceil \{(\beta + \delta) \div 1024\} \rceil$
	BES*	$12220 + \lceil \{\gamma \div 1024\} \rceil$

* 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):

$$\{3056 + A + B + (516 \times a) + (572 \times b) + (312 \times c) + (144 \times d) + (8 \times e) + (1032 \times f) + (44 \times g) + (272 \times h) + (224 \times i) + (44 \times j) + (60 \times k) + (260 \times m) + (56 \times n) + (196 \times p) + (236 \times q) + (744 \times r) + (620 \times s)\} \times 2$$

β (bytes):

$$\{6908 + \alpha + (C \times t) + K + (48 \times a) + (22 \times b) + (8 \times e) + (240 \times i) + (48 \times j) + (4 \times k) + (224 \times m) + (47416 \times t) + (1032 \times u) + (4 \times v)\}$$

γ (bytes):

$$\{37700 + (\alpha \div 2) + C + D + F + H + P + Q + (80 \times a) + (1871 \times b) + (120 \times c) + (26 \times g) + (1532 \times i) + (36 \times j) + (44 \times k) + (1212 \times m) + (40 \times n) + (344 \times p) + (30 \times q) + (16 \times u) + (88 \times v) + (20 \times w)\}$$

δ (bytes):

$$\{69436 + \alpha + D + K + L + M + N + (32 \times a) + (88 \times c) + (4 \times g) + (2156 \times k) + (24 \times t) + (1024 \times u) + (4 \times v) + (50 \times y) + (50 \times z)\}$$

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 `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.
- 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 $(pd_utl_buff_size \times 1024 + 4096) \times 2$; otherwise, the value is 0:

- HiRDB/Parallel Server

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, 32500) for `pdrorg` (in bytes). For a non-FIX table, add $(a + 1) \times 4$.

F: $550 \times 1024 +$ work buffer size specified in the `sort` statement $\times 1024 + G$

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)
			FIX hash	Flexible hash	
<code>div</code> specified	1	Number of row partitions in the server for the table	HiRDB/Single Server: 1024 HiRDB/Parallel Server: (\uparrow 1024	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	$\div g \uparrow$ \times (number of table storage RDAREAs on the server)		

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.
 - 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).
 - The table contains an abstract data type column, or a unique index is defined.
1. `errwork` operand specified: Value of `errwork` operand \times 1024
 2. `errwork` operand omitted: Value of `pd_utl_buff_size` \times 1024 \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:

When the input data file is in DAT format:

Value specified in `maxreclen` operand \times 1024

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 1024
- 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 × number of index definitions > maximum number of processes that can be open - 576

19.2.4 Memory size required for the execution of the database reorganization utility (pdroorg)

The following tables show the formulas for determining the memory size required for the execution of the database reorganization utility (pdroorg). For details about the variables, see (3) *Variables used in the formulas* as follows.

(1) HiRDB/Single Server

Condition	Formula for determining memory size (KB)
32-bit mode	$5466 + \lceil \{(\alpha + \gamma) \div 1024\} \rceil + 1621 + \lceil \{(\beta + \delta) \div 1024\} \rceil$
64-bit mode	$18362 + \lceil \{(\alpha + \gamma) \div 1024\} \rceil + 1621 + \lceil \{(\beta + \delta) \div 1024\} \rceil$

(2) HiRDB/Parallel Server

Condition		Formula for determining memory size (KB)	
32-bit mode	MGR	$1509 + \lceil \{\alpha \div 1024\} \rceil$	
	-g option omitted	DS	$1413 + \lceil \{\beta \div 1024\} \rceil$
		BES ^{1,2}	$3505 + \lceil \{(\gamma + \delta) \div 1024\} \rceil$
	-g option specified	Server machine containing unload data file	$1413 + \lceil \{(\beta + \delta) \div 1024\} \rceil$
BES ²		$3505 + \lceil \{\gamma \div 1024\} \rceil$	
64-bit mode	MGR	$4761 + \lceil \{\beta \div 1024\} \rceil$	
	-g option omitted	DS	$16482 + \lceil \{\beta \div 1024\} \rceil$
		BES ^{1,2}	$12100 + \lceil \{(\alpha + \delta) \div 1024\} \rceil$
	-g option specified	Server machine containing unload data file	$4566 + \lceil \{(\beta + \delta) \div 1024\} \rceil$
		BES ²	$12100 + \lceil \{\gamma \div 1024\} \rceil$

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):

$$\{2592 + A + B + (116 \times a) + (260 \times b) + (6 \times c) + (272 \times d) + (44 \times g) + (272 \times h) + (224 \times i) + (44 \times j) + (60 \times k) + (260 \times m) + (56 \times n) + (196 \times p) + (236 \times q) + (744 \times r) + (620 \times s) + (24 \times t)\} \times 2$$

β (bytes):

$$\{40940 + \alpha + (C \times t) + (D \times t) + (136 \times a) + (56 \times g) + (2200 \times j) + (4 \times k) + (548 \times t)\}$$

γ (bytes):

$$\{101140 + (\alpha \div 2) + C + D + F + H + P + Q + (128 \times a) + (1949 \times b) + (120 \times c) + (154 \times g) + (336 \times i) + (216 \times j) + (32056 \times k) + (1212 \times m) + (131224 \times n) + (344 \times p) + (30 \times q) + (20 \times w)\}$$

δ (bytes):

$$\{33104 + \alpha + (K \times 2) + R + (204 \times a) + (688 \times b) + (306 \times c) + (44 \times g) + (272 \times h) + (224 \times i) + (44 \times j) + (56 \times n) + (716 \times r) + (152 \times v)\}$$

R: Depends on the specified -w option value:

- When -w dat is specified:
Total length of all columns obtained in Table 19-1 *Maximum lengths of converted character strings for columns (in DAT or extended DAT format)*
- When -w extdat is specified:
Total length of all columns obtained in Table 19-1 *Maximum lengths of converted character strings for columns (in DAT or extended DAT format)* $\times 2$
- When -w fixtext is specified:
Total length of all columns obtained in Table 19-2 *Maximum lengths of converted character strings for columns (fixed-length character format)*
- When -w bin is specified or the -w option is omitted: 0

Notes

- The other variables are explained in 19.2.3 *Memory size required for the execution of the database load utility (pdload)* and in (3) *Variables used in the formulas*.
- 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

Condition	Formula for determining memory size (KB)
32-bit mode	$\uparrow \{$ $4 \times a + 56016 \times b + 53016 \times c + 2440 \times d + 1724 \times e$ $+ (94008 \times f) \div 500 + (4008 \times g) \div 1000 + 441231 + h + i + j + k$ $\} \div 1024 \uparrow + 9.8$
64-bit mode	$\uparrow \{$ $4 \times a + 56024 \times b + 53024 \times c + 3040 \times d + 1736 \times e$ $+ (100016 \times f) \div 500 + (4012 \times g) \div 1000 + 451231 + h + i + j + k$ $\} \div 1024 \uparrow + 9.8$

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: $8 \times a + 30720$

Add this value if the `alter HiRDB mode to parallel` statement is used to migrate from a HiRDB/Single Server to a HiRDB/Parallel Server.

i: 46744

Add this value if the `create rdarea` statement is used to add a data dictionary LOB RDAREA.

j: 88064

Add this value if the `alter system` statement is used to change the dictionary table's reference privilege.

k: 54732

Add this value if the `alter system` statement is used to change the dictionary table's column attribute to MCHAR.

(2) HiRDB/Parallel Server

Condition		Formula for determining memory size (KB)
32-bit mode	DS	$\uparrow \{ 4 \times a + 56016 \times b + 53016 \times c + 2440 \times d + 1724 \times e + (94008 \times f) \div 500 + (4008 \times g) \div 1000 + 441231 + h + i + j + 108428 \times m \} \div 1024 \uparrow$
	BES	$\uparrow (4 \times a + 253266 + k) \div 1024 \uparrow$
	FES	0.52
	MGR	9.8
64-bit mode	DS	$\uparrow \{ 4 \times a + 56024 \times b + 53024 \times c + 3040 \times d + 1736 \times e + (100016 \times f) \div 500 + (4012 \times g) \div 1000 + 451231 + h + i + j + 108432 \times m \} \div 1024 \uparrow$
	BES	$\uparrow (4 \times a + 261112 + k) \div 1024 \uparrow$
	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: 46744

Add this value if the `create rdarea` statement is used to add a data dictionary LOB RDAREA.

i: 88064

Add this value if the `alter system` statement is used to change the dictionary table's reference privilege.

j: 54732

Add this value if the `alter system` statement is used to change the dictionary table's column attribute to MCHAR.

k: 1600

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):

$\uparrow (192 \times \text{number of moved RDAREAs} + 160 \times \text{total number of HiRDB files on moved RDAREAs} + 8 \times \text{total number of tables stored on moved RDAREAs} + 8 \times \text{total number of indexes stored on moved RDAREAs} + 8 \times \text{total number of LOB columns stored on moved RDAREAs}) \div 102400 \uparrow$

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	$16384^* + 1.5 \times \text{number of hosts} \times \text{number of HiRDB servers} \times \text{number of UAPs}$
HiRDB/Parallel Server	

* For the 64-bit mode, the value is 18640.

19.2.7 Memory size required for the execution of the database condition analysis utility (pddbst)

The following table shows the formulas for determining the memory size required for the execution of the database condition analysis utility (pddbst).

Condition		Formula for determining memory size (KB)
HiRDB/Single Server (32-bit mode)		Physical analysis of RDAREA: 4600 Logical analysis of RDAREA: $4400 + 1.4 \times a + \uparrow a \div 100 \uparrow \times 672$ Analysis in units of tables: $4400 + 1.4 \times b + \uparrow b \div 100 \uparrow \times 672$ Analysis in units of indexes: $4400 + 1.4 \times c + \uparrow c \div 100 \uparrow \times 672$ Analysis of cluster key: 4600
HiRDB/Single Server (64-bit mode)		Physical analysis of RDAREA: 17915 Logical analysis of RDAREA: $17915 + 1.4 \times a + \uparrow a \div 100 \uparrow \times 672$ Analysis in units of tables: $17915 + 1.4 \times b + \uparrow b \div 100 \uparrow \times 672$ Analysis in units of indexes: $17915 + 1.4 \times c + \uparrow c \div 100 \uparrow \times 672$ Analysis of cluster key: 17915
HiRDB/Parallel Server (32-bit mode)	DS	Physical analysis of RDAREA: 750 Logical analysis of RDAREA: $750 + 0.1 \times a$ Analysis in units of tables: $750 + 0.1 \times b$ Analysis in units of indexes: $750 + 0.1 \times c$ Analysis of cluster key: 750
	BES	Physical analysis of RDAREA: 650 Logical analysis of RDAREA: $650 + 0.1 \times a$ Analysis in units of tables: $650 + 0.1 \times b$ Analysis in units of indexes: $650 + 0.1 \times c$ Analysis of cluster key: 650
	MGR	Physical analysis of RDAREA: 3900 Logical analysis of RDAREA: $3550 + 1.3 \times a + \uparrow a \div 100 \uparrow \times 672$ Analysis in units of tables: $3550 + 1.3 \times b + \uparrow b \div 100 \uparrow \times 672$ Analysis in units of indexes: $3550 + 1.3 \times c + \uparrow c \div 100 \uparrow \times 672$ Analysis of cluster key: 3900

Condition		Formula for determining memory size (KB)
HiRDB/Parallel Server (64-bit mode)	DS	Physical analysis of RDAREA: 16312 Logical analysis of RDAREA: $16312 + 0.1 \times a$ Analysis in units of tables: $16312 + 0.1 \times b$ Analysis in units of indexes: $16312 + 0.1 \times c$ Analysis of cluster key: 16312
	BES	Physical analysis of RDAREA: 11812 Logical analysis of RDAREA: $11812 + 0.1 \times a$ Analysis in units of tables: $11812 + 0.1 \times b$ Analysis in units of indexes: $11812 + 0.1 \times c$ Analysis of cluster key: 11812
	MGR	Physical analysis of RDAREA: 4601 Logical analysis of RDAREA: $4601 + 1.3 \times a + \uparrow a \div 100 \uparrow \times 672$ Analysis in units of tables: $4601 + 1.3 \times b + \uparrow b \div 100 \uparrow \times 672$ Analysis in units of indexes: $4601 + 1.3 \times c + \uparrow c \div 100 \uparrow \times 672$ Analysis of cluster key: 4601

a : Number of tables in RDAREAs + number of indexes in RDAREAs

b : Number of table storage RDAREAs

c : Number of index storage RDAREAs

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		$11763 + 0.1 \times \text{number of table storage RDAREAs} + 0.07 \times \text{number of index storage RDAREAs} + 1.0 \times \text{number of indexes} + 0.04 \times \text{number of tables in schema} + 1.0 \times \text{number of servers} + 11 \times \text{number of columns}$
HiRDB/Parallel Server	BES	$11772 + 0.06 \times \text{number of table storage RDAREAs} + 0.05 \times \text{number of index storage RDAREAs} + 0.03 \times \text{number of indexes}$
	MGR	16402
	DS	$4384 + 0.1 \times \text{number of table storage RDAREAs} + 0.07 \times \text{number of index storage RDAREAs} + 1.0 \times \text{number of indexes} + 0.04 \times \text{number of tables in schema} + 1.0 \times \text{number of servers} + 11 \times \text{number of columns}$

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
 - 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
 - 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

Condition	Formula for determining memory size (KB)
Single server	$88 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$ $+ 49 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) + 64$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$ <p>Add the following value if this server machine contains backup files:</p> $+ 63 + \text{number of backup files} \times (2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) \times 2$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$ <p>Add the following value if differential backup files are to be collected:</p> $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32768 \uparrow \times 32 \times 2$

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

Condition	Formula for determining memory size (KB)
MGR	$88 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$ <p>Add the following value if differential backup files are to be collected:</p> $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32768 \uparrow \times 32 \times 2$
DS	$49 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) + 64$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$
BES	<p>Add the following value if this server machine contains backup files:</p> $+ 63 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $\times (\text{number of servers subject to backup} + 1)$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$

Condition	Formula for determining memory size (KB)
Server machine containing backup files	$63 + \text{number of backup files} \times 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $\times (\text{number of servers subject to backup} + 1)$ $+ \text{number of backup files} \times \{(\text{number of RDAREAs subject to backup} + 9) \div 10\} \times 6$ $+ \{(\text{total number of RDAREA component files subject to backup} + 25) \div 16\} \times 8 + 100$

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

Condition	Formula for determining memory size (KB)
Single server	$65 + \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 50$ $+ 98 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + c$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{number of RDAREAs subject to recovery} + 99) \div 100\} \times 5$ <p>Add the following value if this server contains backup files:</p> $+ 100 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ <p>Add the following value if unload log files are to be input:</p> $+ 57 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + 64$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{maximum number of concurrently executable transactions} + 99) \div 100\} \times 5$ $+ (\text{maximum page size for RDAREA subject to recovery}) \times 50$ $+ 0.6 \times \text{number of RDAREAs subject to recovery} + \text{size of work buffer for sorting (value of -y option)}$ <p>Add the following value if differential backup files are used for recovery:</p> $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32768 \uparrow \times 32$ $+ \uparrow (32 \times \text{differential backup count}) \div 1024 \uparrow$

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 $\text{MAX}(64, \text{write buffer size})$. If not, this value is 60.

The write buffer size is the value specified by the `-y` option.

(2) HiRDB/Parallel Server

Condition	Formula for determining memory size (KB)
MGR	$65 + \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 50$ <p>Add the following value if differential backup files are used for recovery:</p> $+ 32 \times 2 + \uparrow (512 + 128 \times \text{number of RDAREAs subject to backup}) \div 32768 \uparrow \times 32$ $+ \uparrow (256 + 128 \times \text{number of RDAREAs subject to backup} + a + 8 \times b) \div 32768 \uparrow \times 32$ $+ \uparrow (32 \times \text{differential backup count}) \div 1024 \uparrow$
DS	$35 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) + 100$ $+ 98 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + c$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{number of RDAREAs subject to recovery} + 99) \div 100\} \times 5$ <p>Add the following value if this server contains backup files:</p> $+ 100 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ <p>Add the following value if unload log files are to be input:</p> $+ 57 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + 64$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{maximum number of concurrently executable transactions} + 99) \div 100\} \times 5$ $+ (\text{maximum page size for RDAREAs subject to recovery}) \times 50$ $+ 0.6 \times \text{number of RDAREAs subject to recovery} + \text{size of work buffer for sorting (value of -y option)}$

Condition	Formula for determining memory size (KB)
BES	$98 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + c$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{number of RDAREAs subject to recovery} + 99) \div 100\} \times 5$ <p>Add the following value if this server contains backup files:</p> $+ 100 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ <p>Add the following value if unload log files are to be input:</p> $+ 57 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size})$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6 + 64$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$ $+ \{(\text{maximum number of concurrently executable transactions} + 99) \div 100\} \times 5$ $+ (\text{maximum page size for RDAREAs subject to recovery}) \times 50$ $+ 0.6 \times \text{number of RDAREAs subject to recovery} + \text{size of work buffer for sorting (value of -y option)}$
Server machine containing backup files	$100 + 2 \times \text{MAX}(32, \text{value of pd_utl_buff_size}) \times \text{number of servers subject to recovery}$ $+ \{(\text{number of RDAREAs subject to recovery} + 9) \div 10\} \times 6$ $+ \{(\text{number of RDAREA component files subject to recovery} + 25) \div 16\} \times 8 + 100$

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.

