
For Windows Systems
Scalable Database Server

HiRDB Version 8

Description

3020-6-351(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 (3020-6-351(E)): March, 2007

■ Copyright

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

Preface

This manual provides an overview and description of the functions of *HiRDB Version 8*, a scalable database server program product.

Intended readers

This manual is intended for users who will be constructing or operating *HiRDB Version 8* ("HiRDB") relational database systems.

It is assumed that readers of this manual have the following:

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

Organization of this manual

This manual is organized as follows:

Chapter 1. *Overview*

Explains HiRDB features, system configurations, and access modes, and describes HiRDB option program products and other HiRDB-related products.

Chapter 2. *Linking to HiRDB Option Program Products and Other HiRDB-Related Products*

Explains the functions that can be implemented by linking HiRDB to HiRDB option program products and other HiRDB-related products.

Chapter 3. *Database Logical Structure*

Explains the logical structure of a HiRDB database (RDAREAs, schemas, tables, indexes, and expansion into an object relational database).

Chapter 4. *Database Physical Structure*

Explains the physical structure of a HiRDB database (segments, and pages).

Chapter 5. *Database Access Using SQL*

Explains the use of the SQL database manipulation language to manipulate data in HiRDB.

Chapter 6. *HiRDB Architecture*

Explains the architecture of HiRDB.

Chapter 7. *Database Management*

Explains how the HiRDB administrator manages a database.

Chapter 8. *Error-handling Facilities*

Explains error-handling facilities (system switchover facility and recovery-unnecessary front-end servers).

Chapter 9. *Facilities Related to Security Measures*

Describes facilities that provide support for database security measures (security facility, security audit facility, and connection security facility).

Chapter 10. *Plug-ins*

Provides an overview of the HiRDB plug-ins feature and explains its functions and the functions of HiRDB when the plug-ins feature is used.

Chapter 11. *64-Bit-Mode HiRDB*

Explains HiRDB operation in the 64-bit mode, which is provided for support of large systems.

Appendix A. *Functional Differences Between HiRDB Versions on Different Platforms*

Explains the functional differences between HiRDB versions on different platforms.

Appendix B. *Data Dictionary Tables*

Explains the HiRDB data dictionary tables.

Appendix C. *HiRDB Client and HiRDB Server Connectivity*

Provides lists by platform of connectable and non-connectable HiRDB clients and servers.

Appendix D. *Glossary*

Explains the terms used in the HiRDB Version 8 manuals.

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 Installation and Design Guide* (3020-6-352(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

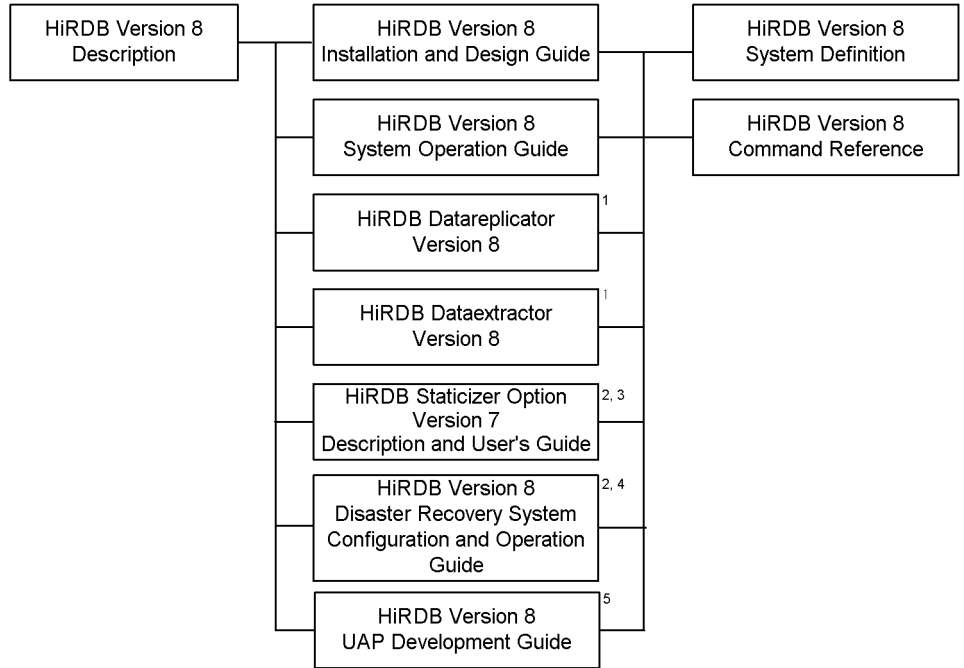
- *HiRDB External Data Access Version 7 Description and User's Guide (3000-6-284(E))*
- *Job Management Partner 1/Integrated Management - Manager System Configuration and User's Guide (3020-3-K01(E))*
- *Job Management Partner 1/Base User's Guide (3020-3-K06(E))*
- *Job Management Partner 1/Automatic Job Management System 2 Description (3020-3-K21(E))*
- *Job Management Partner 1/Integrated Manager - Console (3020-3-F01(E))*

- *Job Management Partner 1/Base* (3020-3-F04(E))
- *Job Management Partner 1/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))
- *JP1 Version 6 JP1/VERITAS NetBackup v4.5 Agent for HiRDB License Description and User's Guide* (3020-3-D79(E))

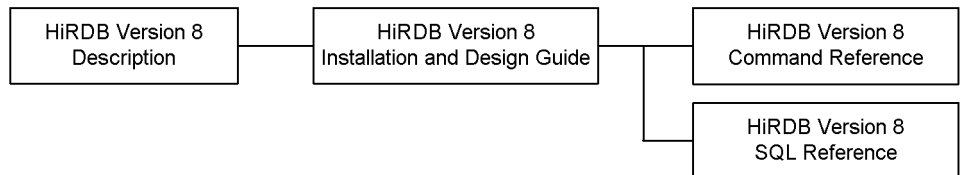
Organization of HiRDB manuals

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.

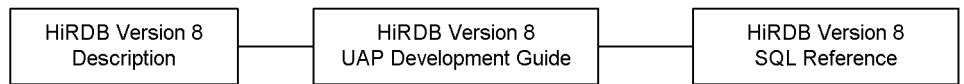
Manuals for system administrators:



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

Name of product or other entity	Representation	
Virtual-storage Operating System 3/Forefront System Product	VOS3/FS	VOS3
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	Windows XP
Microsoft _(R) Windows _(R) XP Home Edition Operating System	Windows XP Home Edition	
Single server	SDS	

Name of product or other entity	Representation
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
CMT	Cassette Magnetic Tape
COBOL	Common Business Oriented Language
CORBA(R)	Common ORB Architecture

Abbreviation	Full name or meaning
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
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GUI	Graphical User Interface

Abbreviation	Full name or meaning
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
LU	Logical Unit
LUN	Logical Unit Number
LVM	Logical Volume Manager

Abbreviation	Full name or meaning
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
PP	Program Product
PR	Protected Retrieve
PU	Protected Update

Abbreviation	Full name or meaning
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
VOS K	Virtual-storage Operating System Kindness
VOS1	Virtual-storage Operating System 1
VOS3	Virtual-storage Operating System 3

Abbreviation	Full name or meaning
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:

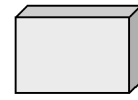
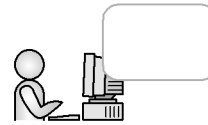
1. Choose **Start, Programs, Administrative Tools (Common)**, and then **Event Viewer**.
2. Choose **Log**, and then **Application**.
3. The application log is displayed. Messages with **HiRDBSingleServer** or **HiRDBParallelServer** displayed in the **Source** column were issued by HiRDB.

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

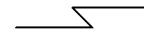
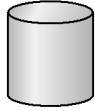
Conventions: Diagrams

This manual uses the following conventions in diagrams:

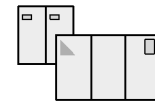
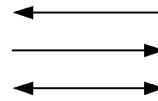
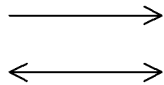
- Personal computer (PC) • Input/output operation or workstation (WS)
- Window or display text
- Program¹ or server



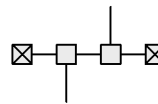
- File² or magnetic disk
- Magnetic disk²
- CMT or DAT²
- Communication line



- Flow of control
- Flow of data
- Other flows
- Mainframe



- Step in procedure
- Network (WAN)
- Network (LAN)



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

² Input data files, unload files, and backup files can be stored on 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.

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	ii
Organization of HiRDB manuals	iv
Conventions: Abbreviations	v
Log representations	xv
Conventions: Diagrams	xv
Conventions: Fonts and symbols	xvi
Notes on Windows path names	xviii
Conventions: KB, MB, GB, and TB	xviii
Conventions: Version numbers	xix
Important notes on this manual	xix
1. Overview	1
1.1 Characteristics of HiRDB	2
1.1.1 Overview of HiRDB systems	2
1.1.2 Advantages of using HiRDB	6
1.2 HiRDB system configuration	11
1.2.1 HiRDB/Single Server configuration	11
1.2.2 HiRDB/Parallel Server configuration	11
1.2.3 Multi-HiRDB configuration	15
1.3 Database access modes	17
2. Linking to HiRDB Option Program Products and Other HiRDB-Related Products	23
2.1 HiRDB option program products	24
2.1.1 HiRDB Advanced High Availability	24
2.1.2 HiRDB Advanced Partitioning Option	25
2.1.3 HiRDB LDAP Option	25
2.1.4 HiRDB External Data Access	26
2.1.5 HiRDB Non-Recover FES	30
2.2 Linkage to data linkage products	31
2.2.1 Linkage to HiRDB Datareplicator and HiRDB Dataextractor	31
2.2.2 Linkage to HiRDB Adapter for XML	35
2.3 Linkage to directory server products	37
2.3.1 Overview of the Directory Server linkage facility	37
2.3.2 Directory servers to which HiRDB can be linked	38

2.3.3	Capabilities provided by the Directory Server linkage facility	39
2.3.4	Prerequisite products	41
2.4	Linkage to OLTP products	42
2.4.1	OLTP products to which HiRDB can be linked	42
2.4.2	HiRDB XA library	42
2.4.3	Functions provided by the HiRDB XA library	43
2.4.4	System configuration	44
2.4.5	Registration of HiRDB in the transaction manager	46
2.5	Linkage to operation support products	48
2.5.1	HiRDB SQL Executer	48
2.5.2	HiRDB Control Manager	48
2.5.3	HiRDB SQL Tuning Advisor	49
2.5.4	JP1/Performance Management - Agent Option for HiRDB	49
2.5.5	JP1/Base	50
2.5.6	JP1/Integrated Management	50
2.5.7	JP1/Automatic Job Management System 2	51
2.6	Linkage to data mining products	53
2.7	Linkage to products that handle multimedia information	54
2.8	Linkage to Cosminexus	58

3. Database Logical Structure 61

3.1	RDAREAs	62
3.1.1	Types of RDAREAs	62
3.1.2	RDAREA creation	64
3.2	Schemas	67
3.3	Tables	68
3.3.1	Basic table structure	68
3.3.2	Table normalization	71
3.3.3	FIX attribute	73
3.3.4	Primary key	73
3.3.5	Cluster key	73
3.3.6	Suppress option	74
3.3.7	No-split option	74
3.3.8	Table row partitioning	76
3.3.9	Table matrix partitioning	85
3.3.10	Changing the partitioning storage conditions of a table	91
3.3.11	Falsification-prevention table	93
3.3.12	Tables used in numbering	95
3.3.13	Repetition columns	96
3.3.14	View table	98
3.3.15	Shared tables	99
3.4	Indexes	101
3.4.1	Basic structure of an index	101
3.4.2	Index row partitioning	103

3.4.3	Index page splitting	109
3.4.4	Exception key value	112
3.4.5	Defining an index for a table that contains data.....	113
3.4.6	Index key value no-lock	113
3.5	Expansion into an object relational database.....	118
3.5.1	Abstract data types	118
3.5.2	Subtypes and inheritance.....	126
3.5.3	Encapsulation	132
4.	Database Physical Structure	137
4.1	Database physical structure	138
4.2	Segment design.....	140
4.3	Page design.....	146
5.	Database Access Using SQL	149
5.1	Use of SQL in HiRDB.....	150
5.1.1	HiRDB SQL functions	150
5.1.2	SQL execution methods	150
5.2	Basic data manipulation	152
5.2.1	Cursor	152
5.2.2	Data retrieval	152
5.2.3	Data updating	153
5.2.4	Data deletion	154
5.2.5	Data insertion	154
5.2.6	Searching for specific data	155
5.2.7	Data operations.....	157
5.2.8	Data processing	158
5.2.9	Manipulation of data in a table containing an abstract data type	159
5.3	Stored procedures and stored functions.....	163
5.4	Java stored procedures and Java stored functions	169
5.4.1	Characteristics of a Java stored routine.....	169
5.4.2	System configuration (position of a Java virtual machine).....	169
5.4.3	Execution of Java stored routines.....	170
5.4.4	Java stored routine creation and execution procedure	171
5.5	Triggers.....	174
5.6	Integrity constraints	176
5.6.1	NOT NULL constraint	176
5.6.2	Uniqueness constraint	176
5.7	Referential constraints	178
5.8	Check constraints.....	182
5.9	Check pending status.....	184
5.10	Improving database access performance	186
5.10.1	Block transfer facility.....	186
5.10.2	Rapid grouping facility.....	188