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

Location	Type	Value (KB)
Single server	Exporting table definition information	17302
	Importing table definition information	$17318 + 0.4 \times (a + b) + 0.1 \times c$
	Exporting procedure	$17302 + d$
	Importing procedure	$17318 + 0.1 \times e$

a: Number of columns in the table to be imported

b: Number of partitions for the table to be imported

c: If a LOB column is defined for the table being imported, number of LOB RDAREAs containing LOB data

d: Length of source of stored procedure¹ (KB)

e: Number of resources used by stored procedure being imported²

¹ Value of the SOURCE_SIZE column in the SQL_ROUTINES table.

² Value of the N_RESOURCE column in the SQL_ROUTINES table.

(2) HiRDB/Parallel Server

Location	Type	Value (KB)
MGR		4067
DS	Exporting table definition information	17043
	Importing table definition information	$16139 + 0.4 \times (a + b) + 0.1 \times c + 0.25 \times f$
	Exporting procedure	$17043 + d$
	Importing procedure	$16139 + 0.1 \times e$
BES	Exporting table definition information	5142
	Importing table definition information	$5142 + 0.4 \times (a + b) + 0.1 \times c + 0.25 \times f$
	Exporting procedure	$5142 + d$

Location	Type	Value (KB)
	Importing procedure	$5142 + 0.1 \times e$

a: Number of columns in table to be imported

b: Number of partitions for the table to be imported

c: If a LOB column is defined for the table being imported, number of LOB RDAREAs containing LOB data

d: Length of source of stored procedure¹ (KB)

e: Number of resources used by stored procedure being imported²

f: Number of partitions with boundary value specified for the table being imported

¹ Value of the SOURCE_SIZE column in the SQL_ROUTINES table.

² Value of the N_RESOURCE column in the SQL_ROUTINES table.

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		<i>a</i>
HiRDB/Parallel Server	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

Formula for determining memory size (KB)
$8756^3 + 536 + 0.02 \times \text{number of columns} + 0.2 \times \text{number of target RDAREAs}$ $+ 1.7 \times \text{number of source RDAREAs} + 0.26 \times \text{number of target RDAREAs} \times \text{number of indexes}$ $+ (0.09 + \uparrow \text{average index statement file length} \div \uparrow 1024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div \uparrow 1024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ \uparrow (\text{length of control information file} + \text{length of execution results file}) \div 1024 \uparrow$ $+ 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ $+ 550 \times 1024 + \text{size of work file for sorting}^1$ <p>-n option specified:</p> $+ \text{page length of RDAREA}^2 \times \text{number of batch input/output buffer sectors} \times y$ <p>Applicable table containing LOB columns:</p> $+ 64 + 0.01 \times \text{number of LOB columns} + 0.18 \times \text{number of target RDAREAs}$ $+ 0.09 \times \text{number of source RDAREAs} + 0.08 \times \text{number of LOB storage RDAREAs}$ <p>When the target table contains BINARY columns:</p> $+ 33 \times (\text{number of BINARY columns} \times \text{number of target RDAREAs} \times \text{number of source RDAREAs})$ <p>Applicable table containing an abstract data type provided by plug-in:</p> $+ 40 + (0.27 + 2 \times \text{length of abstract data type}) \times \text{number of abstract data-type columns}$ $+ 0.3 \times \text{number of unld_func statements}$ $+ (128 + 0.11 \times \text{number of LOB attributes} + 0.1 \times \text{number of functions specifying unld_func}$ $+ 0.07 \times \text{number of abstract data type attributes}) \times \text{number of abstract data-type columns}$ $+ (33 \times \text{number of BINARY attributes} \times \text{number of target RDAREAs} \times \text{number of source RDAREAs}) \times 2$ $+ 0.01 \times \text{number of plug-in indexes} + 0.19 \times \text{number of unld_func statements}$ $+ (\uparrow \text{average length of unld_func statements} \div 1024 \uparrow \times \text{number of unld_func statements})$ $+ (\uparrow \text{average length of reld_func statements} \div 1024 \uparrow \times \text{number of reld_func statements})$ $+ \text{number of abstract data-type columns} \times 1 + (\text{number of LOB attributes} \times 0.05) \times \text{number of RDAREAs}$ $+ \text{number of data type plug-ins} \times 10 + \text{number of plug-in indexes} \times 10$ $+ \text{memory required for plug-ins}$ <p>When the specified index creation method is the batch index creation mode or the index information output mode and the following condition is satisfied:</p> $\text{Number of table partitions} \times \text{number of index definitions} > \text{maximum number of processes that can be open} - 576$ $+ 2048$

y: Use one of the following values:

- When the rebalancing facility is used with a FIX hash-partitioned table

$$(\uparrow 1024 \div \text{number of storage RDAREAs for the entire table} \uparrow) \times \text{number of table storage RDAREAs in the corresponding server}$$

- Other than the above

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 14272.

(2) For a HiRDB/Parallel Server

Condition	Formula for determining memory size (KB)
MGR	$1498^3 + 2 + 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ $+ (0.09 + \uparrow \text{average index statement file length} \div 1024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div 1024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ \uparrow (\text{length of control information file} + \text{length of execution results file}) \div 1024 \uparrow$ <p>Applicable table containing LOB columns:</p> $+ 0.08 \times \text{number of LOB storage RDAREAs}$ <p>Applicable table containing an abstract data type provided by plug-in:</p> $+ 0.19 \times \text{number of unl_d_func statements}$ $+ (\uparrow \text{average length of unl_d_func statement} \div 1024 \uparrow \times \text{number of unl_d_func statements})$ $+ (\uparrow \text{average length of rel_d_func statement} \div 1024 \uparrow \times \text{number of rel_d_func statements})$
DS	$1455^4 + 32 + 0.33 \times \text{number of target BESs} + 0.3 \times \text{number of source BESs}$ $+ 0.2 \times \text{number of target RDAREAs} + 0.22 \times \text{number of source RDAREAs} + 0.34 \times \text{number of FESs}$ $+ (0.09 + \uparrow \text{average index statement file length} \div 1024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div 1024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ <p>Applicable table containing LOB columns:</p> $+ 0.08 \times \text{number of LOB storage RDAREAs}$

Condition	Formula for determining memory size (KB)
	<p>When the target table contain BINARY columns: $+ 33 \times (\text{number of BINARY columns} \times \text{number of target RDAREAs} \times \text{number of source RDAREAs})$</p> <p>Applicable table containing an abstract data type provided by plug-in: $+ 0.01 \times \text{number of target BESs} + 0.19 \times \text{number of unld_func statements}$ $+ (\uparrow \text{average length of unld_func statements} \div \uparrow 1024 \uparrow \times \text{number of unld_func statements})$ $+ (\uparrow \text{average length of reld_func statements} \div \uparrow 1024 \uparrow \times \text{number of reld_func statements})$ $+ \text{number of abstract data type columns} \times 1 + (\text{number of LOB attributes} \times 0.05) \times \text{number of RDAREAs}$ $+ \text{number of data type plug-ins} \times 10 + \text{number of plug-in indexes} \times 10$</p>
BES	<p>$6601^5 + 50 + (517 + 0.01 \times \text{number of columns}) \times \text{number of target BESs}$ $+ (33 + 0.01 \times \text{number of columns}) \times \text{number of source BESs} + 0.2 \times \text{number of target RDAREAs}$ $+ 1.7 \times \text{number of source RDAREAs} + 0.01 \times \text{number of columns}$ $+ 0.26 \times \text{number of target RDAREAs} \times \text{number of indexes}$ $+ (0.09 + \uparrow \text{average index statement file length} \div 1024 \uparrow) \times \text{number of index statements}$ $+ (0.02 + \uparrow \text{average directory length} \div 1024 \uparrow) \times (\text{number of idxwork statements} + \text{number of sort statements})$ $+ 0.05 \times \text{number of columns} + 0.05 \times \text{number of RDAREAs} + 0.15 \times \text{number of indexes}$ $+ 0.05 \times \text{number of index storage RDAREAs}$ $+ 550 \times 1024 + \text{size of work file for sorting}^1$</p> <p>-n option specified: $+ \text{page length of RDAREA}^2 \times \text{number of batch input/output buffer sectors} \times y$</p> <p>Applicable table containing LOB columns: $+ 32 + 0.01 \times \text{number of LOB columns} + (32 + 0.01 \times \text{number of LOB columns}) \times \text{number of target BESs}$ $+ 0.18 \times \text{number of target RDAREAs} + 0.1 \times \text{number of source RDAREAs}$ $+ 0.08 \times \text{number of LOB storage RDAREAs}$</p> <p>When the target table contain BINARY columns: $+ 33 \times (\text{number of BINARY columns} \times \text{number of target RDAREAs} \times \text{number of source RDAREAs})$</p>

Condition	Formula for determining memory size (KB)
	<p>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 $+ (33 \times \text{number of BINARY attributes} \times \text{number of target RDAREAs} \times \text{number of source}$ $\text{RDAREAs}) \times 2$ $+ 0.01 \times \text{number of plug-in indexes} + 0.19 \times \text{number of unld_func statements}$ $+ (\uparrow \text{average length of unld_func statements} \div 1024 \uparrow \times \text{number of unld_func statements})$ $+ (\uparrow \text{average length of reld_func statements} \div 1024 \uparrow \times \text{number of reld_func statements})$ $+ \text{number of abstract data-type columns} \times 1 + (\text{number of LOB attributes} \times 0.05) \times \text{number}$ of RDAREAs $+ \text{number of data type plug-ins} \times 10 + \text{number of plug-in indexes} \times 10$ $+ \text{memory required for plug-ins}$</p> <p>When the specified index creation method is the batch index creation mode or the index information output mode and the following condition is satisfied: $\text{Number of table partitions} \times \text{number of index definitions} > \text{maximum number of processes that}$ $\text{can be open} - 576 + 2048$</p>

y: Use one of the following values:

- When the rebalancing facility is used for FIX hash partitioning tables
 $(\uparrow 1024 \div \text{number of storage RDAREAs for the entire table} \uparrow) \times \text{number of table storage RDAREAs in the corresponding server}$
- Other than the above

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 4660.

⁴ For the 64-bit mode, the value is 15830.

⁵ For the 64-bit mode, the value is 11804.

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 (32-bit mode)		$800 + W$
HiRDB/Single Server (64-bit mode)		$1897 + W$
HiRDB/Parallel Server (32-bit mode)	MGR	$800 + X$
	Servers specified with the <code>-s</code> option ¹	Y
	BES ²	Z
HiRDB/Parallel Server (64-bit mode)	System manager	$1953 + X$
	Servers specified with the <code>-s</code> option ¹	Y
	Back-end server 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	32-bit mode	$11664 + \alpha$	
	64-bit mode	$22696 + \alpha$	
HiRDB/Parallel Server	32-bit mode	MGR	$7577 + \alpha$
		DS	$5772 + \beta$
	64-bit mode	MGR	$9240 + \alpha$
		DS	$12776 + \beta$

α : The value obtained by the following formula:

2175

+ 0.14 × number of columns

+ 0.09 × number of table storage RDAREAs

+ 0.23 × number of indexes

+ 0.09 × number of index storage RDAREAs

+ 0.09 × number of LOB storage RDAREAs

Number of foreign keys

+ $\sum_{r=1} (42 + 0.47 \times \text{number of foreign key component columns } r)$

Number of foreign keys

+ $\sum_{c=1} (5 + 0.29 \times \text{number of columns in the search condition } c + 0.85$

× number of ANDs and ORs in the search condition c + length of the search condition c)

β : The value obtained from the following formula

0.2

+ 0.02 × number of table storage RDAREAs

+ 0.02 × number of index storage RDAREAs

+ 0.02 × number of LOB column storage RDAREAs

Chapter

20. Determining Environment Variables Related to the Number of Resources

This chapter describes the procedures for determining the environment variables related to the number of resources.

This chapter contains the following sections:

- 20.1 HiRDB/Single Server
- 20.2 HiRDB/Parallel Server

20.1 HiRDB/Single Server

Table 20-1 shows the formulas for determining the values of environment variables related to the number of resources. If the obtained value is greater than the default value for the corresponding environment variable, set the obtained value in the environment variable. Otherwise, a problem may result, such as HiRDB cannot be started or a utility cannot be executed.

Table 20-1: Values of environment variables related to the number of resources (HiRDB/Single Server)

Type	Formula	Corresponding environment variable	Default
Number of message queue identifiers	$(7 + f) \times a + 25$	PDUXPLMSGMNI	50
Number of message queue tables	MAX($b \times$ number of message queue identifiers, total value of <code>pdbuffer</code> with <code>-m</code> specified)	PDUXPLMSGTQL	80
Number of semaphore identifiers	$\{ \uparrow \{ 2 \times (b + 3) + 12 \} \div 64 \uparrow + \uparrow c \div 64 \uparrow + g + 5 \} \times a + 2 + d$	PDUXPLSEMMAX	64
Shared memory usage count	$2 \times e + 16$	PDUXPLSHMMAX	4096

a: 1 for a single server

b: Value of the `pd_max_users` operand

c: Number of `pdbuffer` operands defined

d: Add when the system switchover facility is being used. Determine the value from the following table:

Condition		Value of d
<code>pd_ha_acttype=monitor</code> (or default)		0
<code>pd_ha_acttype=server</code>	<code>pd_ha_agent=standbyunit</code>	1
	Omit <code>pd_ha_agent</code>	<code>pd_ha_server_process_standby=Y</code> (or default)
		<code>pd_ha_server_process_standby=N</code>

e: Value of the `pd_max_server_process` operand

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

20.2 HiRDB/Parallel Server

Table 20-2 shows the formulas for determining the values of environment variables related to the number of resources. If the obtained value is greater than the default value for the corresponding environment variable, set the obtained value in the environment variable. Otherwise, a problem may result; for example, HiRDB might be unable to start or a utility might be unable to execute.

These formulas provide the values required for a server machine.

Table 20-2: Values of environment variables related to the number of resources (HiRDB/Parallel Server)

Type	Formula	Corresponding environment variable	Default
Number of message queue identifiers	b $\sum_{i=1} V_i + 2 \times a + 3 \times b + c + d + e + 24$	PDUXPLMSGMNI	50
Number of message queue tables	MAX($h \times$ number of message queue identifiers, total value of <code>pdbuffer</code> with <code>-m</code> specified)	PDUXPLMSGTQL	80
Number of semaphore identifiers	<ul style="list-style-type: none"> When the standby-less system switchover (effects distributed) facility is not used b $\sum_{i=1} \{ \uparrow (S_i + T_i + U_i) \div 64 \uparrow + W_i \}$ $+ 6 \times b + 2 + f$ <ul style="list-style-type: none"> When the standby-less system switchover (effects distributed) facility is used b $\sum_{i=1} \{ \uparrow \{ Y_i \times (j+k) \} \div 64 \uparrow + W_i \}$ $+ 6 \times b + 2 + f$	PDUXPLSEMMAX	64
Shared memory usage count	$2 \times g + 16 \times b$	PDUXPLSHMMAX	4096

a : Number of front-end servers in the server machine

b: Number of dictionary servers and back-end servers in the server machine

c: 2 for the front-end server; 0 for other

d: 5 for the dictionary server; 0 for other

e: 5 for the back-end server; 0 for other

f: Add when the system switchover facility is being used. Determine the value from the following table:

Condition		Value of <i>f</i>	
pd_ha_acttype=monitor (or default)		0	
pd_ha_acttype=server	pd_ha_agent=standbyunit	1	
	pd_ha_agent omitted	pd_ha_server_process_standby=Y (or default)	1
		pd_ha_server_process_standby=N	0

g: Value of the pd_max_server_process operand

h: Value of the pd_max_users operand

j: Number of host BESs

k: Number of guest BESs

S_i: Number of pdbuffer -r operands defined for the RDAREAs in each server

T_i: Number of pdbuffer -i operands defined for the RDAREAs in each server

U_i: Number of pdbuffer -o operands defined

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

Y_i: Number of -c options specified in the pdbuffer operand

Chapter

21. Windows Registry Settings

This chapter explains how to determine the Windows registry settings.

21.1 Estimating desktop heap settings

21.1 Estimating desktop heap settings

(1) Determining the desktop heap settings

The size of the desktop heap can be set in the registry and can be changed with a registry editor. This specification of the size of the heap is set for each account, with a 48-megabyte maximum for the entire server machine. The size of the desktop heap used by HiRDB processes is approximately 5 kilobytes per process. Use the following formula to determine the target registry setting:

Formula

Number of activated HiRDB processes (value of `pd_max_server_process`) × 5 (KB)

Also consider the following:

- For Windows 2000, the default size for the desktop heap is 512 kilobytes. Therefore, on systems for which the value of `pd_max_server_process` is 100 or greater, the desktop heap can become insufficient. If the number of active processes becomes close to 100, a shortage occurs on the desktop heap.
- If the setting is too large, it can be subject to influence by the action of other program products running under other accounts. For this reason, you should not set the value higher than is necessary.
- The processes activated by HiRDB are not limited to OS commands executed by HiRDB server processes and by users, and to utilities. Commands executed to collect maintenance information when a HiRDB process terminates abnormally are also processes that HiRDB activates and which use the desktop heap. If a large number of HiRDB server processes terminate abnormally or are cancelled, and if the cause is an insufficient desktop heap (`end state=0x8000` indicated in the `KFPS01820-E` message indicating that server processes have stopped), do not adjust the size of the desktop heap, but change the system common definition so that maintenance information is not collected.
- HiRDB running on one system account uses the same desktop heap used by programs running under the OS or on other system accounts. Therefore, if, for example, you are using Windows 2000 with a desktop heap size of 512 kilobytes, then even if HiRDB has activated fewer than 100 processes, when new HiRDB processes are activated it may be possible to detect an insufficient desktop heap.

(2) Changing the desktop heap setting

This section describes the procedure for changing the desktop heap. If you misuse the registry editor, it is possible to cause serious problems, so great caution should be

exercised. In addition, the setting for the desktop heap must be adjusted for the particular operating environment.