5.10.3	Functions that use arrays	188
5.10.4	Holdable cursor.....	191
5.10.5	SQL optimization	191
5.11	Narrowed search	197
5.12	Accessing databases using DB access products	200
5.12.1	ODBC Driver.....	200
5.12.2	HiRDB OLE DB Provider.....	200
5.12.3	HiRDB.NET Data Provider.....	200
5.12.4	JDBC Driver.....	200
5.12.5	SQLJ	201

6. HiRDB Architecture 203

6.1	HiRDB environment setup.....	204
6.2	HiRDB file system areas.....	206
6.3	System files.....	209
6.3.1	System log files	209
6.3.2	Synchronization point dump files.....	210
6.3.3	Status files.....	212
6.3.4	System file components.....	214
6.4	Work table files	217
6.5	HiRDB system definitions	220
6.5.1	HiRDB system definitions for a HiRDB/Single Server	220
6.5.2	HiRDB system definitions for a HiRDB/Parallel Server	221
6.5.3	HiRDB system definition file creation	225
6.5.4	System reconfiguration command (pdchgconf command).....	225
6.6	HiRDB startup and termination	226
6.6.1	Startup and termination modes.....	226
6.6.2	HiRDB automatic startup	229
6.6.3	Reduced activation (applicable to HiRDB/Parallel Server only).....	229
6.7	Delayed rerun.....	230
6.8	Database access processing method	233
6.8.1	Global buffers.....	233
6.8.2	Prefetch facility.....	236
6.8.3	Asynchronous READ facility.....	237
6.8.4	Deferred write processing.....	238
6.8.5	Facility for parallel writes in deferred write processing.....	238
6.8.6	Incorporation during commit.....	239
6.8.7	LRU management method for global buffers.....	239
6.8.8	Accessing pages using the snapshot method	240
6.8.9	Global buffer pre-writing.....	240
6.8.10	Local buffers.....	242
6.8.11	BLOB data file output facility	244
6.8.12	BLOB and BINARY data addition update and partial extraction facility.....	248
6.8.13	Locator facility	250

6.9	Transaction control	254
6.9.1	Connection to and disconnection from HiRDB	254
6.9.2	Multi-connection facility	254
6.9.3	Transaction startup and termination	255
6.9.4	Commit and rollback	256
6.9.5	UAP transaction management under OLTP environment	259
6.9.6	Automatic reconnect facility	259
6.10	Locking	261
6.10.1	Units of locking	261
6.10.2	Lock modes	262
6.10.3	Automatic locking by HiRDB	263
6.10.4	Changing the lock based on a user setting	263
6.10.5	Lock period	264
6.10.6	Deadlock	264
6.11	Operation without collecting a database update log	265
7.	Database Management	269
7.1	Database recovery	270
7.1.1	Overview of database recovery	270
7.1.2	Times at which database can be recovered	270
7.2	Preparations for database errors	272
7.2.1	Making backups	272
7.2.2	Creating an unload log file (unloading system log)	273
7.2.3	Differential backup facility	277
7.2.4	Backup-hold	279
7.2.5	Reducing the time needed to make backups of user LOB RDAREAs (frozen update command)	280
7.2.6	NetBackup linkage facility	282
7.3	Reorganizing tables and indexes	285
7.3.1	Table reorganization	285
7.3.2	Index reorganization	291
7.4	Reusing used free pages and used free segments	294
7.4.1	Reusing used free pages	294
7.4.2	Reusing used free segments	297
7.5	Adding, expanding, and moving RDAREAs	299
7.5.1	Adding RDAREAs	299
7.5.2	Expanding an RDAREa	299
7.5.3	RDAREa automatic extension	299
7.5.4	Moving RDAREAs (HiRDB/Parallel Server only)	301
7.6	Space conversion facility	303
7.7	Facility for conversion to a decimal signed normalized number	306
8.	Error-handling Facilities	309
8.1	System switchover facility	310

8.1.1	Overview of the system switchover facility	310
8.1.2	Monitor mode and server mode.....	316
8.1.3	System switchover facility configurations	318
8.1.4	System configuration examples.....	318
8.1.5	Functions that reduce system switchover time (user server hot standby and the rapid system switchover facility)	325
8.2	Recovery-unnecessary front-end servers	327
8.2.1	Overview of recovery-unnecessary front-end servers.....	327
8.2.2	Configuration example of a system that uses a recovery-unnecessary front-end server	328
9.	Facilities Related to Security Measures	329
9.1	Security facility.....	330
9.1.1	User privileges.....	330
9.1.2	Operating the security facility	333
9.2	Security audit facility.....	334
9.2.1	Overview of the security audit facility	334
9.2.2	Audit events.....	340
9.3	Connection security facility	345
9.3.1	Overview of the connection security facility.....	345
9.3.2	Restrictions on password character strings.....	345
9.3.3	Restrictions on the number of consecutive certification failures	347
10.	Plug-ins	349
10.1	Overview of HiRDB plug-ins	350
10.2	Applying a plug-in to a job	351
10.3	HiRDB plug-in facilities	354
10.3.1	HiRDB Text Search Plug-in	354
10.3.2	HiRDB Image Search Plug-in	355
10.3.3	HiRDB File Link	356
10.3.4	HiRDB Spatial Search Plug-in	356
10.4	Preparations for using plug-ins in HiRDB.....	358
10.4.1	Setup and registration of a plug-in	358
10.4.2	Initialization of the registry facility	358
10.4.3	Table definition for plug-in usage.....	359
10.4.4	Delayed batch creation of plug-in index.....	359
11.	64-Bit-Mode HiRDB	361
11.1	Overview.....	362
11.2	Functions not available in 64-bit-mode HiRDB	363
11.3	Differences between 32- and 64-bit modes.....	365
11.3.1	Differences in HiRDB system definition.....	365
11.3.2	Differences in client environment definition.....	366
11.3.3	64-bit-mode HiRDB client support range.....	366

11.4 Migrating from 32-bit mode to 64-bit mode.....	368
Appendixes	373
A. Functional Differences Between HiRDB Versions on Different Platforms	374
B. Data Dictionary Tables	379
C. HiRDB Client and HiRDB Server Connectivity	383
D. Glossary	387
Index	437

List of figures

Figure 1-1: Configuration of a HiRDB system	3
Figure 1-2: Overview of the XDM/RD E2 connection facility.....	6
Figure 1-3: HiRDB/Single Server configuration	11
Figure 1-4: HiRDB/Parallel Server configuration	12
Figure 1-5: Floatable server	14
Figure 1-6: Example of HiRDB/Parallel Server in a heterogeneous system configuration.....	15
Figure 1-7: Multi-HiRDB configuration using HiRDB/Single Servers.....	16
Figure 1-8: Accessing a database by executing UAPs from HiRDB clients	17
Figure 1-9: Accessing a database by executing a UAP running on the HiRDB server	18
Figure 1-10: Accessing a database by executing an OLTP UAP	18
Figure 1-11: Accessing a HiRDB database from an ODBC-compatible application	19
Figure 1-12: Accessing a HiRDB database from an OLE DB-compatible application.....	20
Figure 1-13: Accessing a HiRDB database from a JDBC-compatible application	21
Figure 1-14: Accessing a database from an ADO.NET-compliant application	22
Figure 2-1: Overview of the HiRDB External Data Access facility	27
Figure 2-2: System configuration example when the HiRDB External Data Access facility is being used.....	29
Figure 2-3: Example of using the replication facility	33
Figure 2-4: Overview of the Directory Server linkage facility	38
Figure 2-5: Overview of using the Directory Server linkage facility to perform user authentication (Sun Java System Directory Server linkage facility).....	40
Figure 2-6: Granting of table access privilege to a role	41
Figure 2-7: Position of HiRDB in X/Open DTP model.....	43
Figure 2-8: Linkage between HiRDB/Single Servers and OpenTp1	44
Figure 2-9: Linkage between HiRDB/Parallel Server and OpenTP1	45
Figure 2-10: Linkage to multiple OpenTP1	46
Figure 2-11: Overview of event monitoring using JP1/Integrated Management.....	51
Figure 2-12: Automatic control when system log files are unloaded by linking with JP1	52
Figure 2-13: System configuration of HiRDB-related products	55
Figure 2-14: Linking to Cosminexus	59
Figure 3-1: Relationship between RDAREA, table, and index	62
Figure 3-2: RDAREA configuration example (for HiRDB/Parallel Server).....	66
Figure 3-3: Concept of schemas.....	67
Figure 3-4: Example of a table.....	68
Figure 3-5: Table normalization.....	72
Figure 3-6: Normal data storage method when the actual data size of a variable-size character string is at least 256 bytes.....	75
Figure 3-7: Data storage method used when the no-split option is specified	75
Figure 3-8: Table row partitioning	76
Figure 3-9: Key range partitioning: Example of storage condition specification.....	77

Figure 3-10: Key range partitioning: Example of boundary value specification	79
Figure 3-11: Example of hash partitioning	81
Figure 3-12: Example of table row partitioning: HiRDB/Single Server	82
Figure 3-13: Example of table row partitioning: HiRDB/Parallel Server	83
Figure 3-14: Hash facility for hash row partitioning	84
Figure 3-15: Example of matrix partitioning (combining with key range partitioning with boundary values specified).....	88
Figure 3-16: Example of matrix partitioning (combination of key range partitioning with boundary values specified and hash partitioning).....	90
Figure 3-17: Example of the split facility (boundary value specification).....	92
Figure 3-18: Example of the combine facility (boundary value specification).....	93
Figure 3-19: Example of a table containing repetition columns.....	96
Figure 3-20: Example of a view table.....	98
Figure 3-21: B-tree structure of an index.....	102
Figure 3-22: Index row partitioning.....	103
Figure 3-23: Example of row-partitioning an index: HiRDB/Single Server.....	104
Figure 3-24: Example of row-partitioning an index within the server	106
Figure 3-25: Example of row-partitioning an index among servers.....	107
Figure 3-26: Partitioning key index: Single-column partitioning.....	108
Figure 3-27: Partitioning key index: Multicolumn partitioning	109
Figure 3-28: Example of even-split index page splitting.....	110
Figure 3-29: Example of unbalanced index split.....	111
Figure 3-30: Example of deadlock when the index key value no-lock option is not used	115
Figure 3-31: Example of preventing deadlock by using the index key value no-lock option	116
Figure 3-32: Conceptual model based on real-world information and an abstract data type	119
Figure 3-33: STAFF_TABLE for which an abstract data type is defined	121
Figure 3-34: Encapsulation	122
Figure 3-35: STAFF_TABLE for which values are generated using a constructor function	124
Figure 3-36: Handling of null values in STAFF_TABLE for which an abstract data type is defined.....	125
Figure 3-37: Conceptual model based on real-world information and an abstract data type (for a salesperson)	127
Figure 3-38: Operations related to employees and salespersons in the real world.....	129
Figure 3-39: Override	131
Figure 3-40: Encapsulation levels and access types	134
Figure 4-1: Database physical structure.....	138
Figure 4-2: Ratio of free pages in a segment	140
Figure 4-3: Overview of the free space reuse facility.....	143
Figure 4-4: Ratio of unused area in a page	146
Figure 5-1: Jobs to which a stored procedure can be applied.....	164
Figure 5-2: Stored procedure usage	165
Figure 5-3: Communications processing for an SQL stored procedure	166
Figure 5-4: Position of Java virtual machine in a HiRDB system.....	170
Figure 5-5: Flow from creation to execution of a Java stored routine.....	172

Figure 5-6: Overview of triggers.....	174
Figure 5-7: Example of referenced table and referencing table.....	178
Figure 5-8: Example of the action when update SQL code is executed for a referenced table (with CASCADE specified).....	179
Figure 5-9: Example of the action when update SQL code is executed for a referencing table (with CASCADE specified).....	180
Figure 5-10: Example of the action when update SQL code is executed for a referenced table (with RESTRICT specified).....	181
Figure 5-11: Overview of block transfer facility.....	187
Figure 5-12: Example of a search using a list.....	198
Figure 6-1: Relationship between HiRDB file system areas and file system areas provided by the OS.....	206
Figure 6-2: Organization of system log files.....	210
Figure 6-3: Organization of synchronization point dump files.....	211
Figure 6-4: Organization of status files.....	213
Figure 6-5: Configuration of system files in a HiRDB/Single Server.....	215
Figure 6-6: Configuration of system files in a HiRDB/Parallel Server.....	216
Figure 6-7: Example configuration of HiRDB system definition files (HiRDB/Single Server).....	221
Figure 6-8: Example configuration of HiRDB system definition files (HiRDB/Parallel Server).....	223
Figure 6-9: Example configuration of HiRDB system definition files (when the HiRDB External Data Access facility is being used).....	224
Figure 6-10: Concept of delayed rerun.....	231
Figure 6-11: Concept of global buffers.....	235
Figure 6-12: Overview of the snapshot method.....	240
Figure 6-13: Overview of global buffer pre-writing.....	241
Figure 6-14: Overview of local buffers.....	243
Figure 6-15: Overview of the BLOB data file output facility.....	245
Figure 6-16: Example of a retrieval using the BLOB data file output facility (retrieval of BLOB columns).....	247
Figure 6-17: Example of a retrieval using the BLOB data file output facility (retrieval of an abstract data type with the BLOB attribute).....	248
Figure 6-18: Example of BLOB data addition updating.....	249
Figure 6-19: Example of BLOB data partial extraction.....	250
Figure 6-20: Overview of the locator facility.....	252
Figure 6-21: Examples of transaction startup and termination.....	255
Figure 6-22: Processing for one-phase commit.....	257
Figure 6-23: Processing for two-phase commit.....	258
Figure 6-24: Locked resources and their inclusion relationships.....	261
Figure 7-1: Overview of database recovery process.....	270
Figure 7-2: Transactions that are recovered (with recovery to the most recent synchronization point before the failure occurred).....	271
Figure 7-3: System log file status changes.....	274

Figure 7-4: Relationship between backup and unload log files.....	276
Figure 7-5: Overview of the differential backup facility.....	278
Figure 7-6: Overview of frozen update command processing.....	281
Figure 7-7: How to use the frozen update command to make backups.....	282
Figure 7-8: System configuration example of the NetBackup linkage facility.....	283
Figure 7-9: Table reorganization.....	285
Figure 7-10: Reorganization of an entire table.....	286
Figure 7-11: Reorganization of an RDAREA.....	287
Figure 7-12: Reorganization of a schema.....	288
Figure 7-13: Overview of facility for predicting reorganization time.....	290
Figure 7-14: Index reorganization.....	292
Figure 7-15: Releasing used free pages.....	294
Figure 7-16: Process of used free pages being created for index pages.....	297
Figure 7-17: Releasing used free segments.....	298
Figure 7-18: RDAREA automatic extension.....	300
Figure 7-19: Moving RDAREAs.....	301
Figure 7-20: Level 1 processing.....	304
Figure 7-21: Level 3 processing.....	305
Figure 8-1: Overview of the system switchover facility (standby system switchover facility).....	311
Figure 8-2: Overview of the standby-less system switchover (1:1) facility.....	312
Figure 8-3: Overview of standby-less system switchover (effects distributed) facility (with distribution alternates and multi-stage system switchover).....	315
Figure 8-4: One-to-one system switchover configuration.....	319
Figure 8-5: Two-to-one system switchover configuration.....	320
Figure 8-6: Mutual system switchover configuration.....	320
Figure 8-7: System configuration example of a mutual alternating configuration.....	321
Figure 8-8: System configuration example of a one-way alternating configuration (2-node configuration).....	322
Figure 8-9: System configuration example of the standby-less system switchover (effects distributed) facility.....	323
Figure 8-10: System configuration example of a mixed standby-less (1:1) and standby type setup.....	324
Figure 8-11: Comparison of system switchover times.....	326
Figure 8-12: Operation with and without using a recovery-unnecessary front-end server.....	327
Figure 8-13: Configuration example of a system that uses a recovery-unnecessary front-end server.....	328
Figure 9-1: HiRDB user privileges.....	330
Figure 9-2: Overview of the security audit facility.....	334
Figure 9-3: Accessing audit trails.....	340
Figure 10-1: Application of plug-ins to a job.....	352
Figure 10-2: Delayed batch creation of plug-in index.....	360
Figure 11-1: Comparison between 32- and 64-bit-mode HiRDBs.....	362
Figure D-1: Partitioning key index: Single-column partitioning.....	415

Figure D-2: Partitioning key index: Multicolumn partitioning.....	416
Figure D-3: Example of a table containing repetition columns	423

List of tables

Table 2-1: Products needed on the HiRDB side to use the HiRDB External Data Access facility.....	28
Table 2-2: Products that can be linked to HiRDB by means of the replication facility.....	34
Table 2-3: Functions provided by the HiRDB XA library.....	43
Table 3-1: Types of RDAREAs	62
Table 3-2: Server machines on which RDAREAs can be created.....	65
Table 3-3: Available data types.....	70
Table 3-4: Types of hash partitioning	80
Table 3-5: Executability of operations on falsification-prevented tables	93
Table 3-6: Guidelines for row-partitioning an index: HiRDB/Single Server	104
Table 3-7: Guidelines for row-partitioning an index: HiRDB/Parallel Server).....	105
Table 3-8: Encapsulation levels and access types.....	133
Table 4-1: Types of pages	139
Table 4-2: Segment statuses.....	140
Table 4-3: Page statuses	146
Table 5-1: Operations that cannot be performed on a check pending status table.....	185
Table 5-2: SQL optimization specification facilities	192
Table 5-3: Facilities of SQL optimization options.....	192
Table 5-4: Facilities of SQL extension optimizing options	195
Table 6-1: Advantages and disadvantages of the environment setup methods.....	204
Table 6-2: Type of HiRDB file system area.....	207
Table 6-3: Maximum size of a HiRDB file system area.....	208
Table 6-4: System file components.....	214
Table 6-5: HiRDB system definitions (HiRDB/Single Server)	220
Table 6-6: HiRDB system definitions (HiRDB/Parallel Server).....	221
Table 6-7: HiRDB startup modes.....	226
Table 6-8: HiRDB termination modes	228
Table 6-9: Types of global buffers	233
Table 6-10: Types of local buffers	242
Table 6-11: Commitment control on a HiRDB/Parallel Server	259
Table 6-12: Database update log acquisition modes.....	265
Table 6-13: Methods of specifying the database update log acquisition mode	265
Table 6-14: Relationship between the specification value in the RECOVERY operand and PDDBLOG operand or the -1 option, and the value that is assumed during execution of the UAP (or the utility)	266
Table 6-15: Processing performed by HiRDB and the action to be taken by the user in the event of abnormal termination of the UAP.....	267
Table 6-16: Times for recovering the database	267
Table 7-1: Backup units	272
Table 7-2: Backup acquisition modes.....	273

Table 7-3: Types of backup-hold.....	280
Table 7-4: Benefits gained from releasing used free pages of a table.....	295
Table 7-5: Benefits gained from releasing used free space of an index.....	296
Table 7-6: Space conversion levels.....	303
Table 7-7: Specification of the sign part of signed packed format.....	306
Table 7-8: Rules for conversion of the sign part of signed packed format (other than "0" data).....	306
Table 7-9: Rules for conversion of the sign part of signed packed format ("0" data).....	307
Table 8-1: Functional differences between the monitor mode and the server mode.....	316
Table 8-2: Products needed to operate in the server mode.....	318
Table 9-1: Access privilege types.....	333
Table 9-2: Information collected in audit trails.....	337
Table 9-3: Audit events.....	341
Table 9-4: Overview of connection security facility.....	345
Table 9-5: Restrictions that can be specified for passwords.....	346
Table 10-1: HiRDB plug-ins.....	350
Table 10-2: RDAREAs for storing information related to registry facility.....	359
Table 11-1: Applicability of related products in 64-bit-mode HiRDB.....	363
Table 11-2: Operands with higher maximum value specifications (HiRDB system definition).....	365
Table 11-3: Operands with higher maximum value specifications (client environment definition).....	366
Table 11-4: 64-bit-mode HiRDB client support range.....	367
Table 11-5: Operands with different default values.....	368
Table 11-6: Changes in linkage areas.....	369
Table 11-7: Changes in descriptor areas.....	369
Table 11-8: Changed C-language data descriptions.....	370
Table A-1: Functional differences between HiRDB versions on different platforms.....	374
Table B-1: Data dictionary tables.....	379
Table C-1: HiRDB client and HiRDB server connectivity (HiRDB server Version 5.0 or later).....	383
Table C-2: HiRDB client and HiRDB server connectivity (HiRDB server Version 4.0 or older).....	385
Table D-1: Types of backup-hold.....	390

Chapter

1. Overview

This chapter explains the characteristics, system configuration, and access modes of HiRDB. This chapter also provides an overview of HiRDB option program products and other HiRDB related products.

- 1.1 Characteristics of HiRDB
- 1.2 HiRDB system configuration
- 1.3 Database access modes

1.1 Characteristics of HiRDB

HiRDB is a database management system (DBMS) that enables you to construct relational databases appropriate to the scale of your operations.

Interlinking enables HiRDB to connect independently-operating server machines. HiRDB offers a high degree of flexibility, thanks to the adoption of the Shared Nothing method, in which a processor works exclusively on a database residing on a single disk. HiRDB can be adapted to a broad range of architectures, from a single-node configuration in which HiRDB runs on a single server machine, to a parallel processor configuration in which HiRDB runs on multiple server machines. HiRDB also makes it possible to expand your database system by adding server machines subsequently, which enables scalable system construction.

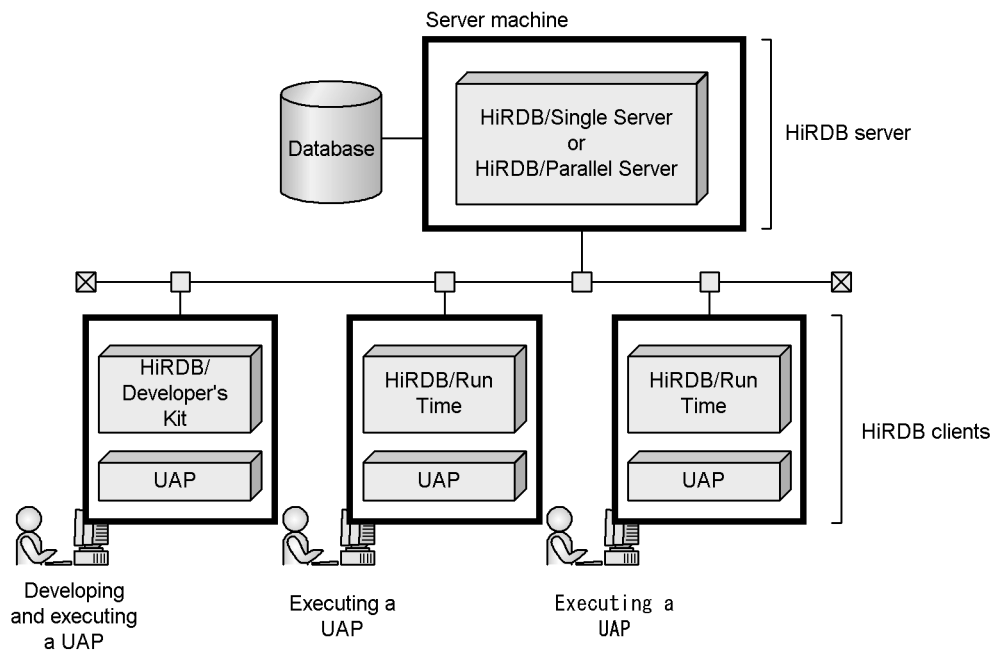
When HiRDB is configured for parallel processors, it is capable of executing data retrievals and update requests internally and in parallel, thus achieving high throughput and fast turnaround time.

Additionally, when used together with a middleware application suite (DataStage, HITSENSER, DATAFRONT, or other suite), HiRDB supports the building of database warehouses, which are needed for implementing leading-edge management systems.

1.1.1 Overview of HiRDB systems

HiRDB is used in a network environment consisting of client/server systems. A server system in which a database is installed is called a *HiRDB server*; a client system in which UAPs are developed and executed is called a *HiRDB client*. HiRDB servers and HiRDB clients are referred to as *HiRDB systems*. Figure 1-1 shows the configuration of a HiRDB system.

Figure 1-1: Configuration of a HiRDB system



(1) HiRDB servers

The two types of HiRDB servers are the HiRDB/Single Server and the HiRDB/Parallel Server. You must select one of these two types, as appropriate to your system mode or the types of operations you will be performing.

A HiRDB server runs on one of the following platforms:

- HP-UX
- Solaris
- AIX 5L
- Linux
- Windows 2000
- Windows XP Professional
- Windows Server 2003

This manual describes the Windows version of HiRDB (which includes Windows 2000, Windows XP Professional, and Windows Server 2003).

(a) HiRDB/Single Server

You would use a HiRDB/Single Server for a database system that consists of a single server machine. Because of its stable processing performance and simple operations as compared with a HiRDB Parallel Server, a HiRDB/Single Server is well-suited for small and medium-sized databases.

(b) HiRDB Parallel Server

A HiRDB/Parallel Server makes it possible to link multiple server machines into a single database system so that a table can be split up and stored on multiple server machines. Because it permits different servers to perform retrieval tasks in parallel, a HiRDB Parallel Server provides improved performance. In addition, its capability to add or update large amounts of data or back up a database concurrently means that a HiRDB Parallel Server is able to maintain high performance even when the database becomes very large.

The Shared Nothing method, which fully exploits the hardware's capabilities, enables you to maintain stable performance by increasing the number of servers when the volume of data to be handled increases. In addition, a HiRDB Parallel Server can achieve well-balanced parallel processing in a manner that prevents concentration of processing on a specific server by delegating sorting, joining, and similar large-overhead tasks to other servers that have smaller workloads.

(2) HiRDB clients

The two types of HiRDB clients are the HiRDB/Developer's Kit and HiRDB/Run Time.

A HiRDB/client runs on one of the following platforms:

- HP-UX
- Solaris
- AIX 5L
- Linux
- Linux for AP8000
- Windows 2000
- Windows XP Professional
- Windows XP Home Edition
- Windows Server 2003

(a) HiRDB/Developer's Kit

HiRDB/Developer's Kit is a program for developing (pre-processing, compiling, linking) and executing UAPs. As the development language for UAPs, you can use C,

C++, COBOL85, OOCOBOL, COBOL 2002, or Java™ (SQLJ).