1. Stop the HiRDB service.
2. Use the registry editor to change the setting for the desktop heap for the non-interactive desktop.

■ Registry key

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\SubSystems
```

■ Registry setting

```
Windows
```

■ Value

```
%SystemRoot%\system32\csrss.exe ObjectDirectory
```

```
=\Windows SharedSection=1024,3072,512
```

```
Windows=On SubSystemType=Windows ServerDll=basesrv,1
```

```
ServerDll=winsrv:UserServerDllInitialization,3
```

```
ServerDll=winsrv:ConServerDllInitialization,2
```

```
ProfileControl=Off MaxRequestThreads=16
```

What you change is the third parameter of `SharedSection` (underlined). If this parameter is set to 512, the system allocates a 512-kilobyte heap to each desktop. If the third parameter is omitted, the second value (3,072 kilobytes) is allocated. Change the third parameter to the value determined in (1) *Determining the desktop heap settings*. In this case, it is imperative that the value specified for `pd_max_server_process` be estimated correctly.

3. Restart the operating system.
4. Restart the HiRDB service.

Chapter

22. Sample Files

This chapter describes the sample files provided by HiRDB (sample database, configuration, and UOC).

This chapter contains the following sections:

- 22.1 Overview of sample files
- 22.2 Table definition information
- 22.3 Use of the sample files

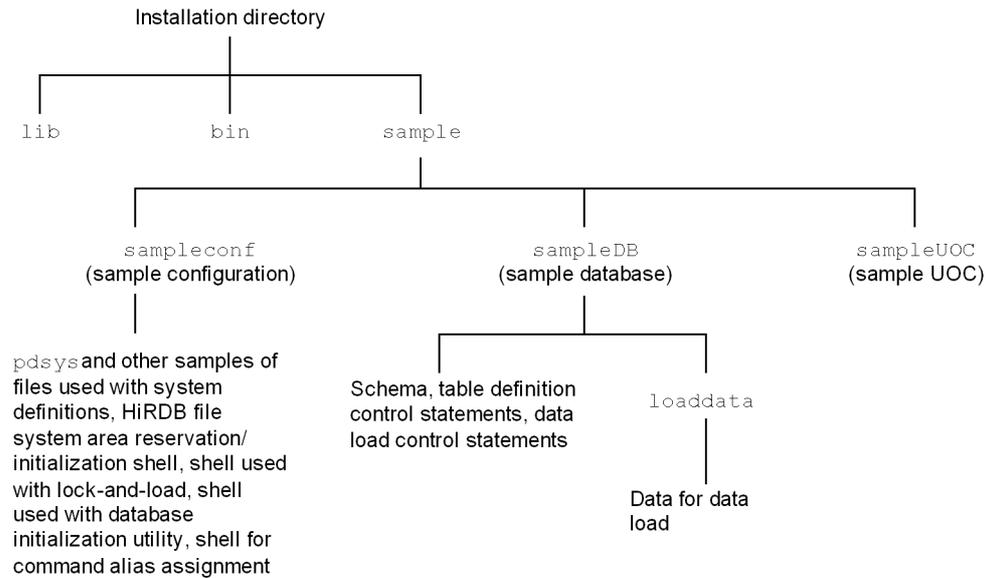
22.1 Overview of sample files

HiRDB provides the following sample files:

- Sample database
- Sample configuration
- Sample UOC

Figure 22-1 shows the directory structure of the sample files. All the directories following `sample` are located under the installation directory.

Figure 22-1: Directory structure of sample files



The following information assumes that the HiRDB environment has already been set up by executing the `SPsetup.bat` batch file. For details about `SPsetup.bat`, see *C. Setting Up an Environment Using a Batch File*.

22.1.1 Names of sample files

This section describes the names of the following sample files:

- Sample database
- Sample configuration
- Sample UOC

(1) Name of sample database file

Table 22-1 lists and explains the directories and files used with the sample database.

Table 22-1: Directories and files used with sample database

Name of directory or file	Contents	Remarks
tblecreate	Table definition statements (including schema definitions)	pddef input format
loaddata	Input data for data loading	Directory
CONTROL_FILE	Control statements for data loading	—

(2) Names of sample configuration files

Table 22-2 lists the contents of 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 22-2: Contents of sample configuration

Classification	Contents	File name ¹
System definitions	System common definition	pdsys
	Unit control information definition	pdutsys
	Single server definition	sds01
Allocation/ initialization of HiRDB file system areas	Batch file for creating the following HiRDB file system areas: <ul style="list-style-type: none"> HiRDB file system area for RDAREAs HiRDB file system area for work table files 	fmkfile.bat
	Batch file for creating the following HiRDB file system area: <ul style="list-style-type: none"> HiRDB file system area for system files 	fmkfs.bat
	Batch file for initializing system log, synchronization point dump, and status files	sysfint.bat
Log unloading	Batch file for unloading system log files Shell for unloading system log files	logunld.bat ²
Database initialization utility (pdinit)	Batch file for creating RDAREAs. This batch file executes the database initialization utility (pdinit).	initdb.bat
	Batch file containing the database initialization utility's (pdinit's) control statements.	mkinit

Classification	Contents	File name ¹
Execution of operation commands under aliases	Sample of executable file for executing operation commands under aliases (batch file)	aliascmd.bat ²

¹ These are files names for a HiRDB/Single Server. For a HiRDB/Parallel Server, see %PDDIR%\HiRDEF\readme.txt.

² logunld.bat is stored in the %PDDIR%\sample directory and aliascmd.bat is stored in the %PDDIR%\sample\sampleconf directory.

(3) Sample UOC files

The UOC shown below is provided. Table 22-3 shows the contents of 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 8 Command Reference*.

Table 22-3: Contents of sample UOC

File name	Contents
sample1.c	Example of UOC for entering DAT-format input files by the database load utility (pdload)
sampleA.c	Example of UOC that prevents output of unneeded data to an unload file.

22.2 Table definition information

Table 22-4 lists the types of table provided as the sample database. Table 22-5 shows the column attributes of the tables.

All the tables presented here have the `FIX` attribute. If you execute the `SPsetup.bat` file without customizing it, these tables are stored in user `RDAREA RDDATA10` and their indexes in user `RDAREA RDINDX10`.

Table 22-4: Tables provided as sample database

Table name	Contents	Number of rows
CUSTOM	Customer master	100
GOODS	Product master	100
VENDOR	Vendor master	50
TAKEODR	Orders received	Data loading not applicable
STOCK	Stock	100
WAREHUS	Warehousing	Data loading not applicable
SHIPMNT	Shipments	Data loading not applicable
SENDODR	Orders placed	Data loading not applicable
LAYIN	Purchasing	Data loading not applicable

Table 22-5: Column attributes and indexes of the tables

Table name	Column name	Column attribute	Index name
CUSTOM	CUSTOM_CD	CHAR (5)	UNIQUE CLUSTER KEY CUSTOMX
	CUSTOM_NAME	CHAR (30)	
	TELNO	CHAR (12)	
	ZIPCD	CHAR (3)	
	ADDRESS	CHAR (30)	

22. Sample Files

Table name	Column name	Column attribute	Index name
GOODS	PRODUCT_CD	CHAR (6)	UNIQUE CLUSTER KEY GOODSX
	PRODUCT_NAME	CHAR (30)	
	PRICE	DECIMAL (7,0)	
	VENDOR_CD	CHAR (5)	
VENDOR	VENDOR_CD	CHAR (5)	UNIQUE CLUSTER KEY VENDORX
	VENDOR_NAME	CHAR (30)	
	TELNO	CHAR (12)	
	ZIPCD	CHAR (3)	
	ADDRESS	CHAR (30)	

Table name	Column name	Column attribute	Index name
TAKEODR	ORDER_ACCEPTED_CD	CHAR (7)	CLUSTER KEY
	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)	
	DELIVERY_DATE	CHAR (6)	
STOCK	PRODUCT_CD	CHAR (6)	
	STOCK	DECIMAL (7,0)	
	RESERVED_QUANTITY	DECIMAL (7,0)	
	ORDER	DECIMAL (7,0)	
	VENDOR_CD	CHAR (5)	
WAREHUS	PRODUCT_CD	CHAR (6)	
	WAREHOUSE	DECIMAL (7,0)	
	LAY_IN_NO	INTEGER	
	WAREHOUSE_DATE	CHAR (6)	
SHIPMNT	PRODUCT_CD	CHAR (6)	
	SHIPMENT	DECIMAL (7,0)	
	ORDER_ACCEPTED_CD	CHAR (7)	
	ORDER_ACCEPTED_DATE	CHAR (6)	
SENDODR	ORDER_NO	INTEGER	CLUSTER KEY SENDODRX
	VENDOR_CD	CHAR (5)	
	PRODUCT_CD	CHAR (6)	
	ORDER_QUANTITY	DECIMAL (7,0)	
	ORDER_DATE	CHAR (6)	
	DELIVERY_DATE	CHAR (6)	

22. Sample Files

Table name	Column name	Column attribute	Index name
LAYIN	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)	

22.3 Use of the sample files

Table 22-6 shows the batch files for creating the sample database. These batch files are located in the %PDDIR%\sample\tools directory (e.g., C:\win32app\hitachi\hirdb_s\sample\tools).

Table 22-6: Batch files for creating the sample database

File name	Description
sampleDB1.bat sampleDB2.bat	Use these batch files to set the table creator to root. <ul style="list-style-type: none"> sampleDB1.bat defines the sample database. sampleDB2.bat loads data to the sample database.
sampleDB3.bat sampleDB4.bat	Use these batch files to set the table creator to USER1. <ul style="list-style-type: none"> sampleDB3.bat grants CONNECT and schema definition privileges to USER1 to define the sample database. sampleDB4.bat loads data to the sample database.

Notes

- If you are customizing any of these batch files, copy them under a different name and then customize the file contents. If you customize the original batch file, it is overwritten during update installation of HiRDB.
- The following information assumes that a HiRDB/Single Server has been installed in the installation directory (C:\win32app\hitachi\hirdb_s). Replace the pathnames and HiRDB types (HiRDB/Parallel Server) with those you are actually using, if necessary. For a HiRDB/Parallel Server, see %PDDIR%\HiRDEF\readme.txt.

22.3.1 Creating the sample database

To create the sample database:

- Register the table definitions and user.

Double-click sampleDB1.bat.

The message `Press any key to continue` is displayed on the screen. If there is no problem, press the **ENTER** key. If there is a problem, solve it, and then reexecute. In this case, the table creator is set to root. To set the table creator to USER1, double-click sampleDB3.bat instead of sampleDB1.bat.

- Load data to the tables.

Double-click sampleDB2.bat.

The message `Press any key to continue` is displayed on the screen. If there

is no problem, press the **ENTER** key. If there is a problem, solve it, then reexecute. In this case, the table creator is set to `root`. If you have set the table creator to `USER1` (by using `sampleDB3.bat`), double-click `sampleDB4.bat` instead of `sampleDB2.bat`.

22.3.2 Customizing the sample database

You can customize the batch files for creating the sample database to define tables and load data to the tables. To customize batch files, you must be familiar with SQL and the database load utility (`pdload`). For details about SQL, see the manual *HiRDB Version 8 SQL Reference*. For details about the database load utility (`pdload`), see the manual *HiRDB Version 8 Command Reference*.

Note

At the point when HiRDB has been installed and has just started, the registered authorization identifier and password are both `root`. The `root` user has the DBA privilege. Therefore, `root` is the authorization identifier and password in the client environment definitions (`hirdb.ini`) during the sample database setup.

(1) Adding a new user

The following shows how to customize the batch files to add a new user:

To customize the batch file:

1. Modify the contents of `sampleDB1.bat`. Change the line executing the table definition to a comment statement so that only the privilege definition is executed:

```
pddef<%PDDIR%\sample\sampleDB\tblecreate
```

↓

```
rem pddef<%PDDIR%\sample\sampleDB\tblecreate
```

2. Modify the contents of the sample privilege definition `%PDDIR%\sample\sampleDB\gr_USER1` as follows:
 - To add a new user, define the `CONNECT` privilege with the `GRANT` statement.
 - Define the schema definition privilege with the `GRANT` statement, as required.
1. When the modification is completed, double-click `sampleDB1.bat`. When a message is displayed on the screen indicating normal termination of the database definition utility (`pddef`), the addition of a new user and the privilege definition is completed.

Note that `sampleDB3.bat` adds `USER1`.

(2) Defining tables and indexes

The following shows how to customize the batch files to define tables and indexes:

To customize the batch file:

1. To add a new user and use this user to define schemas, etc., modify the authorization identifier and password in the client environment definitions (`hirdb.ini`). (Change the value of the `PDUSER` environment variable in the `hirdb.ini` file, which is located immediately below the `windir` environment variable, from `root` to the added authorization identifier and password.)
2. Modify the contents of `sampleDB1.bat`. Change the line executing the privilege definition to a comment statement. If the table definition part is set as a comment statement, uncomment it.
3. Modify the contents of `%PDDIR%\sample\sampleDB\tblecreate`, which is provided as a sample table definition. Modify the schemas and tables to be defined.
4. When the modification is completed, double-click `sampleDB1.bat`. Customization is completed if a message is displayed on the screen indicating normal termination of the database definition utility (`pddef`). If you have defined multiple tables and there is an error in any of the tables, modify the contents of `tblecreate`, then reexecute.

(3) Loading data to the tables

The following shows how to customize the batch files to load data to tables:

To customize the batch file:

1. The authorization identifier and password are set to `root` in `sampleDB2.bat`. Change them to the actual table owner's authorization identifier and password as follows:


```
set PDUSER="root"/"root"
```

↓

```
set PDUSER="authorization-identifier"/"password"
```
2. The sample batch file provided loads data to four tables (`CUSTOM`, `GOODS`, `VENDOR`, and `STOCK`). Modify or change an applicable line to a comment statement as required.
3. `@echo`

```
source%PDDIR%\sample\sampleDB\loaddata\GOODS.CSV>%PDDIR%\TMP\LOD
```

`specifies control information in the LOD control information file. Modify the specified control information as required.`

4. %PDDIR%\bin\pdload -i s -e GOODS %PDDIR%\TMP\LOD executes the database load utility (pdload). Modify information such as table name, GOODS, if necessary.
5. %PDDIR%\sample\sampleDB\loaddata\GOODS.csv is the input file. Create input data in this file. The sample data provided uses DAT-format GOODS.CSV. Use this as an example to create input data.
6. When you finish creating the input data, double-click sampleDB2.bat. Check the message indicating the execution status of the database load utility (pdload).

sampleDB4.bat assumes that USER1 executes the data load operation.

(4) Creating a batch file for executing 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:

- Use 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 batch file for this purpose.

(a) Name of sample batch file provided by HiRDB

Table 22-7 shows the sample batch file provided by HiRDB. This file is stored in the following directory:

- C:\win32app\hitachi\hirdb_s\sample\sampleconf

Table 22-7: Sample batch file for executing commands under aliases

File name	Contents	Notes
aliascmd.bat	Sample batch file	Do not copy this file in the bin or lib directory under the HiRDB directory

(b) Procedure for creating an alias for a command

To create an alias for an operation command:

1. Copy the sample batch file into a desired directory. To create aliases for multiple commands, copy the file 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 `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 Figure 22-2.

Figure 22-2: Sample batch file