Because a HiRDB server includes the HiRDB/Developer's Kit functions, a HiRDB/Developer's Kit is not necessary for developing or executing a UAP on a HiRDB server; a HiRDB/Developer's Kit is necessary, however, to develop or execute a UAP on a client.

Hint:

Execute the UAP with a HiRDB/Developer's Kit that is running on the same platform as the HiRDB/Developer's Kit that was used to develop the UAP.

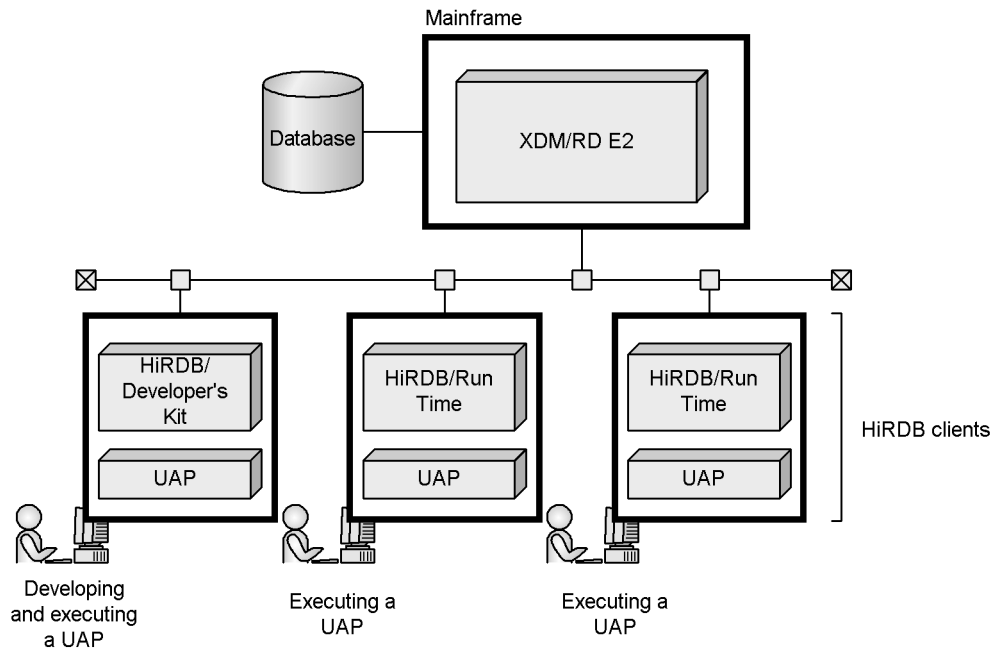
(b) HiRDB/Run Time

HiRDB/Run Time is a run-time program that is used exclusively for executing previously created UAPs. HiRDB/Run Time cannot be used to develop (preprocess, compile, link) a UAP; it can only be used to execute a UAP.

(c) Connecting from a HiRDB client to XDM/RD E2

At a HiRDB client, you can develop and execute UAPs that access XDM/RD E2 databases. Such UAPs can be used to access XDM/RD E2 databases from HiRDB clients. The function that performs this type of access is called the XDM/RD E2 connection facility. Figure 1-2 provides an overview of the XDM/RD E2 connection facility.

Figure 1-2: Overview of the XDM/RD E2 connection facility



Explanation

Using the XDM/RD E2 connection facility, you can directly access XDM/RD E2 databases from UAPs that are running on HiRDB clients.

Although ODBC or other functions can also be used to access XDM/RD E2 from a UAP running on a PC, there are limitations imposed by the application language. Using the XDM/RD E2 connection facility enables you to perform a wider variety of processing by coding SQL statements directly in the UAP, which also improves UAP development efficiency.

1.1.2 Advantages of using HiRDB

HiRDB is a relational database that incorporates both concurrent batch updating and parallel recovery techniques. The characteristics of HiRDB are discussed below.

(1) Excellent scalability

The two types of HiRDB are the HiRDB/Single Server that operates on a single server machine, and the HiRDB/Parallel Server that operates on two or more server machines. You can select either a HiRDB/Single Server or a HiRDB Parallel Server, depending on the scope of the operations to which HiRDB will be applied. You can change a HiRDB/Single Server into a HiRDB/Parallel Server, or you can increase the number of server machines used in a HiRDB/Parallel Server. Thus, you can expand

your system gradually as the scale of your operations increases.

The Shared Nothing method that is incorporated into HiRDB is well-suited for parallel processing and enables you to increase the processing capacity of HiRDB in proportion to the number of processors that are used. In other words, you can increase processing capacity without having to add server machines. When the flexible hash partitioning function is used and a server machine is added, HiRDB modifies the hash function automatically, so that the data is stored automatically in the new server machine.

(2) Achieving a high-performance system

(a) Performance improvement through parallel processing

Concurrent retrieval and updating of tables

Database search and update operations can be distributed to multiple server machines, and table data can also be divided among the server machines. This feature permits concurrent searching and updating of tables, which improves performance in direct proportion to the number of server machines deployed.

Distribution of high-overhead database processing operations

HiRDB enables you to assign high-overhead operations, such as sorting and joining, to a separate server machine so that these operations can be performed concurrently with data accesses. This feature reduces the amount of time required to output search results.

Reduction in batch processing time for large amounts of data

HiRDB can reduce the amount of time required for processing because it can store masses of data concurrently, such as during system creation.

Concurrent database reorganization

HiRDB can reduce the amount of time required for database reorganization operations because it can concurrently reorganize databases by server.

Concurrent backup and recovery operations

In HiRDB, a single command can concurrently back up and recover a failed database. This feature reduces the amount of time required for backup and database recovery operations.

(b) Fine buffer control through the use of global buffers

HiRDB enables you to allocate index data and other frequently accessed data to special buffers. This feature assures stable response by eliminating interference between buffers even in an environment where different operations are performed on a mixed basis, such as index searches and complete data searches.

(c) Performance improvement by using synchronization point dump processing

In an operation called *synchronization point dump processing*, HiRDB stores update information up to a specific point in the database and recovery information up to that point in a file. Other systems stop accepting transactions during a synchronization point dump operation, which results in a decrease in processing performance. HiRDB, by contrast, does not limit the system's ability to accept transactions, so there is no decrease in processing performance attributable to synchronization point dump processing.

(d) Rapid system recovery by high-speed rerun

HiRDB reduces the range over which a recovery operation must be performed in the aftermath of a system failure by periodically collecting synchronization point dumps, which enables HiRDB to quickly complete recovery processing.

In addition, HiRDB performs during a system recovery operation a *delayed rerun operation* that simultaneously starts a rollback operation and acceptance of new transactions, which can speed up system restart.

(3) Achieving a highly reliable system

(a) Storing in files information needed for system restart and for database recovery

Logging information needing for system restart

During operation, HiRDB stores in files system status information that will be needed to restart the system in the event of a failure. These files are called *status files*.

Logging information needed for database recovery

HiRDB stores in files a history of database update information (a system log) that would be needed for recovery of the database during a recovery operation. The files in which this information is stored are called *system log files*. In the event of an error in the system, system log files make it possible for HiRDB to correctly recover the database to its status just before the error occurred.

Duplicate files

HiRDB maintains duplicate status files and system log files, thus providing redundant logging of the information it will need for system and database recovery. File redundancy permits the use of the other file if a problem arises in one of the files, thus increasing system reliability.

(b) Automatic system restart

In the event of a relatively minor error, HiRDB uses the status files to restart the system automatically, so there is no need for operator intervention.

(c) Reducing system downtime by use of system switchover facility

In HiRDB, you can set up a standby server machine that is separate from the currently operating server machine, so that if the current server machine fails, operations can be switched smoothly to the standby machine. This feature is called the *system switchover facility*.

Microsoft Cluster Server (MSCS) is used for the system switchover facility.

(4) Improved maintainability/operability**(a) Intensive control from a specific server machine**

In the case of a HiRDB/Parallel Server, a specified server machine can integrally and intensively control a HiRDB system running on multiple server machines. For example, you can execute a command or a utility on one server machine to start or terminate the HiRDB running on all server machines or on specified server machines.

(b) Support for setting up the HiRDB environment

Support tools are provided for setting up the HiRDB environment. The tools that are provided are listed and described below.

Simple setup tool

A graphic user interface (GUI) is provided for setting up HiRDB. Choosing **Standard Setup**, in which you simply specify the setup directory, enables you to easily configure an environment. You can also choose **Custom Setup** in order to specify more detailed settings. The simple setup tool can also be used to update or edit system definitions that you have already created.

Batch files

By executing the batch files, you can automatically set up the basic HiRDB environment.

(5) Implementing a flexible system in an open environment**(a) XA interface for X/Open**

HiRDB can communicate with OLTP by means of the XA interface of X/Open. A HiRDB XA library is provided so that HiRDB transactions can be controlled by the Transaction Manager.

(b) ODBC, JDBC, and OLE DB interfaces

HiRDB complies with the ODBC, JDBC, and OLE DB industry standards, which means that ODBC, JDBC, and OLE DB applications can be used in HiRDB. You can also use ADO (ADO.NET is also supported), DAO, and RDO.

(6) Support for non-stop service

With the growing popularity of Internet businesses, there is an increasing need to

1. Overview

conduct online operations non-stop, 24 hours per day, 365 days per year. HiRDB provides functionality based on the premise of continuous operation, 24 hours per day. For details about functionality based on 24-hour-per-day operation, see the *HiRDB Version 8 System Operation Guide*.

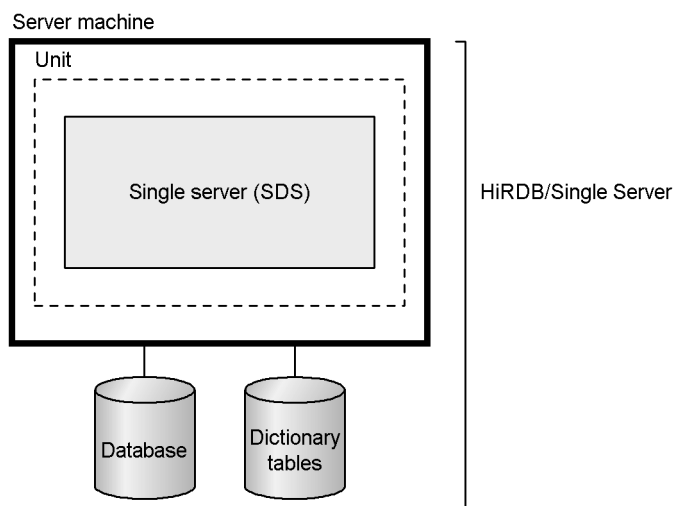
1.2 HiRDB system configuration

This section explains the configurations of a *HiRDB/Single Server*, a *HiRDB/Parallel Server*, and a *multi-HiRDB*.

1.2.1 HiRDB/Single Server configuration

A HiRDB/Single Server consists of one unit (one single server). Figure 1-3 shows the configuration of a HiRDB/Single Server.

Figure 1-3: HiRDB/Single Server configuration



(1) Unit

A HiRDB/Single Server is composed of the following server:

- Single server

The unit controls and monitors execution of the server. The unit can be compared conceptually to a container in which the server is stored.

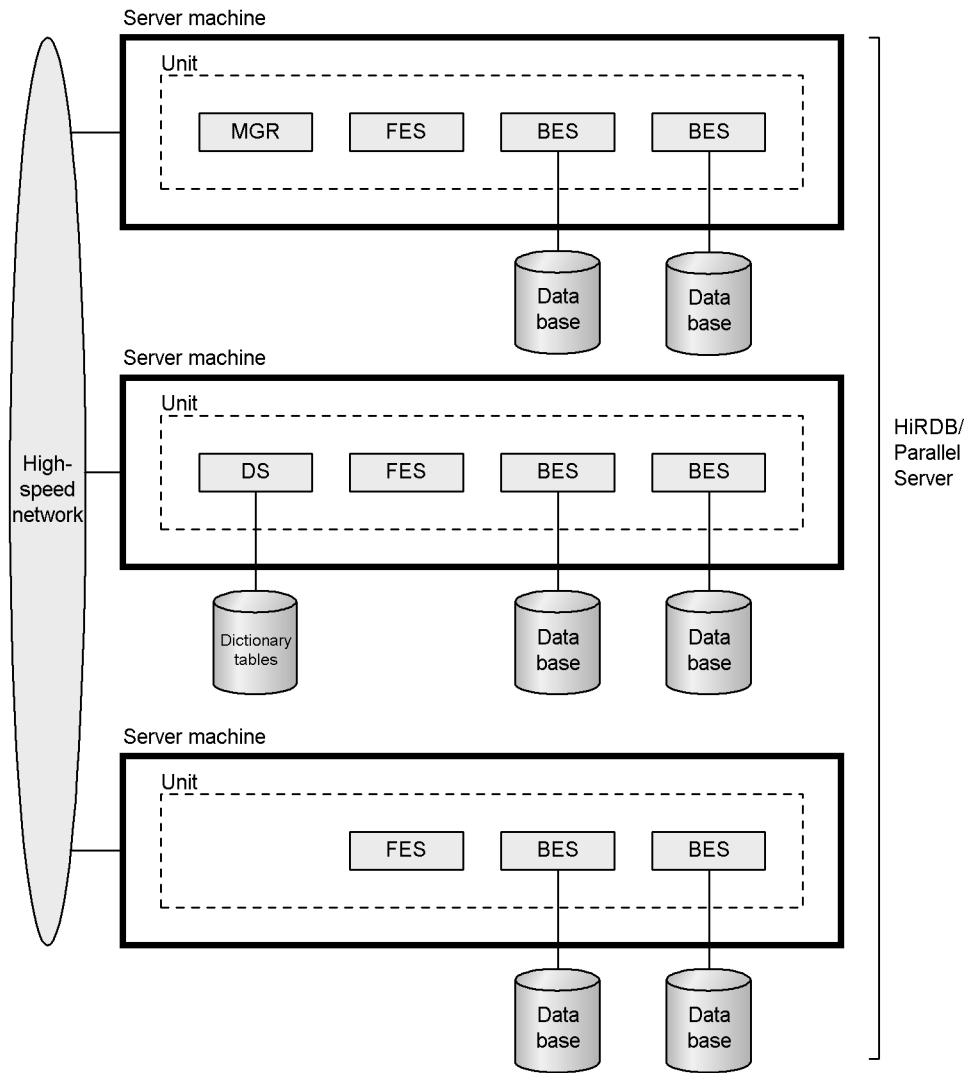
(2) Single server (SDS: Single Database Server)

The single server is the server that manages a database (tables and indexes) and the data dictionaries (dictionary tables) that contain information about the database.

1.2.2 HiRDB/Parallel Server configuration

A HiRDB/Parallel Server is composed of multiple units (multiple servers). Figure 1-4 shows an example of a HiRDB/Parallel Server configuration.

Figure 1-4: HiRDB/Parallel Server configuration



Explanation

- This HiRDB/Parallel Server consists of three server machines.
- In the configuration in this example there are multiple front-end servers.
- Two back-end servers are provided per server machine.

(1) Unit

A HiRDB/Parallel Server consists of the following types of servers:

- System manager
- Front-end server
- Dictionary server
- Back-end server

A unit controls and monitors server execution and manages communication between servers. A unit can be compared conceptually to a container in which servers are stored.

(2) System manager (MGR)

The system manager is the server that controls HiRDB startup and termination. It also manages system configuration information and detects server errors.

One system manager is required per system.

(3) Front-end server (FES)

A front-end server determines the procedure by which databases are accessed and provides directives to back-end servers on the contents of tasks that are to be executed. A front-end server also analyzes SQLs, optimizes SQLs, provides processing instructions to back-end servers, and edits search results.

Each system must have at least one front-end server (up to a maximum of 1024 front-end servers). A configuration in which there are multiple front-end servers is called a *multi-front-end server configuration*. When SQL processing results in a high CPU workload that exceeds the processing capability of a single front-end server, a multi-front-end server configuration is appropriate. A multi-front-end server configuration can distribute the processing load among the machines on which front-end servers are running.

(4) Dictionary server (DS)

The dictionary server provides batch management of the data dictionaries (dictionary tables) that contain database definition information.

Each system requires one dictionary server.

(5) Back-end server (BES)

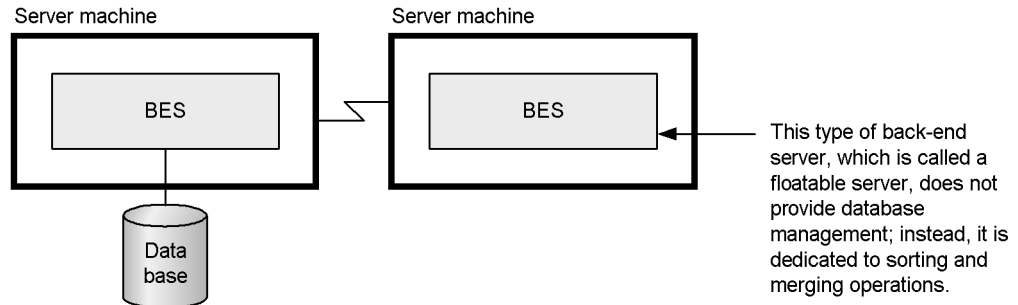
A back-end server manages the database. On the basis of execution directives received from front-end servers, the back-end servers access and lock the database and perform computational operations. Back-end servers also sort, merge, and join search results.

At least one back-end server is required per system (up to a maximum of 1024 back-end servers). When multiple back-end servers are provided, a table can be

divided among them so that it can be managed on a split basis.

Performance can be improved in a HiRDB/Parallel Server by providing back-end servers that do not provide database management but instead are dedicated specifically to processing high-overhead operations, such as sorting and joining. Such back-end servers are called *floatable servers*. Figure 1-5 shows an example of a floatable server.

Figure 1-5: Floatable server



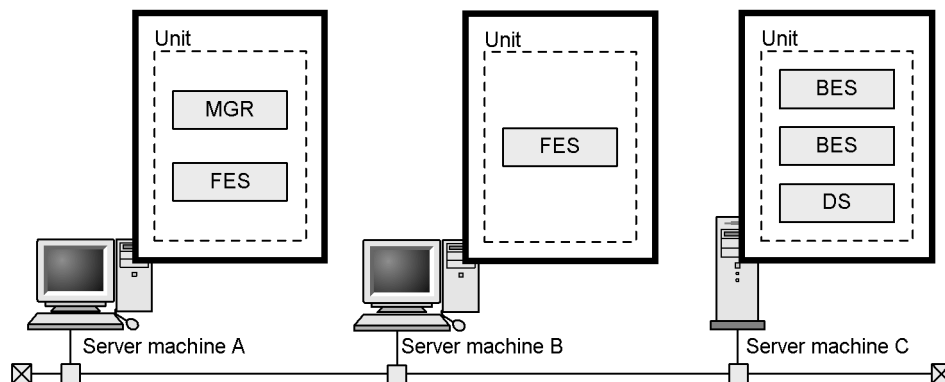
(6) Heterogeneous system configuration of HiRDB/Parallel Server

Normally, all HiRDB/Parallel Server units must be running on the same platform. However, if the following conditions are satisfied, you can set up a HiRDB/Parallel Server in what is called a *heterogeneous system configuration* in which some units run on different platforms:

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

Figure 1-6 shows an example of a heterogeneous system configuration.

Figure 1-6: Example of HiRDB/Parallel Server in a heterogeneous system configuration



Explanation

- This HiRDB/Parallel Server configuration consists of three server machines. Server machines A and B are used as application servers, and server machine C is used as a database server.
- The platform for server machines A and B is Windows (x64), while the platform for server machine C is Windows Server 2003 (IPF).

1.2.3 Multi-HiRDB configuration

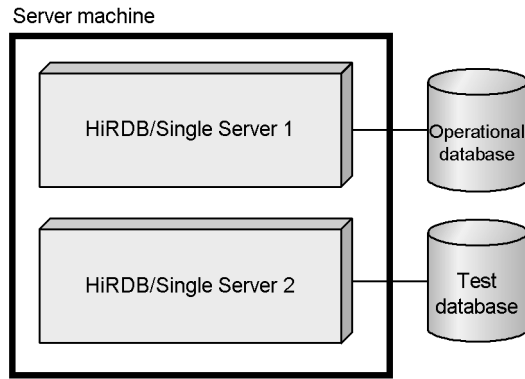
You can install multiple, separately operating HiRDB systems on one server machine. Such a system configuration is called a *multi-HiRDB* configuration. A multi-HiRDB configuration would be appropriate under the following circumstances:

- Running an operational system and a test system on the same server machine
- Running systems for different applications on the same server machine

Note, however, that you cannot combine HiRDB/Single Servers and HiRDB Parallel Servers in a multi-HiRDB system.

Figure 1-7 shows an example of a multi-HiRDB configuration implemented on a HiRDB/Single Server.

Figure 1-7: Multi-HiRDB configuration using HiRDB/Single Servers



Explanation

This is a multi-HiRDB configuration implemented on a HiRDB/Single Server. HiRDB/Single Server 1 is designated as the operational system, and HiRDB/Single Server 2 serves as a test system.

1.3 Database access modes

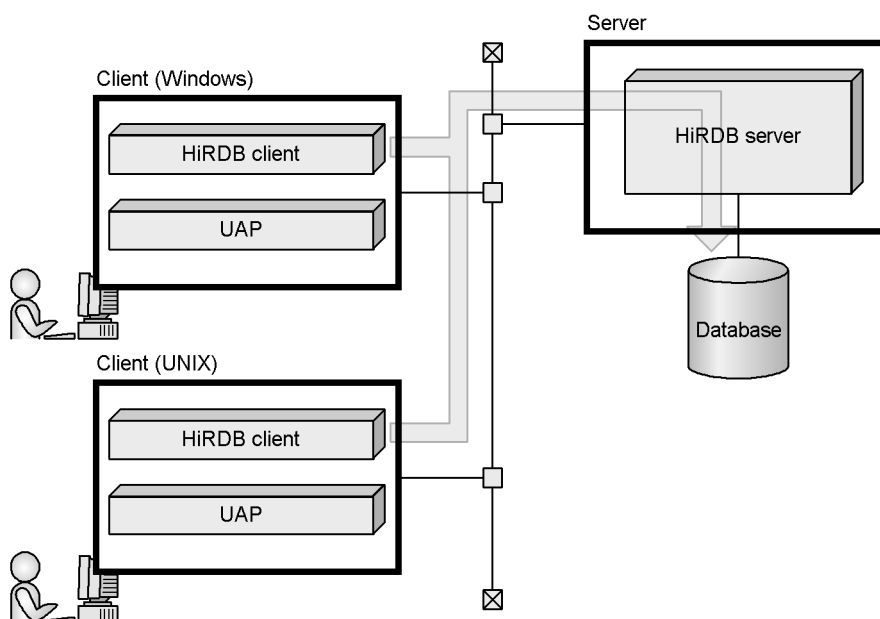
You access a HiRDB database by means of the SQL that is provided by HiRDB. This section explains the operating modes under which an SQL can be executed from a UAP.

(1) Accessing a database by executing a UAP from a HiRDB client

This is the basic mode of operation. In it, you access a database by executing a UAP on a HiRDB client.

Note however, that the HiRDB server platforms to which you can connect are limited, depending on the version of the HiRDB client. For details, see *Appendix C. HiRDB Client and HiRDB Server Connectivity*. Figure 1-8 shows a mode of operation in which a database is accessed by executing UAPs running on HiRDB clients.

Figure 1-8: Accessing a database by executing UAPs from HiRDB clients

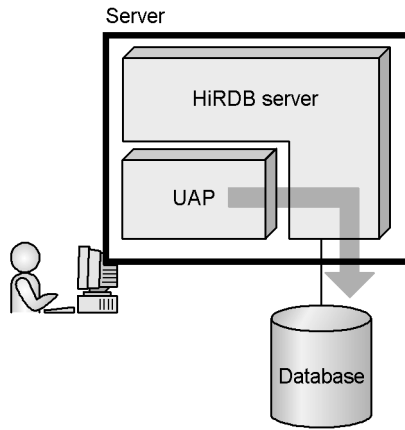


(2) Accessing a database by executing a UAP running on the HiRDB server

In this mode, you can access a database by executing a UAP running on the HiRDB server.

Figure 1-9 illustrates the operating mode in which a database is accessed by executing a UAP running on the HiRDB server.

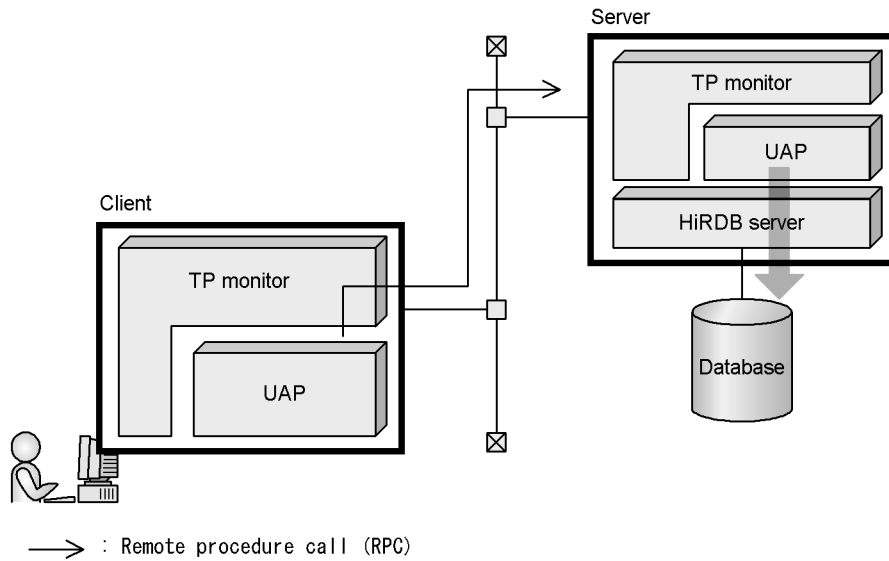
Figure 1-9: Accessing a database by executing a UAP running on the HiRDB server



(3) Accessing a database by executing an OLTP UAP

In this mode, a HiRDB database is accessed when a UAP running on an OLTP machine (TP monitor) issues a service request. The client must have a service-requesting UAP; the server must have a service-providing UAP. Figure 1-10 illustrates the operating mode in which a database is accessed by executing an OLTP UAP.