```
@echo off
setlocal

rem#set HiRDB command here
set HiRDB_COMMAND= [ cc...cc ] ← Specify the HiRDB operation command name here (to
set PARAM= change pdmod to hirmod, specify pdmod)
:GETPARAM
if "%1"==" " goto EXEC
set PARAM=%PARAM% %1
shift
goto GETPARAM
:EXEC
%PDDIR%\bin\%HIRDB_COMMAND%%PARAM%
endlocal
```

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 may be deleted in their entirety during uninstallation. For this reason, sample files must not be copied into these directories.
3. To call a sample batch file from within another batch file, use `call`. Otherwise, the system may not be able to return control to the initial batch file.
4. The contents of the sample files must not be changed, except for setting a HiRDB operation command name.
5. 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.
6. 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.

22.3.3 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 22-8 lists the names and sizes of the HiRDB file system areas that are used with the sample database.

Table 22-8: Names and sizes of HiRDB file system areas created by SPsetup.bat

Type of HiRDB file system area	Size of HiRDB file system area (MB)	HiRDB file system area name
System files	Small → 74 Medium → 148 Large → 296	rdsys011* rdsys012* rdsys013* rdsys014* rdsys015* rdsys016*
System RDAREAs	Small → 20 Medium → 40 Large → 80	rdsys02*
Work table files	—	rdsys03*
User RDAREAs	Small → 40 Medium → 80 Large → 160	rdsys04*
User LOB RDAREA	40	rdsys05*

* Created under the %PDDIR%\area directory.

(2) Names of user-created files

Table 22-9 lists the names of the user-created files that are used with the sample database.

Table 22-9: Names of the user-created files created by SPsetup.bat

	File type	File name	Remarks
HiRDB system definition files	System common definition	%PDDIR%\conf\pdsys	The directories are created during installation.
	Unit control information definition file	%PDDIR%\conf\pduetsys	
	Single server definition file	%PDDIR%\conf\sds01	
System files	System log files	rdsys011\log1* rdsys012\log2* rdsys013\log3* rdsys014\log4* rdsys015\log5* rdsys016\log6*	6 groups
	Synchronization point dump files	rdsys014\spd1* rdsys015\spd2* rdsys016\spd3*	3 groups
	Unit status files	rdsys011\utsts1a* rdsys012\utsts1b* rdsys013\utsts2a* rdsys014\utsts2b* rdsys015\utsts3a* rdsys016\utsts3b*	2 per server × 3
	Server status files	rdsys011\sts1a* rdsys012\sts1b* rdsys013\sts2a* rdsys014\sts2b* rdsys015\sts3a* rdsys016\sts3b*	2 per server × 3
System RDAREAs	Master directory RDAREA	rdsys02\rdmast*	RDAREA name: RDMAST
	Data directory RDAREA	rdsys02\rd dirt*	RDAREA name: RDDIRT

22. Sample Files

File type		File name	Remarks
	Data dictionary RDAREA	rdsys02\rdict*	RDAREA name: RDDICT
	Data dictionary LOB RDAREA (for storing source)	rdsys02\rtm_src*	RDAREA name: DIC_RTN_SRC
	Data dictionary LOB RDAREA (for storing objects)	rdsys02\rtm_obj*	RDAREA name: DIC_RTN_OBJ
Work table files		rdsys03*	—
RPC trace files		%PDDIR%\spool\pdrpctr	—
User RDAREAs (for storing data)		rdsys04\rddata10*	RDAREA name: RDDATA10
User RDAREAs (for storing indexes)		rdsys04\rdindx10*	RDAREA name: RDINDX10
User LOB RDAREAs		rdsys05\rlob1*	RDAREA name: RLOB1
		rdsys05\rlob2*	RDAREA name: RLOB2

Legend:

— : Not applicable

* Created under the %PDDIR%\area directory.

Chapter

23. 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.

- 23.1 Connecting to a HiRDB server with an FQDN specified
- 23.2 Settings for a DNS server to manage IP addresses
- 23.3 Settings when a firewall and NAT are installed
- 23.4 Port numbers used by HiRDB
- 23.5 HiRDB reserved port facility

23.1 Connecting to a HiRDB server with an FQDN specified

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, connection may not be established using the host name specified in the `pdunit` operand. In an environment using DNS, see *23.1.1 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 *23.1.2 Using the multi-connection address facility to connect to a HiRDB server*.

23.1.1 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`.

Table 23-1 lists the names that can be specified in the client environment definition.

Table 23-1: 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.

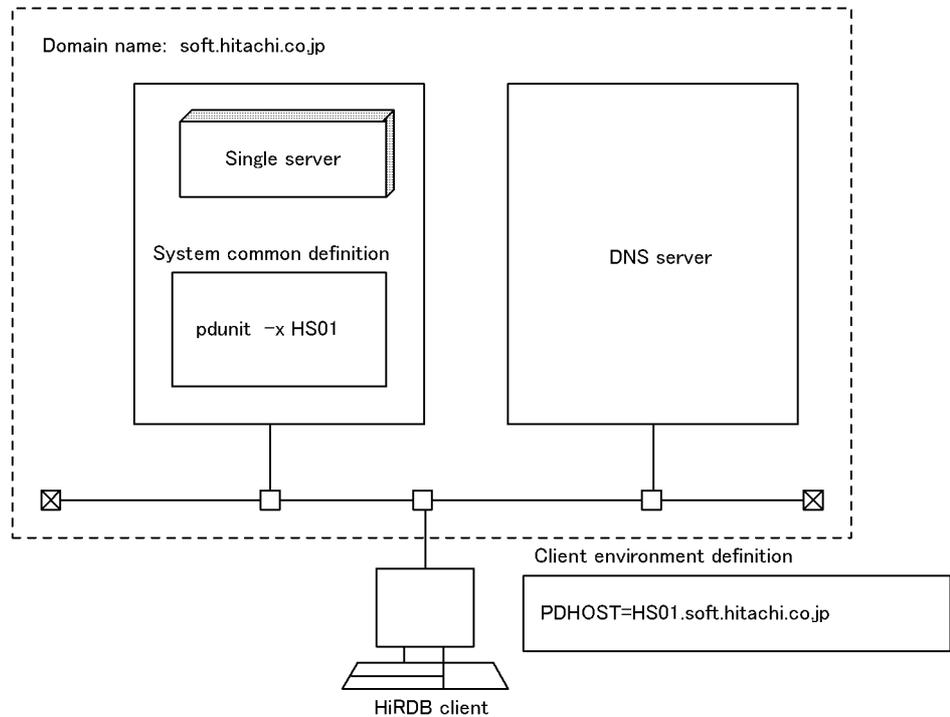
* This is used in a large-scale network environment to avoid having to modify the host's file when a host name or IP address is registered or when an IP address is

changed.

(1) Example of network configuration and definition for connecting to a HiRDB server with an FQDN specified

Figure 23-1 provides an example of a network configuration and definition for connecting to a HiRDB server with a fully qualified domain name (FQDN) specified.

Figure 23-1: 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 may 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 *23.1.2 Using the multi-connection address facility to connect to a HiRDB server*.

23.1.2 Using the multi-connection address facility to connect to a HiRDB server

It may 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/PDFESHOT` 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. To also use the system switchover facility, the `-n` option must be specified in addition to the `-m` option.

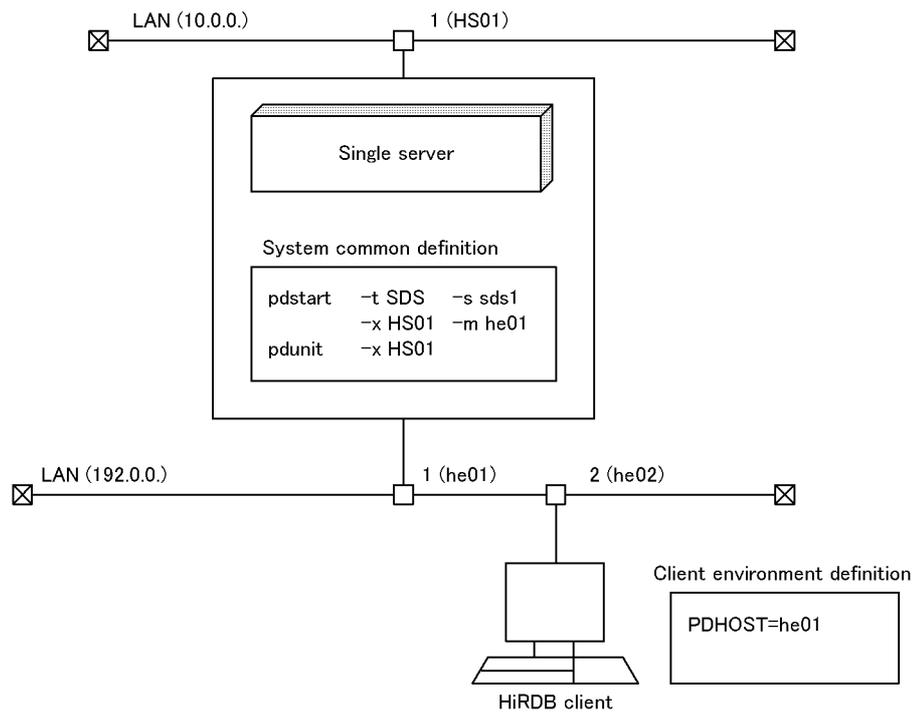
The HiRDB client specifies in the `-m` and `-n` options the host name of the HiRDB system to which connection can be established in the network; this does not have to be the host name specified in the `pdunit` operand.

(2) Examples of network configurations and definitions using the multi-connection address facility

(a) HiRDB/Single Server

Figure 23-2 provides an example of a network configuration and definition when the multi-connection address facility is used with a HiRDB/Single Server.

Figure 23-2: Example of network configuration and definition: HiRDB/Single Server



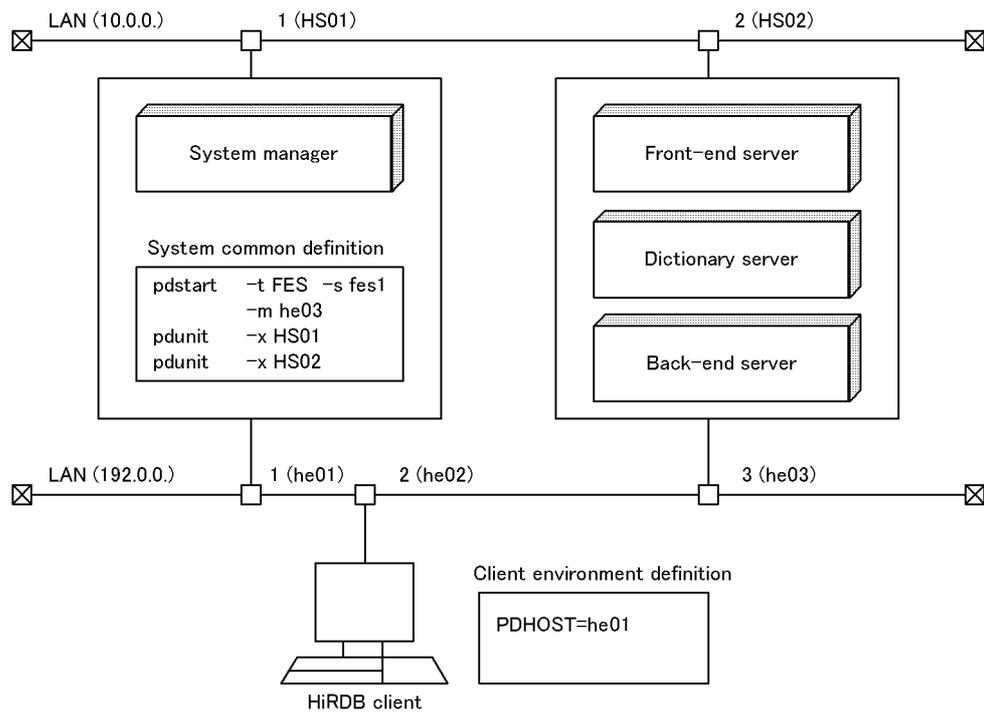
Explanation:

- 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 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 is specified in the `PDHOST` operand in the client environment definition.

(b) HiRDB/Parallel Server

Figure 23-3 provides an example of a network configuration and definition when the multi-connection address facility is used with a HiRDB/Parallel Server.

Figure 23-3: Example of network configuration and definition: HiRDB/Parallel Server



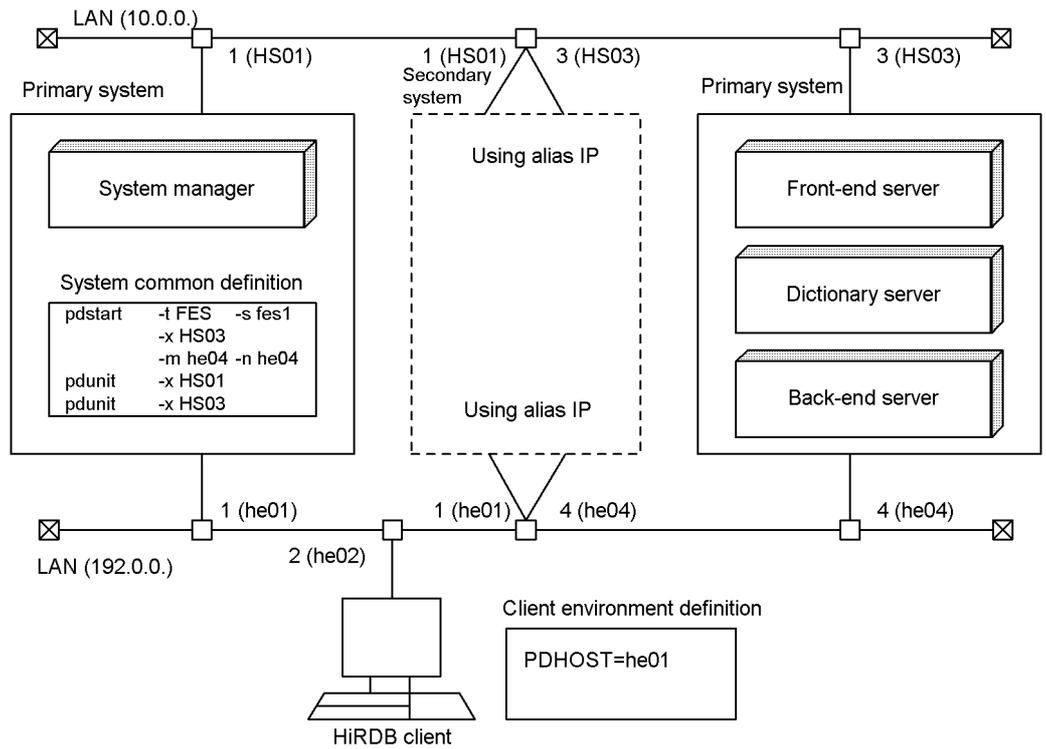
Explanation:

- 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 (with inheritance of IP addresses during system switchover)

Figure 23-4 provides an example of a network configuration and definition when the multi-connection address facility is used and IP addresses are inherited during system switchover.

Figure 23-4: Example of network configuration and definition: With inheritance of IP addresses during system switchover



Explanation:

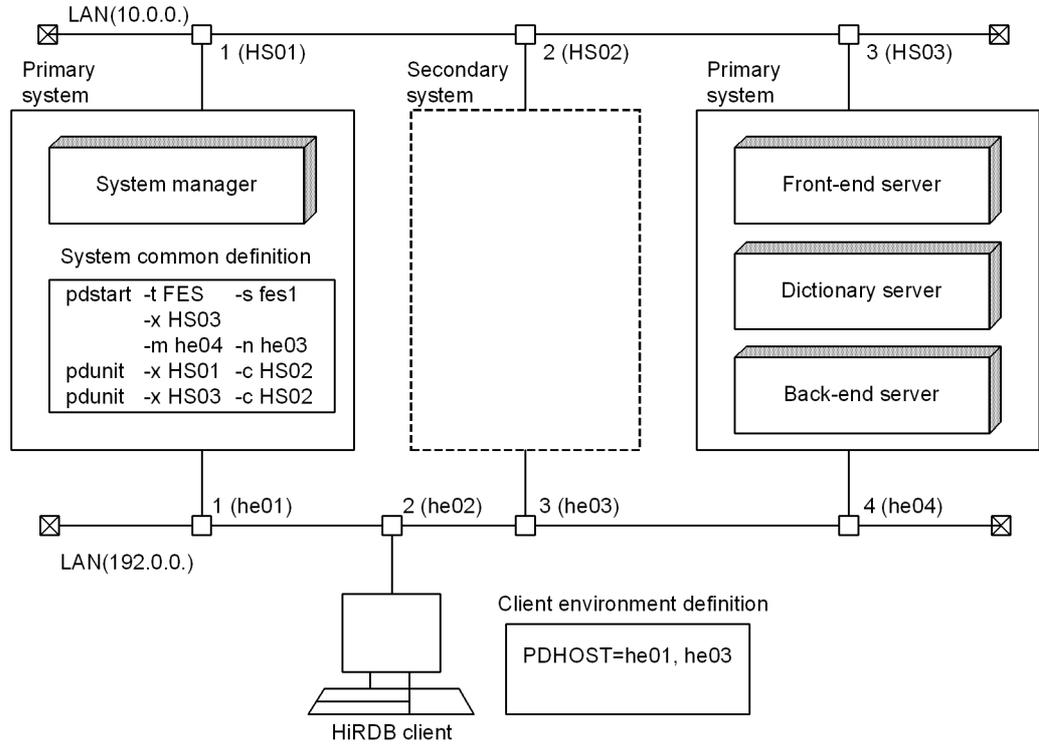
- The host names (HS01 and HS03) used for the network communications between the HiRDB systems are specified in the -x option of the pdunit operand.
- The host name (he04) in the network that is used between the HiRDB client and the HiRDB system is specified in the -m option of the pdstart operand (for defining the front-end server). The host name of the secondary system (he04) 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.

(d) HiRDB/Parallel Server (without inheritance of IP addresses during system switchover)

Figure 23-5 provides an example of a network configuration and definition when the

multi-connection address facility is used and IP addresses are not inherited during system switchover.

Figure 23-5: 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.

23.2 Settings for a DNS server to manage IP addresses

There are two ways for an 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

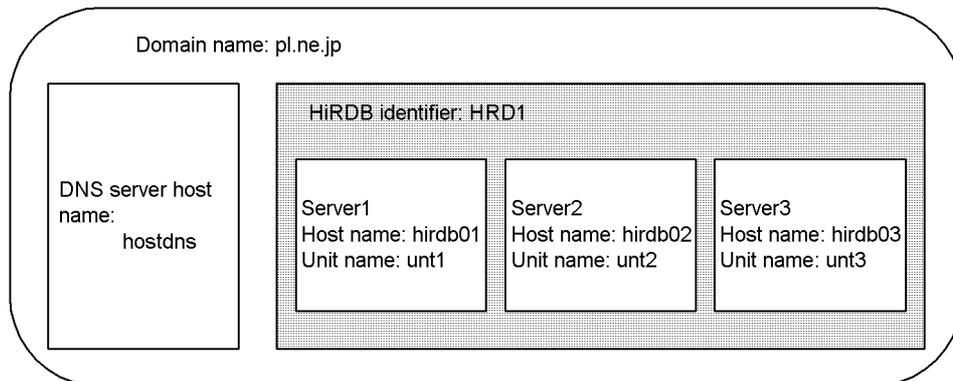
If the server machines reside in the same domain, specify either the host name or the FQDN (fully-qualified domain name; maximum of 32 characters) as the host name in the `pdunit` and `pdstart` operands. In this way, a DNS server can manage IP addresses, thereby making the `hosts` file unnecessary.

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

Figure 23-6 shows an example system configuration using a single domain.

Figure 23-6: 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

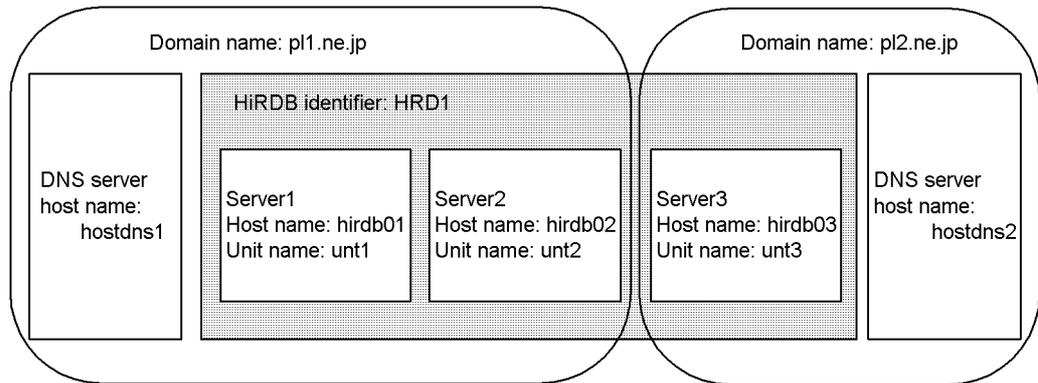
If the server machines reside in multiple domains, specify the FQDN (fully-qualified domain name; maximum of 32 characters) as the host name in the `pdunit` and `pdstart` operands. In this way, a DNS server can manage the IP addresses, thereby making the `hosts` file unnecessary.

Basically, the FQDN is specified using the following options.

- `pdunit` operand: `-x` and `-c` options
- `pdstart` operand: `-x`, `-m`, and `-n` options

Figure 23-7 shows an example system configuration using multiple domains.

Figure 23-7: 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 . . .

23.3 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.

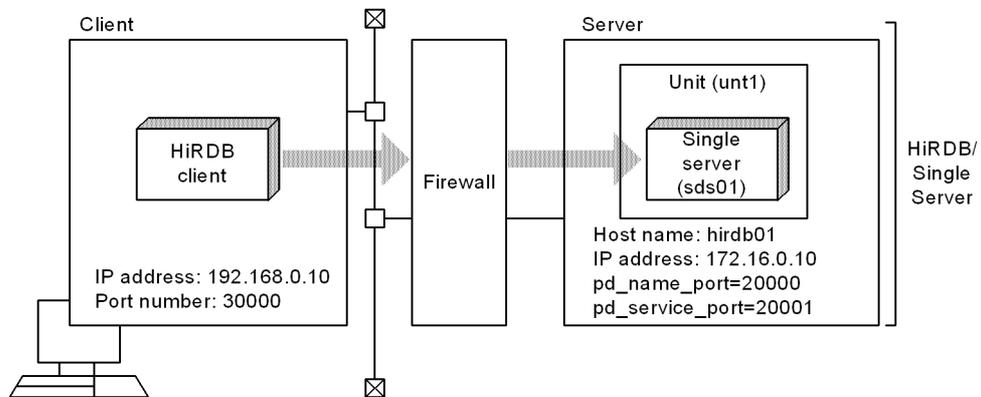
23.3.1 When a firewall is installed on the HiRDB/Single Server side

As shown in Figure 23-8, a firewall has been installed on the HiRDB/Single Server side, and it has been configured as follows:

Firewall settings

- Direction: Receive
- IP address: 172.16.0.10
- Port numbers: 20000, 20001

Figure 23-8: Network configuration example with a firewall installed on the HiRDB/Single Server side



In this configuration, the settings for the server and client machines are as listed below. If you are installing a firewall, you must specify the `pd_service_port` 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.

23.3.2 When a firewall and NAT are installed on the HiRDB/Single Server side

As shown in Figure 23-9, a firewall and NAT have been installed on the HiRDB/Single Server side, and they have been configured as follows:

Firewall settings

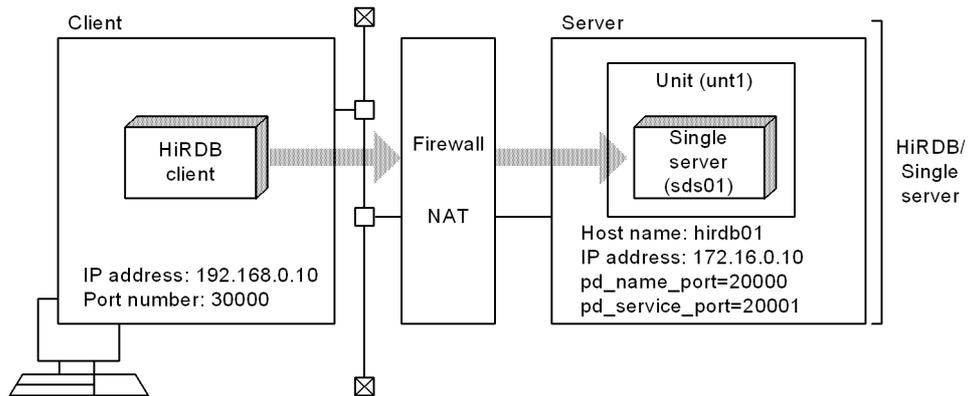
- Direction: Receive
- IP address: 172.16.0.10
- Port numbers: 20000, 20001

NAT address translation