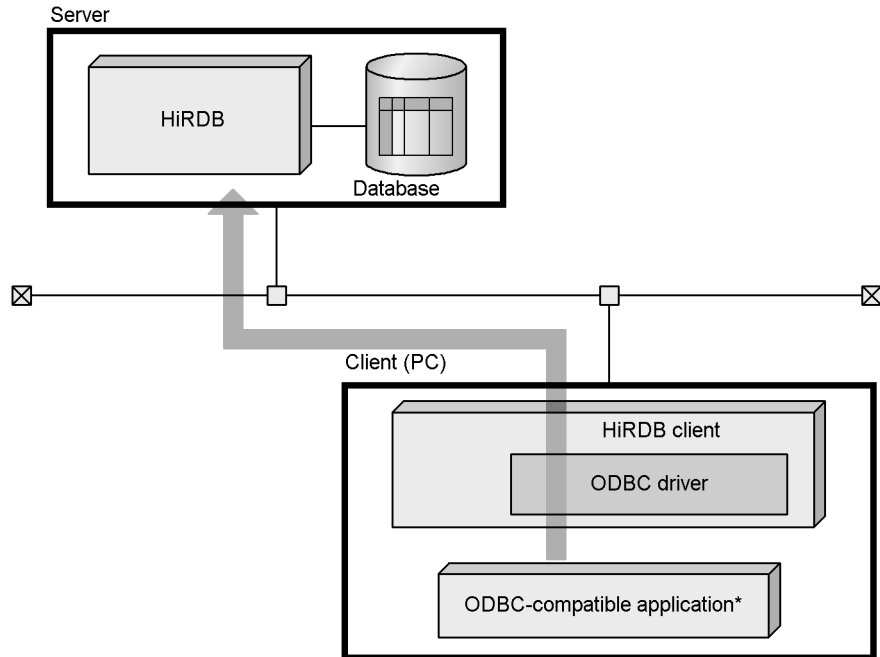
Figure 1-10: Accessing a database by executing an OLTP UAP



(4) Linkage to ODBC-compatible applications

HiRDB provides an ODBC driver. When the ODBC driver is installed in a HiRDB client, ODBC-compatible applications can access the HiRDB database. ODBC-compatible applications include many commercially available programs, such as Microsoft Access and PowerBuilder. UAPs that use the ODBC function provided by HiRDB can also access the HiRDB database. Figure 1-11 shows a situation in which an ODBC-compatible application accesses a HiRDB database.

Figure 1-11: Accessing a HiRDB database from an ODBC-compatible application



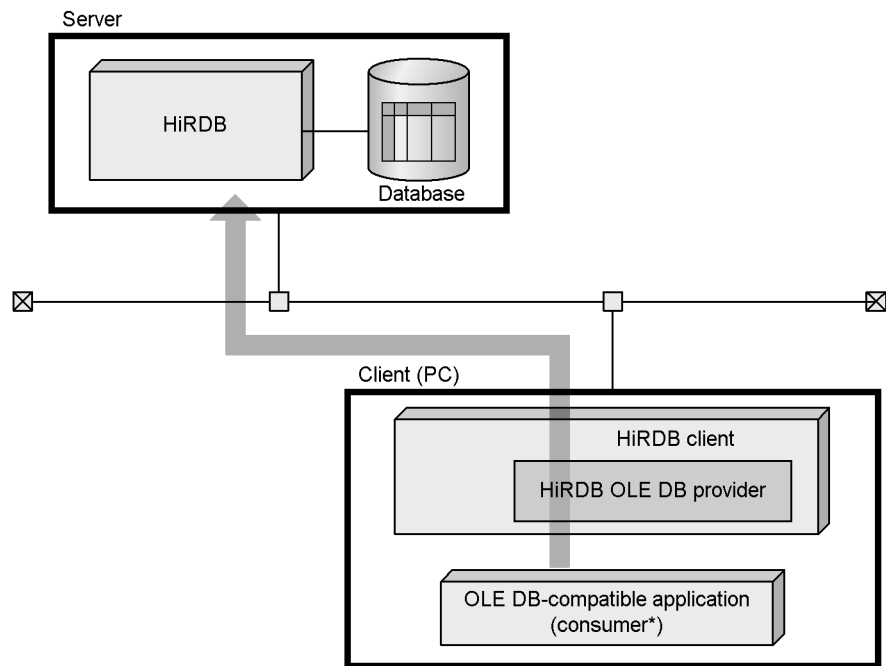
* Includes ODBC-compatible UAPs.

For details about accessing a HiRDB database from an ODBC-compatible application, see the *HiRDB Version 8 UAP Development Guide*.

(5) Linkage to OLE DB-compatible applications

HiRDB provides an OLE DB provider. By selecting the OLE DB provider when you install a HiRDB client, you can enable applications that support the OLE DB to access HiRDB databases. Figure 1-12 shows a situation in which a HiRDB database is accessed from an OLE DB-compatible application.

Figure 1-12: Accessing a HiRDB database from an OLE DB-compatible application



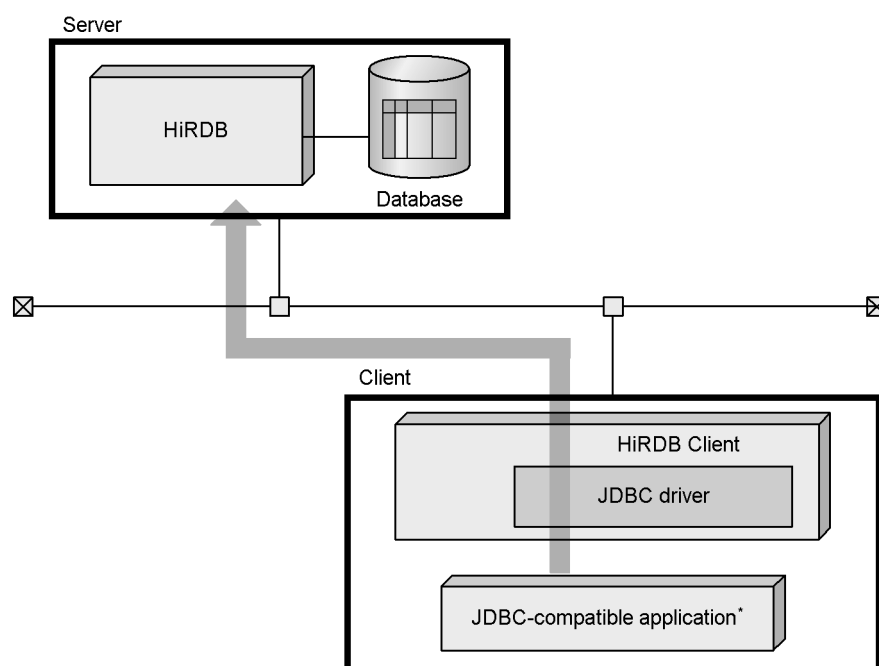
* Consumer refers to a program that calls an OLE DB method and interface.

For details about accessing a HiRDB application from an OLE DB-compatible application, see the *HiRDB Version 8 UAP Development Guide*.

(6) Accessing a database from a JDBC-compatible application

HiRDB provides a JDBC driver. By selecting the JDBC driver when you install a HiRDB client, you can enable applications that support JDBC to access HiRDB databases. Figure 1-13 shows a situation in which a HiRDB database is being accessed from a JDBC-compatible application.

Figure 1-13: Accessing a HiRDB database from a JDBC-compatible application



*Also includes programs that call Java stored procedures and Java stored functions.

For details about accessing HiRDB from a JDBC-compatible application (using Java stored procedures and Java stored functions), see the *HiRDB Version 8 UAP Development Guide*.

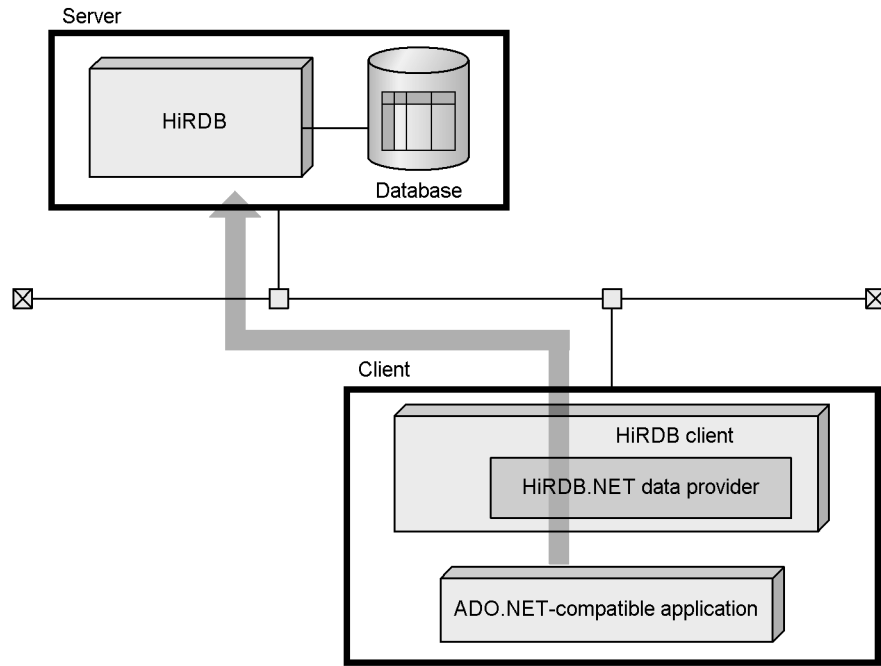
(7) Accessing a database from an ADO.NET-compatible application

HiRDB uses ADO.NET to provide the HiRDB.NET data provider, which is needed to access HiRDB. The HiRDB.NET data provider complies with the ADO.NET specifications. By selecting the HiRDB.NET data provider when you install a HiRDB client, you can enable applications that support ADO.NET to access HiRDB databases.

HiRDB.NET data provider is bundled with a common set of basic interfaces provided by the System.Data space of Net Framework. HiRDB.NET data provider also includes its own extensions to perform array insertion and access repetition columns.

Figure 1-14 shows an example in which a HiRDB database is accessed from an ADO.NET-compatible application.

Figure 1-14: Accessing a database from an ADO.NET-compliant application



For details about accessing HiRDB from an ADO.NET-compliant application, see the *HiRDB Version 8 UAP Development Guide*.

Chapter

2. Linking to HiRDB Option Program Products and Other HiRDB-Related Products

This chapter describes the functionality enabled by linking HiRDB to HiRDB option program products and other HiRDB-related products.

- 2.1 HiRDB option program products
- 2.2 Linkage to data linkage products
- 2.3 Linkage to directory server products
- 2.4 Linkage to OLTP products
- 2.5 Linkage to operation support products
- 2.6 Linkage to data mining products
- 2.7 Linkage to products that handle multimedia information
- 2.8 Linkage to Cosminexus

2.1 HiRDB option program products

This section describes the functions and operations that you can enable by using the following HiRDB option program products:

- HiRDB Advanced High Availability
- HiRDB Advanced Partitioning Option
- HiRDB LDAP Option
- HiRDB External Data Access
- HiRDB Non Recover FES

2.1.1 HiRDB Advanced High Availability

Installing HiRDB Advanced High Availability allows you to use the following functions:

- Standby-less system switchover facility
- System reconfiguration command (`pdchgconf` command)
- Dynamic updating of global buffers (`pdbufmod` command)

The *standby-less system switchover facility* implements a system switchover method that enables HiRDB to continue providing services when a failure occurs by transferring processing to another unit. Because the standby-less system switchover facility does not require that standby resources be placed in reserve, it is significantly more economical than the previously available system switchover facility (the *standby system switchover facility*). With the previous system switchover facility, you had to set aside a server, CPU, and memory resources to be available for transfer of operations in the event of a failure. With the standby-less system switchover facility, there is no need to provide these redundant resources. You simply register another server as the alternate, and that server's unit takes over processing if a failure occurs. Although the processing performance of the unit that has taken over may be degraded, the overall cost of the system is reduced because you can make more effective use of resources.

The standby-less system switchover facility includes the *standby-less system switchover (1:1) facility* and the *standby-less system switchover (effects distributed) facility*. For details about the standby-less system switchover facility, see *8.1 System switchover facility*.

Previously, changing HiRDB system definitions meant having to stop HiRDB; however, by using the *system reconfiguration command* (`pdchgconf` command), you can change HiRDB system definitions without having to stop HiRDB. This allows you to perform system-related operations while HiRDB is running, such as changing the configuration of units or servers and adding system files. For details about the system

reconfiguration command, see *6.5.4 System reconfiguration command (pdchgconf command)*.

Previously, adding, changing, or removing global buffers required you to change the `pdbuffer` operand in the HiRDB system definitions, which meant you had to stop HiRDB. Using the `pdbufmod` command, however, allows you to add, change, and remove global buffers while HiRDB is running. This is called *dynamic updating of global buffers*. For details about dynamic updating of global buffers, see *6.8.1(3) Dynamic updating of global buffers*.

2.1.2 HiRDB Advanced Partitioning Option

Installing HiRDB Advanced Partitioning Option enables you to use the following functions:

- Table matrix partitioning

Table matrix partitioning is a function that performs key range partitioning by using multiple columns as keys. The use of multiple columns as partitioning keys improves the performance of SQL code that is executed in parallel, and the use of multiple keys for searches accelerates processing by narrowing the searched range. In addition, partitioning database storage areas (RDAREAs) into smaller segments also reduces the time it takes to reorganize the database, to make backups, and to recover the database. For details about table matrix partitioning, see *3.3.9 Table matrix partitioning*.

- Changing the partitioning storage conditions

Using `ALTER TABLE`, you can change the partitioning storage conditions of tables that have been row-partitioned by key range partitioning. * Changing a table's partitioning storage conditions enables you to reuse RDAREAs that contain outdated data and reduces work time. For details about changing partitioning storage conditions, see *3.3.10 Changing the partitioning storage conditions of a table*.

*

`ALTER TABLE` can be used to change a table's partitioning storage condition when one of the following partitioning methods was used:

- Boundary value specification
- Storage condition specification (if the equal sign (=) was used as the comparison operator for the storage condition)

2.1.3 HiRDB LDAP Option

Installing HiRDB LDAP Option allows you to link to the Sun Java System Directory Server and manage user IDs and passwords of users connected to HiRDB under the Sun Java System Directory Server. As an LDAP-compliant directory server, the Sun

Java System Directory Server is a product that centrally manages information about users, machines, and other resources on a network. Linking HiRDB to a Sun Java System Directory Server enables users utilizing Web applications, groupware, and other products to be managed together with their user information, thus reducing the workload on system administrators. For details about linking to the Sun Java System Directory Server, see *2.3 Linkage to directory server products*.

HiRDB High Availability runs on Windows 2000.

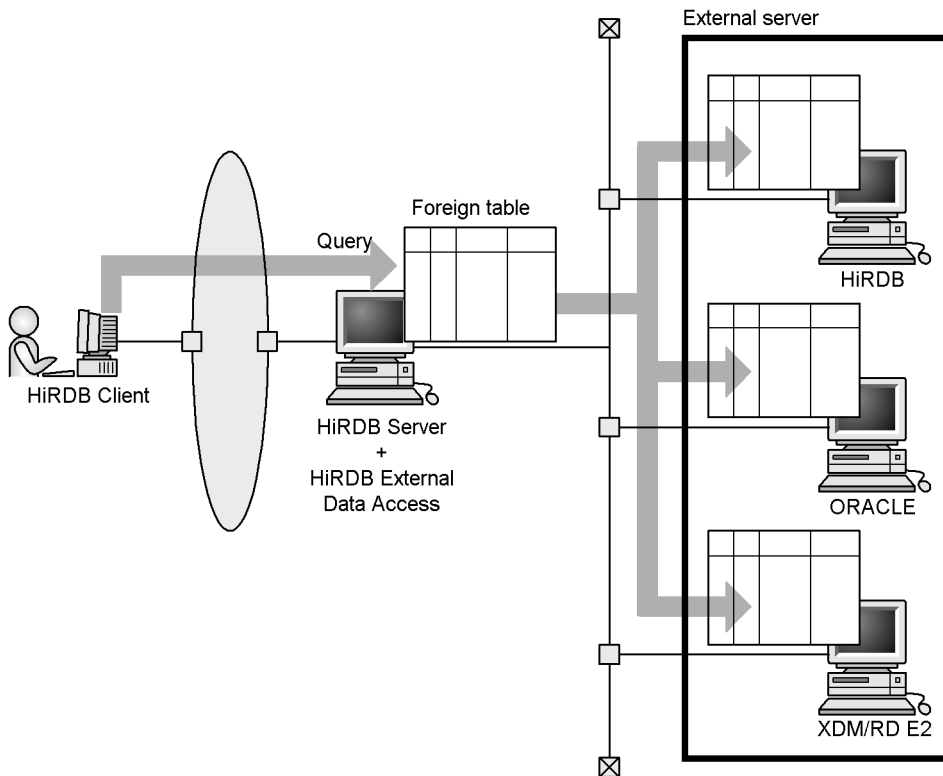
2.1.4 HiRDB External Data Access

Installing HiRDB External Data Access allows you to use the *HiRDB External Data Access facility*. The HiRDB External Data Access facility can be used with the 32-bit version of HiRDB.

(1) Overview of the HiRDB External Data Access facility

The HiRDB External Data Access facility allows you to access, through the HiRDB interface, multiple database tables built by DBMSs on various types of machines, including those built with third party products. It even allows you to view and update in a single table information stored in different types of databases (if you are using a foreign table or a view table). This facility is called the *HiRDB External Data Access facility*. Figure 2-1 provides an overview of the HiRDB External Data Access facility. For details about the HiRDB External Data Access facility, see the *HiRDB External Data Access Version 7 Description and User's Guide*.

Figure 2-1: Overview of the HiRDB External Data Access facility



Explanation

You can view and update external DBMS databases from a HiRDB client. An external DBMS is called a *foreign server*.

A table that can be viewed or updated with the HiRDB External Data Access facility is called a *foreign table*. A foreign table is a table defined on HiRDB based on the definition information for a table on a foreign server. A foreign table is required in order to view or update tables in foreign servers. Note that HiRDB manages only the definition information for the table; the foreign server manages the actual content of the table.

HiRDB External Data Access provides the following features:

- Ability to view and update in a single table database information derived from multiple DBMSs in different locations
- Ability to access tables using only the HiRDB interface, even in environments running multiple DBMSs

You can expect the following benefits when you use HiRDB External Data Access:

- Reduction in development times and operation costs through the use of legacy assets
- Flexible support for modification and expansion of database environments
- Simplification of application development through use of the HiRDB interface (SQL interface)
- Improvement in the ability to efficiently deploy information resources through realization of on-demand access
- Ability to provide the latest information available

(2) Supported DBMSs

The following lists the DBMSs that HiRDB can connect to as a foreign server:

- HiRDB (HiRDB Version 5.0 05-06 or later)*
- XDM/RD E2 06-00 or later
- ORACLE (Oracle 8i 8.1.7)
- ORACLE (Oracle 9i 9.2)
- ORACLE (Oracle 10g 10.1)

*

- An instance of HiRDB connected to as a foreign server is called a *foreign HiRDB*.
- Both HiRDB/Single Server and HiRDB/Parallel Server can be used as foreign servers.
- Neither the Linux version nor the Windows version of HiRDB version 5.0 can be used as a foreign server.

(3) System configuration example

You must have a back-end server to connect to a foreign server (in order to view or update a foreign table). Such a back-end server is referred to as a *back-end server for connecting to foreign servers* (or a *back-end server for viewing or updating foreign tables*). On the HiRDB side, you also need the products shown in Table 2-1.

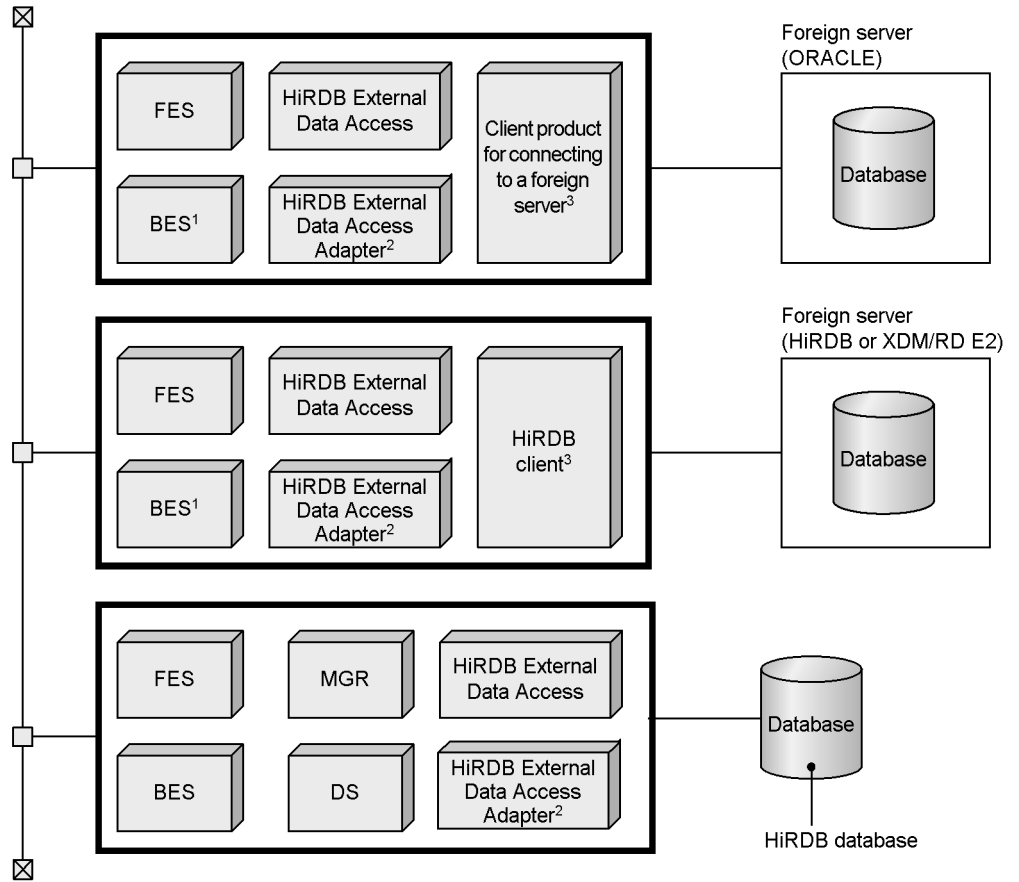
Table 2-1: Products needed on the HiRDB side to use the HiRDB External Data Access facility

Product name	Description
HiRDB External Data Access	Needed to execute the HiRDB External Data Access facility.

Product name	Description
HiRDB External Data Access Adapter	Handles the interface differences between HiRDB and the connection-target DBMS. A HiRDB External Data Access Adapter is available for ORACLE . Because HiRDB Data Access Adapter is included in HiRDB External Data Access, you do not need a separate HiRDB External Data Access Adapter when the connection-target DBMS is either HiRDB or XDM/RD E2.
ORACLE client product	Needed when the connection-target DBMS is ORACLE.

Figure 2-2 shows an example of a system configuration when the HiRDB External Data Access facility is being used.

Figure 2-2: System configuration example when the HiRDB External Data Access facility is being used



1

Back-end server for connecting to foreign servers

2

- If the foreign server is ORACLE, you need a HiRDB External Data Access Adapter for ORACLE.
- If the foreign server is HiRDB or XDM/RD E2, you do not need HiRDB External Data Access Adapter. The HiRDB External Data Access Adapter functionality is included in HiRDB External Data Access.

3

- If the foreign server is ORACLE, you need the ORACLE client product.
- If the foreign server is HiRDB or XDM/RD E2, you need a HiRDB client. However, the HiRDB server includes the HiRDB client functionality, which means that you do not need to install a HiRDB client.

For details about the products you need when using the HiRDB External Data Access facility, see the *HiRDB External Data Access Version 7 Description and User's Guide*.

2.1.5 HiRDB Non-Recover FES

Installing Non-Recover FES enables you to use front-end servers that do not need to be recovered in the event of a failure. In this case, when the unit containing such a front-end server fails and is forcibly terminated, HiRDB automatically completes any uncommitted transactions, which means that the database is updated even if the unit containing the front-end server is not restarted. For details about recovery-unnecessary front-end servers, see *8.2 Recovery-unnecessary front-end servers*.

2.2 Linkage to data linkage products

This section describes the functions that you can enable by using the data linkage products listed as follows:

- HiRDB Datareplicator and HiRDB Dataextractor
- HiRDB Adapter for XML

2.2.1 Linkage to HiRDB Datareplicator and HiRDB Dataextractor

Linking to HiRDB Datareplicator and HiRDB Dataextractor allows you to use the *replication facility*. The replication facility provides the ability to duplicate the content of a distributed database into another database. Using the replication facility, you can duplicate information from one database into another database on a different system, which helps support the management of data in a distributed system environment. The replication facility includes the following two modes:

- Data linkage facility
- Database extraction/reflection service facility

(1) *Data linkage facility*

The data linkage facility causes the updates to a database residing on a mainframe DBMS or a HiRDB system to be reflected automatically in HiRDB databases at local nodes. The data linkage facility requires the HiRDB Datareplicator, which is a related HiRDB product. The data linkage facility has the following characteristics:

- At fixed time intervals, the data linkage facility copies in sequence all updates to a corporate database into departmental databases so that the latest data available in the corporate database can be used from the departmental databases.
- The data linkage facility can extract selected data from a corporate database and copy in sequence all updates to the selected data from the corporate database into departmental databases. In this manner, the data linkage facility is able to provide data that is well-suited for data warehouse applications.

Note

Some columns are not eligible for processing by the data linkage facility, depending on the column attributes. For details about the column attributes that are not supported by the data linkage facility, see the *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide*.

(2) *Database extraction/reflection service facility*

The database extraction/reflection service facility transfers data stored in a mainframe DBMS or another HiRDB system to HiRDB databases at local nodes. The database

extraction/reflection service facility requires HiRDB Dataextractor, which is a related HiRDB product. The database extraction/reflection service facility has the following characteristics:

- At a specific time, the database extraction/reflection service facility can copy in batch information from a corporate database into departmental databases. In this manner, departmental database tables can be initialized or all data can be updated to the latest version.
- By specifying data extraction conditions, the database extraction/reflection service facility can be used to extract selected data from a corporate database so that departmental databases that are appropriate for particular types of operations can be created.
- The database extraction/reflection service facility eliminates the need for UAPs that extract data or provide such operations as character code conversion and file transfer.