```
128.1.1.1 ← → 172.16.0.10
```

Note that HiRDB does not support a function that converts global and local IP addresses as a pair, such as NAT (IP masquerade). It supports only 1-to-1 conversion.

Figure 23-9: Network configuration example with a firewall and NAT installed on the HiRDB/Single Server 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 PC
PDCLTRCVPORT 30000*
```
- hosts file


```
128.1.1.1 hirdb01
```

* Specify this when there is a firewall on the client side.

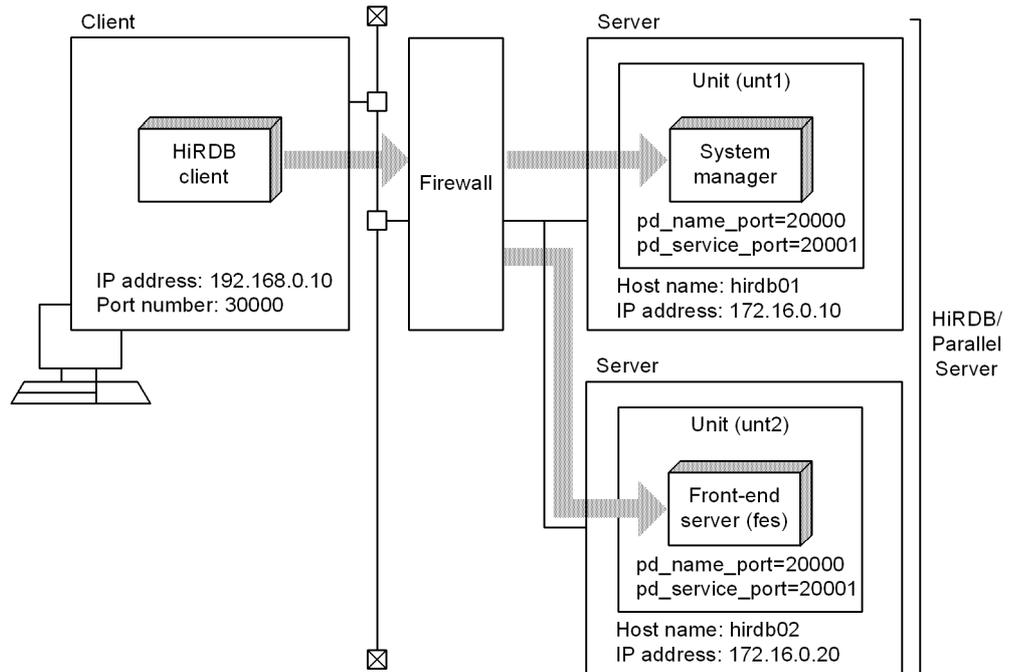
23.3.3 When a firewall is installed on the HiRDB/Parallel Server side

As shown in Figure 23-10, a firewall has been installed on the HiRDB/Parallel Server side, and it has been configured as follows:

Firewall settings

- Direction: Receive
- IP addresses: 172.16.0.10, 172.16.0.20
- Port numbers: 20000, 20001

Figure 23-10: Network configuration example with a firewall installed on the HiRDB/Parallel Server side



In this configuration, the settings for the server and client machines are as listed below. If you are installing a firewall, you must specify the `pd_service_port` 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.

23.3.4 When a firewall and NAT are installed on the HiRDB/Parallel Server side

As shown in Figure 23-11, a firewall and NAT have been installed on the HiRDB/Parallel Server side, and they have been configured as follows:

Firewall settings

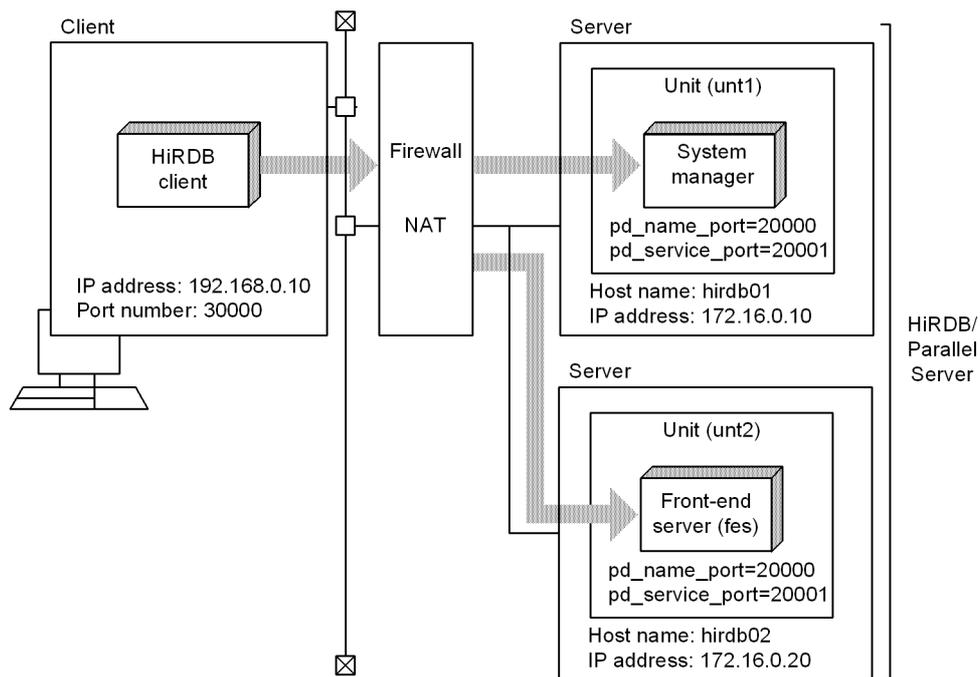
- Direction: Receive
- IP addresses: 172.16.0.10, 172.16.0.20
- Port numbers: 20000, 20001

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 NAT (IP masquerade). It supports only 1-to-1 conversion.

Figure 23-11: Network configuration example with a firewall and NAT installed on the HiRDB/Parallel Server 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
```

23. Communication Between HiRDB Servers and HiRDB Clients

```
PDNAMEPORT 20000
PDSERVICEGRP fes
PDSERVICEPORT 20001
PDFESHOST hirdb02
PDSRVTYPE PC
PDCLTRCVPORT 30000*
```

- `hosts` file

```
128.1.1.1 hirdb01
```

```
128.1.1.2 hirdb02
```

* Specify this when there is a firewall on the client side.

23.4 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 *23.5 HiRDB reserved port facility*.

23.4.1 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) Single server

Formula

$$\text{pd_max_users} \times 2 + 1000$$

(2) Parallel Server

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

$$\{b \times [k \times [(b - 1) \div 2 + B + 1] + 1] \\ + f \times (k \times b + D + 2) + d \times 2\} \\ \times \text{pd_max_users} + 1000$$

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 : $2 \div 3$

23.4.2 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.
- If the system has a large maximum number of concurrent connections (`pd_max_users`), specify 1 in the `PDTCPCONOPT` client environment definition in order to reduce the number of ports that UAPs use to connect to HiRDB. For details about the `PDTCPCONOPT` client environment definition, see the manual *HiRDB Version 8 UAP Development Guide*.

23.4.3 Calculation examples

- A 1-unit configuration of a HiRDB/Parallel Server (with a FES, a DS, and 5 BESs), where `pd_max_users=1000`

$$\begin{aligned} & \{ 5 \times [2 \div 3 \times (0 + 0) + 1] \\ & + 1 \times (2 \div 3 \times 0 + 0 + 2) + 1 \times (0 + 1) \} \\ & \times 1000 + 1000 = 9000 \end{aligned}$$

The target number of ports is 9000.

- A parallel unit configuration (unit 1: a FES, a DS, and 2BESs; unit 2: 3 BESs), where `pd_max_users=900`

Unit 1:

$$\begin{aligned} & \{ 2 \times [2 \div 3 \times (3 + 0) + 1] \\ & + 1 \times (2 \div 3 \times 3 + 0 + 2) + 1 \times (0 + 1) \} \\ & \times 900 + 1000 = 10900 \end{aligned}$$

Unit 2:

$$\begin{aligned} & \{ 3 \times [2 \div 3 \times (2 + 1) + 1] \\ & + 0 \times (2 \div 3 \times 2 + 1 + 2) + 0 \times (1 + 1) \} \\ & \times 900 + 1000 = 9100 \end{aligned}$$

The target number of ports is 10,900 for unit 1 and 9,100 for unit 2.

23.5 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.

Consider using the HiRDB reserved port facility in the following cases:

- When `CONNECT` or `DISCONNECT` from HiRDB clients occur several thousand times a minute
- When transactions occur several thousand times a minute on a HiRDB/Parallel Server

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 8 System Definition*.

23.5.1 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 8 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 *23.4.1 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
- B. Questions and Answers
- C. Setting Up an Environment Using a Batch File

A. HiRDB Maximum and Minimum Values

A.1 Maximum and minimum values for the system configuration

Table A-1 lists the 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	41	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 per RDAREA	0	500	Count
Number of indexes per RDAREA	0	500	Count
Number of lists per RDAREA	0	50,000	Count

Item	Minimum value	Maximum value	Units
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	1	2000	Count
Maximum number of users that can be created by HiRDB ²	1	Unlimited	Count
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	1	1,048,575	MB
Windows file size	0	263 ¹	Bytes

¹ For a configuration with multiple front-end servers, the maximum for number of front-end servers \times value of `pd_max_users` is 2,000.

2

- When the Directory Server linkage facility is not used
The value depends on the size of data dictionary RDAREAs because one row of the data dictionary table (`SQL_USERS`) is used per user.
- When the Directory Server linkage facility is used
For a user with `CONNECT` privilege only, the value depends on the directory server specifications. For a user with other privileges, the value is same as when the Directory Server linkage facility is not used.

³ However, for the entire system, the maximum is 2,147,483,647.

A.2 Maximum and minimum values for databases

Table A-2 lists the 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
Length of mixed character data (defined length)	MCHAR	1	30,000	Bytes
	MVARCHAR	1	32,000	Bytes
Precision of packed decimal	DECIMAL	1	29	Digits
Decimal places for packed decimal	DECIMAL	0	29	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	16	Columns
Number of cluster key component columns		1	16	Columns
Number of RDAREAs for storing table partitions		1	1024	Count
Number of BESS where partitioned tables are placed		1	1024	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	16	Columns
Number of foreign key component columns		1	16	Columns
Number of foreign keys per table		0	255	Count

Item	Minimum value	Maximum value	Units
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, group ID, query name, trigger identifier, statement label, loop variable name, host identifier, list name, roll 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	—	— ¹	—
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 whether or not plug-ins are used and the number of RDAREAs in the server. 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

Table A-3 lists the 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. Questions and Answers

This appendix discusses in question-and-answer format some of the topics concerning HiRDB system construction that can be easily misunderstood. The topics that are discussed are listed as follows:

Topic	Refer to
Installation directory	(1)
Error 1073	(2)
Method for determining the size of virtual memory	(3)
Use of a network drive	(4)
Host name specified in the <code>pdstart</code> operand of the system common definitions	(5)
HiRDB/Developer's Kit	(6)
Execution of the database definition utility (<code>pddef</code>)	(7)
Maximum size of a database	(8)
OpenTP1 and XA interface	(9)
Performance of <code>FIX</code> tables	(10)
Duplicate key index	(11)
Indexes	(12)
Index definition for a row-partitioned table	(13)
Handling of synchronization point dumps	(14)
Handling of status files	(15)-(18)
Maximum size of HiRDB file system area for work tables	(19)
HiRDB cannot be started by the <code>pdstart</code> command	(20)
Any particular unit cannot be started by the <code>pdstart</code> command	(21)
HiRDB startup using the <code>pdstart</code> command is slow	(22)
<code>pdstart</code> command results in an error (reason code= <code>TIMEOUT</code>)	(23)
Database definition utility (<code>pddef</code>) results in an error	(24)
<code>CREATE TABLE</code> statement LOB column definition	(25)

(1) Installation directory

Question

Are there any points to be noted about the installation directory?

Answer

When specifying the name of the installation directory, note the following points. The pathname of the installation directory is set in the `PDDIR` environment variable.

- Specify a character string that begins with the drive letter and consists of the following characters only:
 - Alphanumeric characters
 - _ (underscore)
 - . (period)
 - \ (backslash for separating paths)
- Do not specify the drive letter only.
- Do not use any double-byte character, or special symbol.
- The length must be no greater than 128 bytes.
- Use the name of the HiRDB directory that is specified in the `pdunit` system common definition operand.

(2) Error 1073

Question

Can error 1073 occur during installation?

Answer

Error 1073 occurs if HiRDB is uninstalled and then installed again without restarting the operating system. You need to restart the operating system after installing or uninstalling HiRDB. For details about how to handle errors, see 2.2.5 *What to do if error 1073 occurs during installation.*

(3) Method for determining the size of virtual memory

Question

How can I determine the size of virtual memory?

Answer

In general, the size of virtual memory is about two to three times the size of real memory. Because virtual memory uses a disk, a shortage of disk space may occur if the virtual memory is too large. The operating system controls whether to use

real memory or virtual memory; HiRDB cannot control it.

(4) Use of a network drive

Question

Can I use a network drive?

Answer

No. Use local drives for backing up utilities as well as installing HiRDB.

(5) Host name specified in the pdstart operand of the system common definitions

Question

Where is the following host name defined: the host name that is specified in the `pdstart` operand of the system common definitions?

Answer

In the **Control Panel**, double-click the **Network** icon to display the TCP/IP properties. On the **DNS** tab, the host name is defined. You can also check this setting by executing the `hostname` command at the command prompt.

(6) 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.

(7) 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 blanks after the semicolon (;) in a `pddef` control statement. If there is a blank, that SQL statement will not execute.

Wrong: `CREATE TABLE ; Δ` [Δ : blank]

Correct: `CREATE TABLE ;`

(8) Maximum size of a database

Question

What is the maximum size of a HiRDB database?

Answer

One table can be divided and stored in a maximum of 512 RDAREAs, and one RDAREA can consist of a maximum of 16 HiRDB files.

(9) 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 may not be apparent because fewer steps are required than in the case of an updating SQL.

(10) 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
- No columns will be added

(11) 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.

(12) How to use indexes**Question**

If an index is defined for a table in a configuration in which HiRDB is accessed from DBPARTNER2, does DBPARTNER2 need to know about the index?

Answer

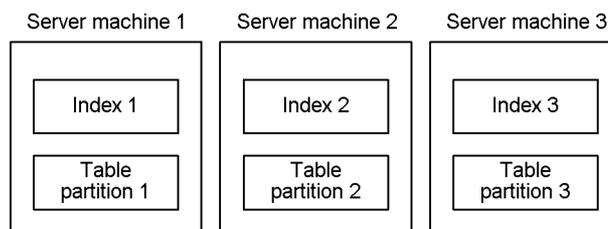
No. HiRDB handles DBPARTNER2 in the same manner as regular client UAPs.

(13) 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.

**(14) Handling of synchronization point dumps****Question**

How many guaranteed-valid generations of synchronization point dump files should be provided?

Answer

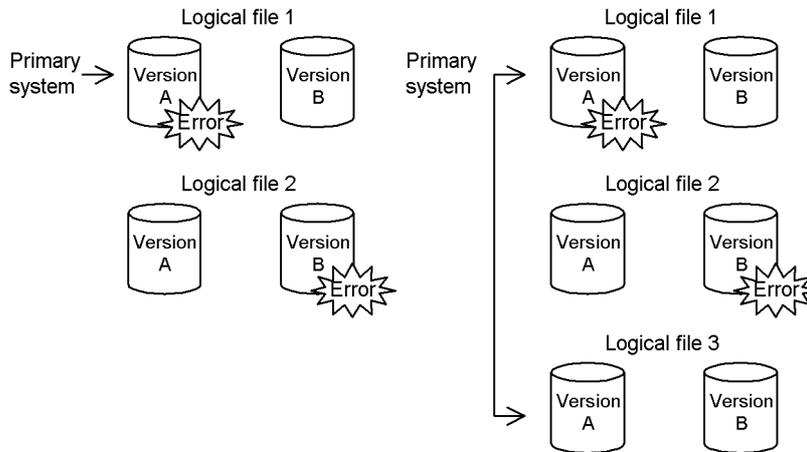
Information such as the read operation start location in the system log file is acquired in the synchronization point dump file each time a synchronization point dump is collected in order to prepare for a full rerun. The portion of the system log file beginning at the location indicated in the synchronization 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.

(15) 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`.

(16) 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 HiRDB/Parallel Server) 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).

(17) 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 (dual files \times 3 = 6 files).

If there is not enough room on the disk, two logical status files are reasonable (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.

(18) Handling of status files (status file placement)

Question

How should status files be placed on disks?

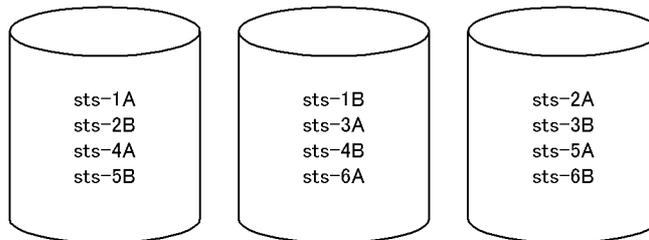
Answer

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.

(19) Peak capacity of a HiRDB file system area for work table files

Question

How can I determine the peak capacity of a HiRDB file system area for work table files (back-end server definition: pdwork)?

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 for the allocated HiRDB file system area. The peak capacity value that is output is the maximum utilization value.

The maximum utilization value can be cleared with the `pdfstatfs -c` command:

```
pdfstatfs -c name-of-HiRDB-file-system-area-for-work-table-files
```

The `-c` option sets the maximum utilization value for the allocated HiRDB file system area to 0.

Notes

The `-d` and `-c` options of the `pdfstatfs` command are applicable only when the usage of the HiRDB file system area is `WORK` or `UTL`. If the usage is `SYS` or `DB`, this information is not displayed.

The `-k` option of the `pdfmkfs` command is used to specify the usage of a HiRDB file system area.

(20) 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

A possible cause is:

- HiRDB has not been set up correctly.

Set up HiRDB correctly.

(21) 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?

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`.

(22) 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.

(23) 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 messages output to the event log and correct the erroneous definition.

(24) 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 may 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 of the `hirdb.ini` file.

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 8 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.

(25) 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

Whether or not 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.

C. Setting Up an Environment Using a Batch File

C.1 Overview of environment setup using a batch file

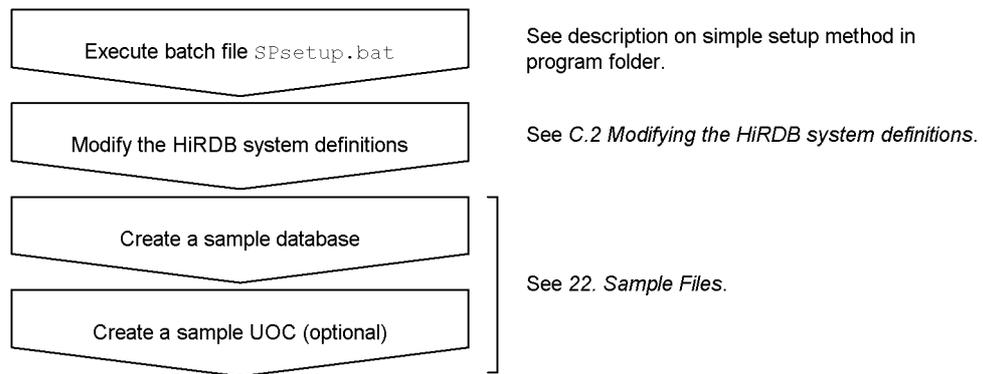
This section describes the environment setup method using a batch file.

Notes about this chapter

For details about how to set up an environment using a batch file, see the description about the simple setup method in the HiRDB/Single Server or HiRDB/Parallel Server program folder. In this chapter, the description of a procedure is given assuming that the HiRDB/Single Server has been installed in the default installation directory (C:\win32app\hitachi\hirdb_s)

Figure C-1 shows the environment setup procedure using a batch file.

Figure C-1: Environment setup procedure using a batch file



(1) Contents of the batch file (SPsetup.bat)

HiRDB provides the SPsetup.bat batch file (C:\win32app\hitachi\hirdb_s\sample\SPsetup.bat), which automatically sets up a HiRDB environment. SPsetup.bat automatically executes the following operations:

- Creation of HiRDB system definition files
- Creation of HiRDB file system areas
- Creation of system files
- Creation of RDAREAs

SPsetup.bat uses the files listed in Table C-1 to automatically create an environment. These files are called *sample configuration*. You can construct a system suitable for your applications by customizing the contents of these files. These files are

located in the C:\win32app\hitachi\hirdb_s\conf directory.

Table C-1: Contents of sample configuration

Filename	Description
pdsys pdutsys sds01	HiRDB system definition files <ul style="list-style-type: none"> • pdsys: System common definitions • pdutsys: Unit control information definitions • sds01: Single server definitions
fmkfile.bat	Batch file for creating the following HiRDB file system areas: <ul style="list-style-type: none"> • HiRDB file system area for RDAREAs • HiRDB file system area for work table files
fmkfs.bat	Batch file for creating the following HiRDB file system area: <ul style="list-style-type: none"> • HiRDB file system area for system files
sysfint.bat	Batch file for creating system files
initdb.bat	Batch file for creating RDAREAs. It executes the database initialization utility (pdinit).
Mkinit	Batch file containing the control statements of the database initialization utility (pdinit).

(2) Configuration of the system to be constructed

Figures C-2 and C-3 show the sample system configuration that is created when SPsetup.bat is executed. If not customized, the batch file creates this system configuration.

Figure C-2: Sample system configuration created by SPsetup.bat

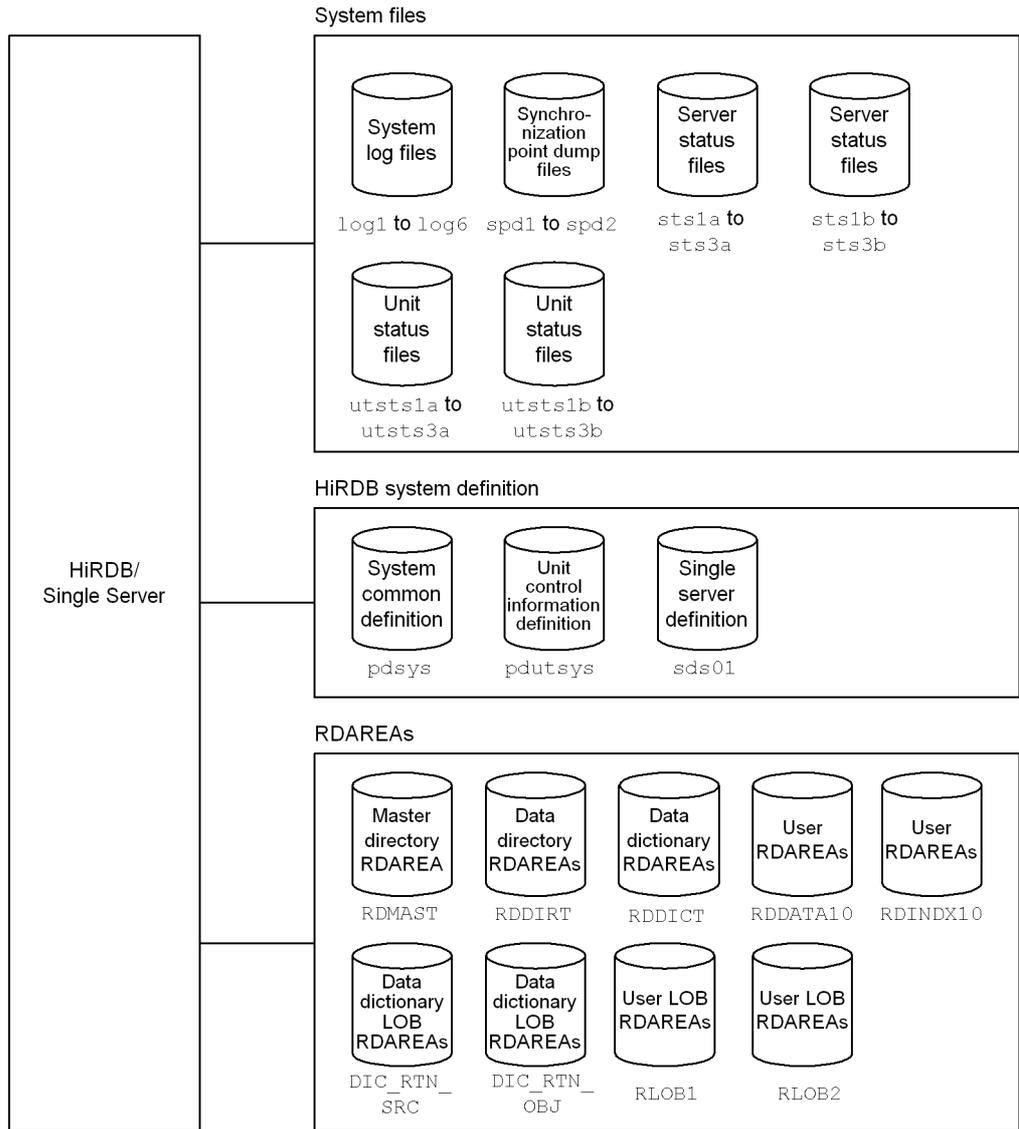
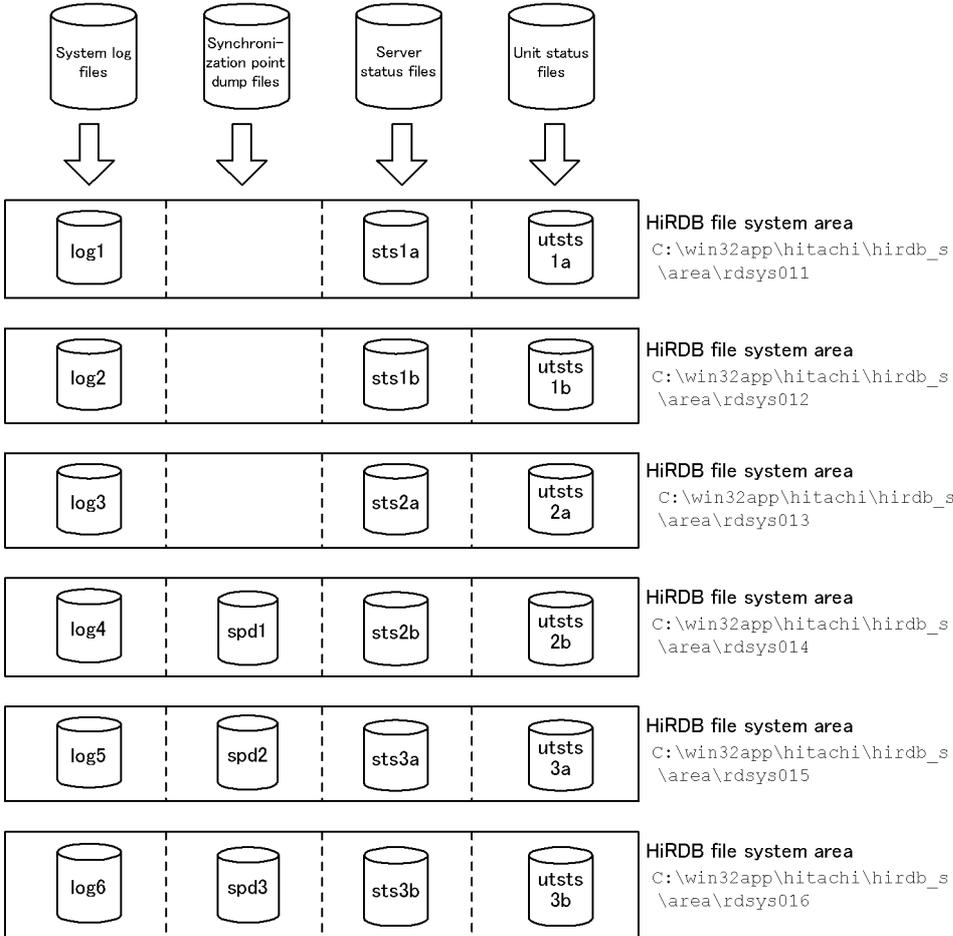


Figure C-3: Organization of sample HiRDB file system areas created by SPsetup.bat

- HiRDB file system areas for system files

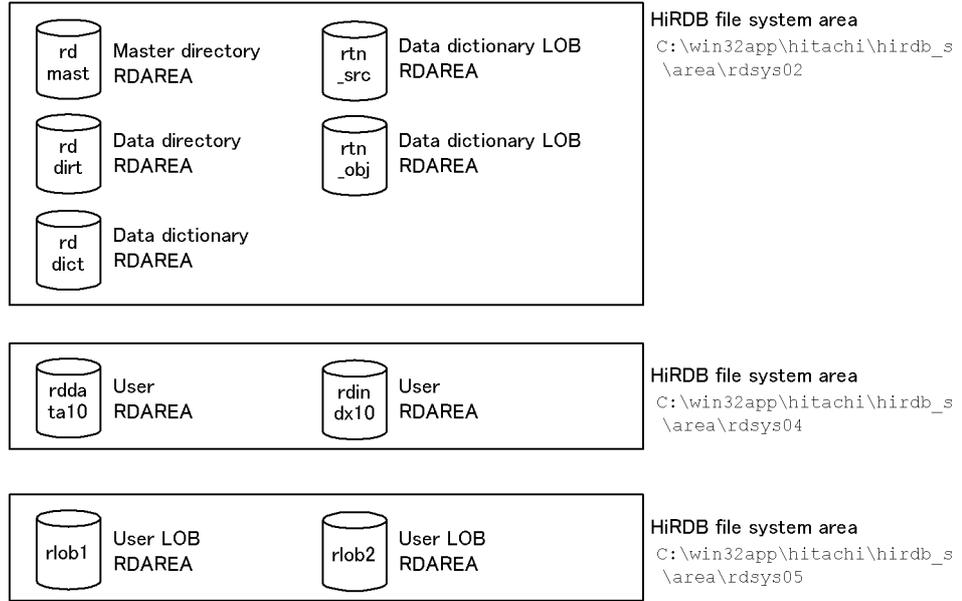


- HiRDB file system area for work table files



C. Setting Up an Environment Using a Batch File

- HiRDB file system areas for RDAREAs



(3) HiRDB file system areas and files to be created

Tables C-2 and C-3 list the HiRDB file system areas and files that are created by SPsetup.bat. If not customized, the batch file creates these HiRDB file system areas and files.

Table C-2: Names and sizes of HiRDB file system areas created by SPsetup.bat

Type of HiRDB file system area	Size of HiRDB file system area (MB)	HiRDB file system area name
For system files	296	rdsys011 ¹ rdsys012 ¹ rdsys013 ¹ rdsys014 ¹ rdsys015 ¹ rdsys016 ¹
For RDAREAs (system RDAREAs)	80	rdsys02 ²
For work table files	—	rdsys03 ²
For RDAREA (user RDAREAs)	160	rdsys04 ²

Type of HiRDB file system area	Size of HiRDB file system area (MB)	HiRDB file system area name
For RDAREAs (user LOB RDAREAs)	40	rdsys05 ²

¹ Created in the %PDDIR%\area directory.

² Created in the %PDDIR%\area directory.

Table C-3: Names of files created by SPsetup.bat

File type		Filename	Remarks
HiRDB system definition files	System common definition file	%PDDIR%\conf\pdsys	The directory is created during installation.
	Unit control information definition file	%PDDIR%\conf\pdutsys	
	Single server definition file	%PDDIR%\conf\sds01	
System files	System log files	rdsys011\log1 ¹ rdsys012\log2 ¹ rdsys013\log3 ¹ rdsys014\log4 ¹ rdsys015\log5 ¹ rdsys016\log6 ¹	6 groups
	Synchronization point dump files	rdsys014\spd1 ¹ rdsys015\spd2 ¹ rdsys016\spd3 ¹	3 groups
	Unit status files	rdsys011\utsts1a ¹ rdsys012\utsts1b ¹ rdsys013\utsts2a ¹ rdsys014\utsts2b ¹ rdsys015\utsts3a ¹ rdsys016\utsts3b ¹	Dual files × 3 per unit
	Server status files	rdsys011\sts1a ¹ rdsys012\sts1b ¹ rdsys013\sts2a ¹ rdsys014\sts2b ¹ rdsys015\sts3a ¹ rdsys016\sts3b ¹	Dual files × 3 per server

File type		Filename	Remarks
System RDAREAs	Master directory RDAREA	rdsys02\rdmast ²	RDAREA name: RDMAST
	Data directory RDAREA	rdsys02\rdmdir ²	RDAREA name: RDDIRT
	Data dictionary RDAREA	rdsys02\rddict ²	RDAREA name: RDDICT
	Data dictionary LOB RDAREA (for storing source)	rdsys02\rtm_src ²	RDAREA name: DIC_RTN_SRC
	Data dictionary LOB RDAREA (for storing object)	rdsys02\rtm_obj ²	RDAREA name: DIC_RTN_OBJ
Work table file		rdsys03 ²	—
RPC trace file		%PDDIR%\spool\pdrpctr	—
User RDAREA (for storing data)		rdsys04\rddata10 ²	RDAREA name: RDDATA10
User RDAREA (for storing indexes)		rdsys04\rdindx10 ²	RDAREA name: RDINDX10
User LOB RDAREA		rdsys05\rlob1 ²	RDAREA name: RLOB1
		rdsys05\rlob2 ²	RDAREA name: RLOB2

¹ Created in the %PDDIR%\area directory.

² Created in the %PDDIR%\area directory.

C.2 Modifying the HiRDB system definitions

Executor: HiRDB administrator

The HiRDB system definition is created when you perform the procedure described in *C.1 Overview of environment setup using a batch file*. Modify the HiRDB system definitions as required. Before modifying the HiRDB system definitions, be sure to terminate HiRDB normally.

The following are notes about modifying (creating) the HiRDB system definitions:

Notes

1. Specify and maintain the permissions for the HiRDB system definition file so that

only the owner of the file (the HiRDB administrator) has read and write permissions.

2. If you are specifying the `pdunit` operand in the system common definitions, be sure to specify the same value as the `PDDIR` environment variable for the HiRDB directory. This information is case sensitive.
3. Make sure that the host name specified in `-x` of the `pdstart` operand in the system common definition is the same as the host name specified in `-x` of the `pdunit` operand. You can obtain this host name by executing `hostname` at the command prompt.
4. Make sure that the value of the `TZ` operand (time zone) in the system common definitions is the same as the value of the `TZ` system environment variable in the server machine. You can display the server machine's environment variable by entering the `set` command at the command prompt. If the `TZ` system environment variable is not specified or the specified value is different from the `TZ` operand value in the system common definitions, HiRDB operation cannot be guaranteed. To specify the system environment variable, on the **Control Panel**, double-click **System**, then from the **System Properties** dialog box, choose the **Environment** tab.
5. Do not assign a file extension such as `.txt` to the HiRDB system definition files.

Index

A

abstract data type 385
 defining 151
 null value for 153
abstract data type column structure base table 155
Administrators privilege 42
AFTER trigger 344
alias (command) 822
ALTER TRIGGER 345
ARRAY option (CREATE TABLE) 383
asynchronous READ facility 285
asynchronous XA call 176
automatic opening
 HiRDB/Parallel Server 249, 253
 HiRDB/Single Server 215, 219

B

back-end server definition, creating 64
back-end server, to be linked with foreign server 69
base row log information, determining amount of 707
basic attribute (JP1 linkage) 199
batch file
 for executing commands under aliases,
 creating 822
 overview of environment setup using 868
batch index creation, handling errors during 161
BEFORE trigger 344
BINARY type 366
BLOB type 366
boundary value, specifying 315
branch row log information, determining amount
of 711
buffer hit rate 281

C

candidate key 360
character codes, selection of 47
check constraint 434

 table 434
check pending status 412, 436
Chinese EUC encoding 47
client environment definition
 registering, in transaction manager 189
 setting (database creation) 130
 specifying (multi-HiRDB) 270
close character string 188
cluster key
 specifying 361
CLUSTER KEY option (CREATE TABLE) 362
column information work table
 obtaining maximum number of 741
 obtaining size of 738
command
 environment setup using 57
 overview of environment setup using 58
commit-time reflection processing 289
communication port's service name
 changing (multi-HiRDB) 269
 specifying (multi-HiRDB) 268
constructor function 152, 389
CREATE INDEX 447
create rdarea statement 96, 101, 497
CREATE SCHEMA 130
CREATE TRIGGER 344
CREATE TYPE 151

D

data
 in table, storing 157
 loading 821
data conversion facility, specifying (database
creation) 131
data dictionary LOB RDAREA
 creating 109
 determining size of 684
 page length of 684, 685
 placing (HiRDB/Parallel Server) 260

- placing (HiRDB/Single Server) 226
 - total number of pages in 684, 685
 - data dictionary RDAREA
 - determining size of 646
 - determining size of (for storing database state analyzed tables and database management tables) 679
 - page length of 646
 - total number of pages in 646
 - data directory RDAREA
 - determining size of 683
 - page length of 683
 - total number of pages in 683
 - data global buffer 278
 - allocating 275
 - data integrity
 - check constraint 438
 - referential constraint 420
 - data length
 - for abstract data type 628
 - for repetition columns 629
 - for variable-length character string type 627
 - list of 622
 - of abstract data type column, obtaining 627
 - of repetition columns, obtaining 629
 - data loading
 - with synchronization point specification 136
 - with synchronization point specification (handling utility abnormal termination errors) 167
 - data local buffer 297
 - allocating 298
 - data manipulation
 - check constraint 438
 - referential constraint 420
 - data page 283
 - data storage status, checking 150, 160
 - data types for which indexes canifot be defined 448
 - database
 - creating 129
 - maximum size of 860
 - maximum value for 853
 - minimum value for 853
 - database definition utility
 - execution of (pddef) 859
 - results in error (pddef) 866
 - database initialization utility 96
 - database load utility 132
 - database structure modification utility 101
 - database update log acquisition method 158
 - database update log acquisition mode 132
 - types of 132
 - DBPARTNER2, HiRDB accessed from 861
 - deadlock between referenced and referencing tables 406
 - DECIMAL signed normalized number, facility for conversion to 131
 - default constructor function 153, 389
 - deferred write processing 286
 - facility for parallel writes in 288
 - deferred write trigger 286
 - updated output page rate for 286
 - deletion prevented duration 375
 - desktop heap setting, estimating 808
 - dictionary server definition, creating 64
 - directory
 - created by HiRDB 7
 - created by HiRDB administrator 7
 - Directory Server linkage facility 4
 - disk size, checking (installation) 37
 - distributed transaction processing 174
 - divided-input data file 137
 - creating 137
 - DROP TRIGGER 345
 - DTP 174
 - duplicate key index 860
 - dynamic registration 184
 - dynamic transaction registration 176
 - dynamic updating of global buffers 274
- E**
- encapsulation level 152, 389
 - environment
 - overview of setting up, using command 58
 - overview of setup, using batch file 868
 - setting up, using command 57
 - environment variable

- determining, related to number of resources 801
- installation 44
- number of resources (HiRDB/Parallel Server) 804
- number of resources (HiRDB/Single Server) 802
- setting (multi-HiRDB) 270
- environment variable specification batch file, how to use (multi-HiRDB) 271
- error 1073 858
 - during installation 45
- error occurs while updating to HiRDB update version, operation when 31
- event
 - managing by JP1/IM 201
 - monitoring by JP1/IM (overview) 202
- event attribute definition file 202
- event notice, sending (JP1 linkage) 199
- EXCEPT VALUES option (CREATE INDEX) 456
- exceptional key 455
- extended attribute (JP1 linkage) 199

F

- failed page searches by free space reuse facility, number of 495
- falsification prevented table 374
- falsification prevention facility 374
- FES host direct connection facility 235
- file
 - created by HiRDB 7
 - created by HiRDB administrator 7
- file size
 - required for execution of access path display utility (pdvwopt) 769
 - required for execution of database condition analysis utility (pddbst) 763
 - required for execution of database copy utility (pdcopy) 764
 - required for execution of database load utility (pdload) 752
 - required for execution of database reorganization utility (pdroorg) 754

- required for execution of dictionary import/export utility (pdexp) 768
- required for execution of integrity check utility (pdconstck) 772
- required for execution of optimizing information collection utility (pdgetcst) 769
- required for execution of rebalancing utility (pdrbal) 770
- required for execution of statistics analysis utility (pdstedit) 761
- required for utility execution, determining 752
- first dimension partitioning column 336
- FIX attribute, specifying 359
- FIX hash partitioning 315
 - examples of 324
- FIX table, performance of 860
- flexible hash partitioning 315
 - examples of 324
- floating machine 333
- floating server 333
 - placement of 230
- FOREIGN KEY 400
- foreign key 399
- foreign server information definition, creating 66
- free page reuse mode 488
- free space required for upgrading 13
- free space reuse facility 488
- front-end server definition, creating 63

G

- global buffer
 - allocating 274, 278
 - designing 273
 - LRU management of 290
 - pre-writing of 295
 - setting number of sectors of 281
- global buffer definition, example 279
- global buffer pool, notes about allocating to list RDAREA 277
- global buffer residence utility 296
- guaranteed-valid generations
 - number of (HiRDB/Parallel Server) 252
 - number of (HiRDB/Single Server) 218

H

hash facility for hash row partitioning 307, 334
 hash functions, types of 316
 hash partitioning 315
 heterogeneous system configuration 264
 high-speed connection facility 235
 HiRDB
 continuous 24-hour operation 2
 directory organization of 7
 file organization of 7
 in transaction manager, registering 183
 installation procedure 42
 installing 35
 linking to other products 171
 linking with OLTP system 174
 maximum value of 852
 minimum value of 855
 notes about installation of (multi-HiRDB) 268
 notes after installation 45
 notes before installation 42
 port number used by 845
 pre-installation procedure for installing 36
 replacing existing version with new version 16
 starting, for the first time 100
 stopping service of 15
 system construction procedure 2
 terminating normally 15
 uninstallation procedure 51
 update version 22
 updating to update version of 22
 upgrading 12
 upgrading error 19
 upgrading plug-in of 18
 HiRDB administrator
 registering (installation) 40
 registering (multi-HiRDB) 268
 HiRDB Advanced Partitioning Option 336
 HiRDB client to HiRDB server, connecting 828
 HiRDB Dataextractor, linking to 173
 HiRDB Datareplicator, linking to 172
 HiRDB directory, deleting file from 47
 HiRDB environment setup 2

HiRDB event
 that can be sent 199
 that can be sent to JP1/Base 200
 HiRDB External Data Access facility 4, 197
 HiRDB file name
 maximum value for 855
 minimum value for 855
 HiRDB file system area
 creating 72
 creating, for list RDAREA 78
 creating, for RDAREA 74
 creating, for system file 75
 creating, for utility 77
 creating, for work table file 76
 creating, using raw I/O facility 73, 78
 designing (HiRDB/Parallel Server) 241
 designing (HiRDB/Single Server) 207
 designing, for list RDAREA (HiRDB/Parallel Server) 244
 designing, for list RDAREA (HiRDB/Single Server) 210
 designing, for RDAREA (HiRDB/Parallel Server) 241
 designing, for RDAREA (HiRDB/Single Server) 207
 designing, for system file (HiRDB/Parallel Server) 242
 designing, for system file (HiRDB/Single Server) 208
 designing, for utility (HiRDB/Parallel Server) 243
 designing, for utility (HiRDB/Single Server) 209
 designing, for work table file (HiRDB/Parallel Server) 242
 designing, for work table file (HiRDB/Single Server) 208
 determining size of (work table file) 737
 for work table files, peak capacity of 864
 maximum size of 72, 210, 245
 types of 72
 HiRDB reserved port facility 848
 HiRDB server
 connecting HiRDB client to 828

- using multi-connection address facility to connect to 830
 - with FQDN specified, connecting to 828
 - HiRDB status, checking (upgrading) 15
 - HiRDB system construction with OLTP linkage 176
 - HiRDB system definition
 - creating 60
 - creating (HiRDB/Parallel Server) 62
 - creating (HiRDB/Single Server) 60
 - modifying 874
 - modifying (excluding UAP environment definitions) 69
 - specifying (multi-HiRDB) 270
 - HiRDB system definition file
 - configuring (HiRDB/Parallel Server) 66
 - configuring (HiRDB/Single Server) 62
 - HiRDB Text Search Plug-in 146
 - HiRDB XA library 174
 - functions supported by 175
 - HiRDB/Developer's kit 859
 - HiRDB/Parallel Server
 - estimating size of memory required for 543
 - uninstalling 52
 - HiRDB/Single Server
 - estimating size of memory required for 502
 - uninstalling 51
 - host name 828, 859
 - specified in pdstart operand 859
 - Hub optimization information definition, creating 66
- I**
- index 447
 - creating 447
 - defining 157, 821
 - definition for partitioned table 861
 - designing 445
 - having B-tree structure 447
 - how to use 861
 - items to be examined during design of 446
 - key length, list of 637
 - maximum key length of 448
 - multicolumn 453
 - non-partitioning key index 457
 - partitioning guideline of 459
 - partitioning key index 457
 - priority order of 449
 - row partitioning of 457
 - single-column 453
 - storage pages, calculating number of 633, 638
 - storage pages, number of 669
 - using, with exceptional key value set 455
 - index global buffer 278
 - allocating 274
 - index local buffer 297
 - allocating 297
 - index log information
 - determining amount of 714
 - determining amount of, for index page splitting 714
 - index page splitting 714
 - index storage RDAREA, notes on estimating size of 474
 - inheritance 153, 386
 - input data file UOC 137
 - insert history maintenance column 375
 - INSERT ONLY operand (CREATE TABLE) 375
 - installation directory 858
 - installer to update HiRDB, using 22
 - installing
 - multi-HiRDB 268
 - option program product 49
 - plug-in 119
 - inter-process memory communication shared memory
 - HiRDB/Parallel Server 549
 - HiRDB/Single Server 507
 - IP address 828
- J**
- JP1/AJS2 203
 - JP1/Automatic Job Management System 2 199
 - JP1/Base 199
 - JP1/IM 201
 - JP1/Integrated Management - Manager 199
- K**
- key range partitioning 314

examples of (with boundary values specified) 323
 examples of (with storage condition specified) 321
 KFPS01861-E message 866
 KFPS05078-I message 865

L

libraries for multi-thread, notes about 195
 list RDAREA
 creating 113
 creating HiRDB file system area for 78
 designing 485
 designing HiRDB file system areas for 210, 244
 determining size of 697
 placing (HiRDB/Parallel Server) 263
 placing (HiRDB/Single Server) 228
 LOB column 366
 LOB global buffer 278
 allocating 278
 local buffer 297
 designing 273
 location information work table
 obtaining maximum number of 743
 obtaining size of 743
 log acquisition mode 132, 159
 LRU management method 290

M

master directory RDAREA
 determining size of 682
 page length of 682
 total number of pages in 682
 matrix partitioning 336
 matrix-partitioned table 336
 memory
 allocation of (HiRDB/Parallel Server) 543
 allocation of (HiRDB/Single Server) 502
 memory requirement
 calculation of (HiRDB/Parallel Server) 547
 calculation of (HiRDB/Single Server) 505
 checking (upgrading) 15
 during array FETCH (front-end server) 615

during array FETCH (HiRDB/Single Server) 541
 during BLOB data retrieval (back-end server) 614
 during BLOB data retrieval (dictionary server) 614
 during BLOB data retrieval (front-end server) 614
 during BLOB data retrieval (HiRDB/Single Server) 540
 during BLOB data updating (back-end server) 614
 during BLOB data updating (dictionary server) 614
 during BLOB data updating (front-end server) 614
 during BLOB data updating (HiRDB/Single Server) 540
 during block transfer (front-end server) 615
 during block transfer (HiRDB/Single Server) 541
 during execution of rapid grouping facility (HiRDB/Parallel Server) 603
 during execution of rapid grouping facility (HiRDB/Single Server) 531
 during hash join 533, 605
 during SQL execution (HiRDB/Parallel Server) 603
 during SQL execution (HiRDB/Single Server) 531
 during SQL preprocessing (HiRDB/Parallel Server) 612
 during SQL preprocessing (HiRDB/Single Server) 539
 during subquery hash execution 533, 605
 estimating, for HiRDB/Parallel Server 543
 estimating, for HiRDB/Single Server 502
 for foreign table retrieval 607
 for Java virtual machine (HiRDB/Parallel Server) 550
 for Java virtual machine (HiRDB/Single Server) 508
 memory size
 checking (upgrading) 14

- required for execution of access path display utility (pdvwopt) 792
- required for execution of database condition analysis utility (pddbst) 784
- required for execution of database copy utility (pdcopy) 786
- required for execution of database definition utility (pddef) 774
- required for execution of database initialization utility (pdinit) 773
- required for execution of database load utility (pload) 774
- required for execution of database recovery utility (pdrstr) 788
- required for execution of database reorganization utility (pdrorg) 779
- required for execution of database structure modification utility (pdmod) 781
- required for execution of dictionary import/export utility (pdexp) 790
- required for execution of optimizing information collection utility (pdgetest) 785
- required for execution of statistics analysis utility (pdstedit) 783
- required for utility execution, determining 773
- message queue identifiers, number of 802, 804
- message queue tables, number of 802, 804
- monitoring free area for system log files facility
 - HiRDB/Parallel Server 246
 - HiRDB/Single Server 212
- multi-connection facility 176
 - X/Open XA interface environment 185
- multi-HiRDB
 - installing 268
 - setting environment for 270
 - system design for 268
- multi-thread, XA interface supporting 176
- multicolumn partitioning 457
- multiple front-end server 231
 - example of configuration using 233
 - setting up 232
 - using 231

N

- network configuration
 - example of (FQDN) 829
 - example of (multi-connection address facility) 830
- network definition
 - example of (FQDN) 829
 - example of (multi-connection address facility) 830
- network drive, use of 859
- new page allocation mode 488
- new values correlation name 345
- NO SPLIT option 365
- no-log mode 132, 159
- no-split option, specifying 364
- non-UNIQUE attribute 861
- normalizing table 308
- NOT NULL constraint 360
- number reserved 493

O

- old correlation name 345
- old or new values alias 345
- OLTP products supported for linking 174
- open character string 185
- OpenTP1 174, 860
- optimizing based on cost 449
- OS environment file, setting up (installation) 40
- OTS 176
- override 388

P

- page 480
 - allocating 483
 - determining length of 480
 - free 480
 - full 480
 - releasing 484
 - unused 480
 - used 480
 - used free 480
- page search mode switchovers, number of 494
- partitioning key 314

- selecting 315
- password, changing (database creation) 130
- PATH 44
- PCTFREE option (CREATE TABLE) 478, 483
- pd_assurance_table_no operand 493
- pd_check_pending operand 400, 434
- pd_dbbuff_lru_option operand 290
- pd_dbbuff_rate_updpage operand 288
- pd_dbsync_point operand 286, 289
- pd_dfw_awt_process operand 288
- pd_foreign_server_libpath operand 197
- pd_hb_ary_fec_num operand 607
- pd_jp1_event_level operand 199
- pd_jp1_use operand 199
- pd_large_file_use operand 73, 210, 245
- pd_log_dual operand 215, 249
- pd_log_rec_leng operand 215, 249
- pd_log_rerun_reserved_file_open operand 215, 249
- pd_log_rpl_no_standby_file_opr operand 172
- pd_log_singleoperation operand 215, 249
- pd_max_foreign_server operand 197
- pd_ntfs_cache_disable operand 210, 245
- pd_pageaccess_mode operand 293
- pd_registered_port operand 845, 848
- pd_rpl_hdepath operand 172
- pd_rpl_init_start operand 172
- pd_shared_rdarea_use operand 497
- pd_spd_assurance_count operand 218, 253
- pd_spd_dual operand 217, 252
- pd_spd_reduced_mode operand 219, 253
- pd_spd_reserved_file_auto_open operand 219, 253
- pd_spool_cleanup operand 47
- pd_spool_cleanup_interval operand 47
- pd_spool_cleanup_interval_level operand 47
- pd_spool_cleanup_level operand 48
- pd_sts_singleoperation operand 222, 256
- pd_sysdef_default_option operand 12
- pd_syssts_singleoperation operand 222, 256
- pdadmvr command 23
- pdbuffer operand 278
- pdbufmod command 274
- pdchgconf command 70
- pdconfchk command 60
- PDCONFPATH 44
- pdcspool command 47
- pddef control statement 859
- PDDIR 44
- pdfmkfs command 72, 497
- pdhubopt operand 197
- pdinit 96
- pdload 132
- pdloginit command 80, 81
- pdls -d ust command, when upgrading HiRDB 15
- pdls command 26
- pdmod 101
- pdntenv command 39, 41
- pdpgbfon 296
- pdplgrgst command 119
- pdplugin operand 122
- pdprgcopy command 26
- pdprgrefresh command 26
- pdprlstart command 237
- pdprlstop command 237
- pdstart command
 - fails to start any particular unit by 865
 - fails to start HiRDB by 865
 - results in error 866
 - when HiRDB startup is slow using 865
- pdstbegin operand 197
- pdstsinit command 82
- PDUXPLDIR 44
- PDUXPLMSGMNI 802, 804
- PDUXPLMSGTQL 802, 804
- PDUXPLSEMMAX 802, 804
- PDUXPLSHMMAX 802, 804
- peak capacity (HiRDB file system area for work table file) 864
- percentage of free pages in a segment 478
- percentage of unused space in a page 482
 - obtaining 483
 - setting 482
- plug-in
 - deleting 127
 - installing 119
 - owner of 121
 - registering 119
 - uninstalling 128
 - upgrading 125

plug-in environment, setting up 117
 plug-in index 463
 defining 149
 effects of row partitioning of 464
 row partitioning of 464
 port number 845
 for connection from client, specifying (multi-HiRDB) 271
 post-installation procedures 47
 pre-update log acquisition mode 132, 159
 prefetch facility 283, 476
 previous version, restoring to 19
 PRIMARY KEY option (CREATE TABLE) 360
 primary key, specifying 360
 PRIVATE 389
 process private area (memory requirement) 506, 548
 PROTECTED 389
 Psp4017 865
 PUBLIC 389

Q

questions and answers 857

R

raw I/O facility 73
 creating HiRDB file system area using 73, 78
 RDAREA
 creating HiRDB file system area for 74
 deleting unneeded 137
 designing 471
 designing HiRDB file system areas for 207, 241
 determining size of 617
 items to be examined during design of 472
 maximum value for 474
 minimum value for 474
 placing (HiRDB/Parallel Server) 259
 placing (HiRDB/Single Server) 225
 read only 175
 reason code=TIMEOUT 866
 rebalancing facility 307, 335
 record length of system log file 215, 249
 RECOVERY operand 159
 CREATE TABLE 133

recovery-unnecessary front-end server 236
 recovery-unnecessary front-end server unit 236
 reduced mode operation
 HiRDB/Parallel Server 253
 HiRDB/Single Server 219
 reference buffer 274
 reference-only back-end server 391
 referenced table 399
 referencing table 399
 referential constraint 399
 registry facility, initializing 121
 registry information, registering 123
 registry LOB RDAREA
 determining size of 696
 page length of 696
 total number of pages in 696
 registry RDAREA, determining size of 694
 registry, deleting 128
 reload-not-completed data status 377
 repetition column 382
 replication facility, linking to 172
 resource manager 174
 RM 174
 RM name 185
 RM switch name 185
 RM-related object name 188
 routine 389
 row partitioned table 314
 creating 138

S

sample configuration 868
 sample configuration file, name of 813
 sample database
 creating 819
 customizing 820
 sample database file, name of 813
 sample file 811
 use of 819
 sample UOC file, name of 814
 sampleDB1.bat 819
 sampleDB2.bat 819
 sampleDB3.bat 819
 sampleDB4.bat 819

- schema, defining (database creation) 130
- second dimension partitioning column 336
- segment 476
 - allocating 479
 - determining size of 476
 - free 476
 - full 476
 - number of 684, 685, 693
 - releasing 479
 - setting percentage of free pages in 478
 - unused 476
 - used 476
 - used free 476
- SEGMENT REUSE option 493
 - ALTER TABLE 493
- semaphore identifiers, number of 802, 804
- server machine environment (installation) 36
- server name operand 497
- service port number
 - registering, in OS (multi-HiRDB) 269
 - specifying (multi-HiRDB) 268
- setting
 - when firewall is installed 838
 - when NAT is installed 838
- Setup with Identifier (multi-HiRDB) 268
- shared memory
 - formula for, used by each server (HiRDB/Parallel Server) 588
 - formula for, used by single server (HiRDB/Single Server) 524
 - memory requirement 507, 549
 - usage count, number of 802, 804
 - used by back-end server 594
 - used by dictionary server 590
 - used by front-end server 588
 - used by global buffers (HiRDB/Parallel Server) 599
 - used by global buffers (HiRDB/Single Server) 529
 - used by single server 524
 - used by unit controller (HiRDB/Parallel Server) 556
 - used by unit controller (HiRDB/Single Server) 514
- shared RDAREA 496
- shared table 391
- simple setup tool 55
- single operation
 - HiRDB/Parallel Server 249
 - HiRDB/Single Server 215
- single server definition, creating 61, 63
- single-byte character encoding 47
- single-column partitioning 457
- single-phase optimization 175
 - notes on 195
- snapshot method 293
- space conversion facility 131
- SPsetup.bat 868
- SQL reserved word definition
 - creating (HiRDB/Single Server) 62
 - creating (HiRDB/Parallel Server) 65
- SQL, notes about (X/Open XA interface environment) 195
- static registration 184
- status file
 - creating 82
 - designing (HiRDB/Parallel Server) 253
 - designing (HiRDB/Single Server) 219
 - determining number of records in 727
 - determining size of 727
 - handling of (dual status file) 862
 - handling of (status file definition) 863
 - handling of (status file placement) 863
 - handling of (when error occurs) 862
 - single operation of (HiRDB/Parallel Server) 256
 - single operation of (HiRDB/Single Server) 222
- status-file double operation
 - HiRDB/Parallel Server 256
 - HiRDB/Single Server 222
- status-file single operation
 - HiRDB/Parallel Server 256
 - HiRDB/Single Server 222
- storage condition, specifying 315
- storage requirement
 - for HiRDB 501
 - for utility execution 751

- substitutability 154, 387
- subtype 386
- Sun Java System Directory Server 4
- supertype 386
- supported machine, checking (installation) 36
- SUPPRESS option (CREATE TABLE) 363
- suppress option, specifying 363
- synchronization point dump file
 - automatic opening of (HiRDB/Parallel Server) 253
 - automatic opening of (HiRDB/Single Server) 219
 - creating 81
 - designing (HiRDB/Parallel Server) 250
 - designing (HiRDB/Single Server) 216
 - determining number of records in 726
 - determining size of 726
 - duplexing of 217, 251
 - number of guaranteed-valid generations for (HiRDB/Parallel Server) 252
 - number of guaranteed-valid generations for (HiRDB/Single Server) 217
 - reduced mode operation for (HiRDB/Parallel Server) 253
 - reduced mode operation for (HiRDB/Single Server) 219
- synchronization point dump, handling of 861
- synchronization point, line number of 136
- system cache, checking (installation) 39
- system common definition
 - creating (HiRDB/Parallel Server) 62
 - creating (HiRDB/Single Server) 60
- system configuration
 - checking (multi-HiRDB) 268
 - maximum value for 852
 - minimum value for 852
 - of HiRDB/Parallel Server 231
 - of HiRDB/Single Server 206
- system design
 - for HiRDB/Parallel Server 230
 - for HiRDB/Single Server 206
 - for multi-HiRDB 268
- system file
 - checking (installation) 40
 - creating 80
 - creating HiRDB file system area for 75
 - designing (HiRDB/Parallel Server) 246
 - designing (HiRDB/Single Server) 212
 - designing HiRDB file system areas for 208, 242
 - determining size of 699
 - example of creating (HiRDB/Parallel Server) 86
 - example of creating (HiRDB/Single Server) 82
- system log file
 - automatic opening of (HiRDB/Parallel Server) 249
 - automatic opening of (HiRDB/Single Server) 215
 - creating 80
 - designing (HiRDB/Parallel Server) 246
 - designing (HiRDB/Single Server) 212
 - determining size of 700
 - double operation of (HiRDB/Parallel Server) 249
 - double operation of (HiRDB/Single Server) 215
 - duplexing of (HiRDB/Parallel Server) 249
 - duplexing of (HiRDB/Single Server) 214
 - record length of 700
 - record length of (HiRDB/Parallel Server) 249
 - record length of (HiRDB/Single Server) 215
 - records, number of 701
 - single operation of (HiRDB/Parallel Server) 249
 - single operation of (HiRDB/Single Server) 215
 - total size of 700
- system log information
 - determining size of 701
 - output depending on SQL manipulation, amount of 722
 - output during database creation by utility, amount of 719
 - output during execution of RDAREA automatic extension facility, amount of 722
 - output during index definition, amount of 704

- output during table data updating, amount of 705
 - output during table definition, amount of 703
 - system manager, placement of 230
 - system RDAREA
 - backing up (upgrading) 14
 - creating 96
 - placing (HiRDB/Parallel Server) 259
 - placing (HiRDB/Single Server) 225
 - system reconfiguration command 70
 - system switchover configuration, migrating to (multi-HiRDB) 272
 - system switchover facility
 - relationship with (multi-HiRDB) 272
 - using 4
- T**
- table
 - actual 356
 - containing abstract data type 385
 - containing repetition column 382
 - creating, containing plug-in-provided abstract data type 146
 - creating, containing user-defined abstract data type 151
 - creating, with LOB column 142
 - defining 155, 821
 - definition information of (sample file) 815
 - design method for row partitioning of 314
 - designing 301
 - effects of row partitioning of 326
 - forms of row partitioning of 325
 - items to be examined during design of 302
 - matrix partitioning of 336
 - normalizing 308
 - partitioning, among multiple servers 462
 - partitioning, within one server 461
 - procedure for creating, with abstract data type (SGMLTEXT type) 146
 - row partitioning of 314
 - storage pages, calculating number of 619, 630, 647
 - storing data in 157
 - view 356
 - table integrity
 - how to check (check constraint) 438
 - how to check (referential constraint) 423
 - TIMEOUT 866
 - TM 174
 - TPBroker 174
 - transaction completion type 195
 - transaction manager 174
 - example of registering in 190
 - information to be registered in 185
 - modifying registration information in 192
 - registering HiRDB in 183
 - transaction transfer 175, 180
 - trigger
 - defining 343
 - management of 350
 - trigger action search conditions 343
 - trigger event 343
 - trigger event SQL 343
 - trnstring operand 182
 - TUXEDO 174
 - TZ system environment variable, checking (installation) 36
- U**
- UAP environment definition
 - creating (HiRDB/Parallel Server) 65
 - creating (HiRDB/Single Server) 61
 - modifying 71
 - Unicode 47
 - uninstalling plug-in 128
 - unique index 136
 - uniqueness constraint 360
 - unit control information definition
 - creating (HiRDB/Parallel Server) 63
 - creating (HiRDB/Single Server) 60
 - UOC 137
 - updatable back-end server 391
 - updatable column 375
 - update buffer 274
 - update patch to update HiRDB, using 22
 - upgrading
 - backing up system RDAREAs before 14
 - before 12

- checking HiRDB status before 15
- checking memory requirement before 15
- checking memory size before 14
- checking to see whether HiRDB is online before 14
- checking total number of records in system log file before 16
- free space checking before 12
- HiRDB 12
- HiRDB plug-in 18
- plug-in 125
- stopping HiRDB service before 15
- terminating HiRDB normally before 15
- used free pages, release of 484
- user LOB RDAREA
 - creating 105
 - determining size of 693
 - page length of 693
 - placing (HiRDB/Parallel Server) 262
 - placing (HiRDB/Single Server) 227
 - total number of pages in 693
- user RDAREA
 - creating 101
 - determining size of 618
 - formula for calculating total number of pages in 618
 - placing (HiRDB/Parallel Server) 261
 - placing (HiRDB/Single Server) 227
- user's own coding 137
- user, adding new 820
- UTF-8 47
- utility
 - creating HiRDB file system area for 77
 - designing HiRDB file system areas for 209, 243

V

- view table, creating 356
- virtual memory
 - checking (installation) 38
 - method for determining size of 858

W

- WebLogic Server 174

- WITHOUT ROLLBACK option, specifying 371
- work console, how to use (multi-HiRDB) 271
- work disk 333
- work table file 736
 - creating HiRDB file system area for 76
 - designing HiRDB file system areas for 208, 242
 - determining maximum number of 747
 - determining maximum number of extensions 749
 - determining size of 735
 - used by SQL statement, size of 738
 - used by utility, size of 744

X

- X/Open XA interface 174
- XA interface 860
 - supporting multi-thread 176
- xa_switch_t structure-name 185

Reader's Comment Form

We would appreciate your comments and suggestions on this manual. We will use these comments to improve our manuals. When you send a comment or suggestion, please include the manual name and manual number. You can send your comments by any of the following methods:

- Send email to your local Hitachi representative.
- Send email to the following address:
WWW-mk@itg.hitachi.co.jp
- If you do not have access to email, please fill out the following information and submit this form to your Hitachi representative:

Manual name:	
Manual number:	
Your name:	
Company or organization:	
Street address:	
Comment:	

(For Hitachi use)