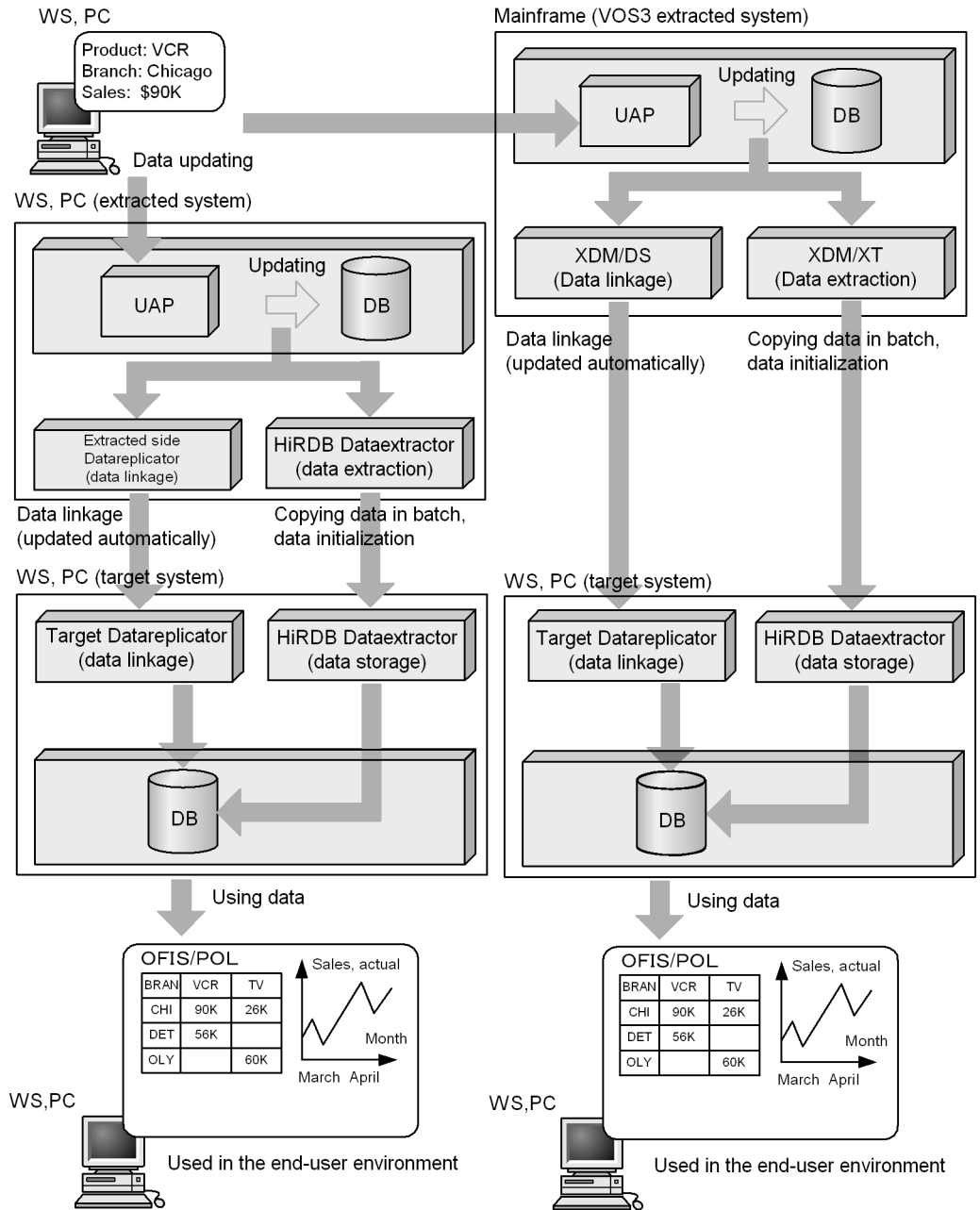
Note

Some columns are not eligible for processing by the database extraction/reflection service facility, depending on the column attributes. For details about the column attributes that are not supported by the database extraction/reflection service facility, see the *HiRDB Dataextractor Version 8 Description, User's Guide and Operator's Guide*.

(3) Example of using the replication facility

Figure 2-3 shows an example of using the replication facility.

Figure 2-3: Example of using the replication facility



For details about application examples of the replication facility, see the *HiRDB*

Datareplicator Version 8 Description, User's Guide and Operator's Guide, and the HiRDB Dataextractor Version 8 Description, User's Guide and Operator's Guide.

(4) Products required for replication facility

(a) Products required in HiRDB systems

The data linkage facility requires HiRDB Datareplicator. The database extraction/reflection service facility requires HiRDB Dataextractor, except when the remote system is RDB1 E2 or PDM II E2 of VOS1.

(b) DBMS products that can be linked to HiRDB using the replication facility

Table 2-2 lists DBMS products that can be linked to HiRDB by means of the replication facility:

Table 2-2: Products that can be linked to HiRDB by means of the replication facility

Linkable DBMS product (and applicable operating system)	Required replication facility product	
	Data linkage	Database extraction/reflection service facility
XDM/RD E2 (VOS3)	XDM/DS	XDM/XT
XDM/SD E2 (VOS3)		
ADM (VOS3)	VOS3 Database Datareplicator or XDM/DS	
PDM II E2 (VOS3) ¹	VOS3 Database Datareplicator or XDM/DS	
	File transfer program ²	
PDM II E2 (VOS1)	File transfer program ²	PDM II Dataextractor
TMS-4V/SP (VOS3)	VOS3 Database Datareplicator or XDM/DS	XDM/XT
RDB1 E2 (VOS1)	File transfer program ²	RDB1 Dataextractor

¹ In the case of PDM II E2 of VOS3, it is possible to perform data linkage using a data linkage product at the same time data linkage is being performed using a file transfer program.

² Both XFIT and an XFIT-related product are needed for the file transfer program. For details, see the *HiRDB Datareplicator Version 8 Description, User's Guide and Operator's Guide*.

(5) Setting the HiRDB environment when the data linkage facility is used

When the data linkage facility is used (in conjunction with extracting data for updating HiRDB databases), the following system definition operands must be specified:

`pd_rpl_hdepath`

Specify the HiRDB Datareplicator operation directory for the extracted side.

`pd_rpl_init_start`

Specify when the HiRDB linkage facility is to be started.

`pd_log_rpl_no_standby_file_opr`

Specify the action to be taken if system log information cannot be created in the swapped-in file.

2.2.2 Linkage to HiRDB Adapter for XML

Linking to HiRDB Adapter for XML allows you to use the *database mapping facility*.

(1) Overview of the database mapping facility

XML text consists of tags in a tree structure and sets of character strings enclosed by these tags. In contrast, databases consist of sets of related record data. HiRDB Adapter for XML provides database mapping functionality that maps XML text tags to tables and columns, and converts the character strings embedded in the XML text to appropriate data types. The facility then saves the converted data to a database, and can restore the data it has saved to the database back to the original XML text. The database mapping functionality allows you to perform the following operations:

- Saving XML text to a database
- Deleting saved XML text from a database
- Updating XML text stored in a database
- Restoring XML text from the information in a database

There are two ways to map XML text to a database: using an API or using a simple command. If you use the API, you can generate SQL code internally to access the database, or you can use the `pdload` command of the database load utility (`pdload`).

(2) Advantages of the database mapping facility

The database mapping facility provides the following advantages:

Flexible support for changing XML text and databases

Information such as the XML text structure and table formats is maintained in a definition file independent of the application. This allows you to develop applications that flexibly support changes in the content and structure of the data handled.

An abundance of supported data types

In addition to numeric data and character strings, you can store not only XML text itself in a database, but also image and other data that is referenced from within XML text. You can delete and update the stored data, as well as restore it to its original XML text structure.

Ability to perform a variety of processing with a single definition file

The definition file that is created when you save XML text to a database can be used as a definition file for updating the data and restoring it to the original XML text.

Easy manipulation of data elements

You can create callbacks to be invoked by HiRDB Adapter for XML when mapping is being executed. Through the use of such callbacks, you can perform the following processes on data extracted from XML text:

- Convert character strings to data of a specific type.
- Pass multiple data items as parameters to be manipulated as desired.
- Store data in abstract data type columns.

(3) Other features

In addition to the advantages described in the preceding subsection, the database mapping facility also includes the following features:

Linkage to XML servers

When uCosminexus Interschema - Parsing Kit is used, HiRDB Adapter for XML can link at an object level to DOM and SAX parsers.

Database access

When DABroker for C++ is used, C++ applications can be developed with HiRDB Adapter for XML to access databases. When DABroker for Java (JDBC) is used, Java applications can be developed to access databases as well. The application developer can manage database connection, disconnection, and transaction processing by directly issuing DABroker for C++ or DABroker for Java (JDBC) APIs.

2.3 Linkage to directory server products

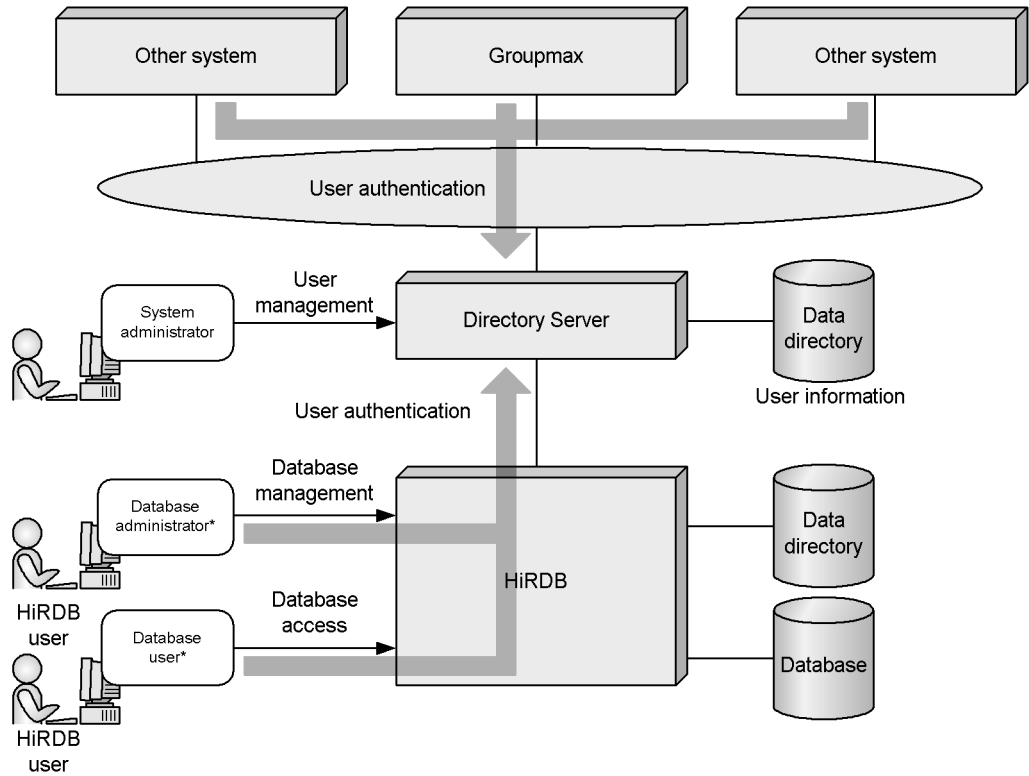
Linking to a directory server product allows you to use the *Directory Server linkage facility*. This section provides an overview of the Directory Server linkage facility. For details about how to operate the Directory Server linkage facility, see the *HiRDB Version 8 System Operation Guide*.

2.3.1 Overview of the Directory Server linkage facility

The Directory Server linkage facility is a program that provides a service (called the directory service) for using the Internet or an intranet to achieve centralized information management in a distributed system environment in order to minimize management's workload. This is made possible by using an open protocol called the Lightweight Directory Access Protocol (LDAP).

You can manage on a centralized basis in the Directory Server the organizational and user information (user IDs, passwords, the organizational units to which users belong, and user job titles) that is otherwise managed separately in HiRDB, Groupmax, and other systems. The Directory Server also allows you to retrieve and update user information from multiple nodes in the network. Figure 2-4 provides an overview of the Directory Server linkage facility.

Figure 2-4: Overview of the Directory Server linkage facility



* The Directory Server authenticates each user who attempts to access HiRDB, provided that the requisite user information (user IDs and passwords) has been registered in the Directory Server.

For details about setting the environment for and using the Directory Server linkage facility, see the *HiRDB Version 8 System Operation Guide*.

2.3.2 Directory servers to which HiRDB can be linked

HiRDB can be linked to Sun Java System Directory Servers by means of a facility that is called the *Sun Java System Directory Server linkage facility* (or simply the *Directory Server linkage facility*).

Prerequisites

- HiRDB LDAP Option
- HiRDB LDAP Option running on Windows 2000
- 32-bit version of HiRDB

2.3.3 Capabilities provided by the Directory Server linkage facility

(1) *Using a directory server to centrally manage HiRDB user information*

You can use a directory server to centrally manage user information used by HiRDB (such as authorization identifiers and passwords). You can also use a directory server for user authentication when users connect to HiRDB.

Hint:

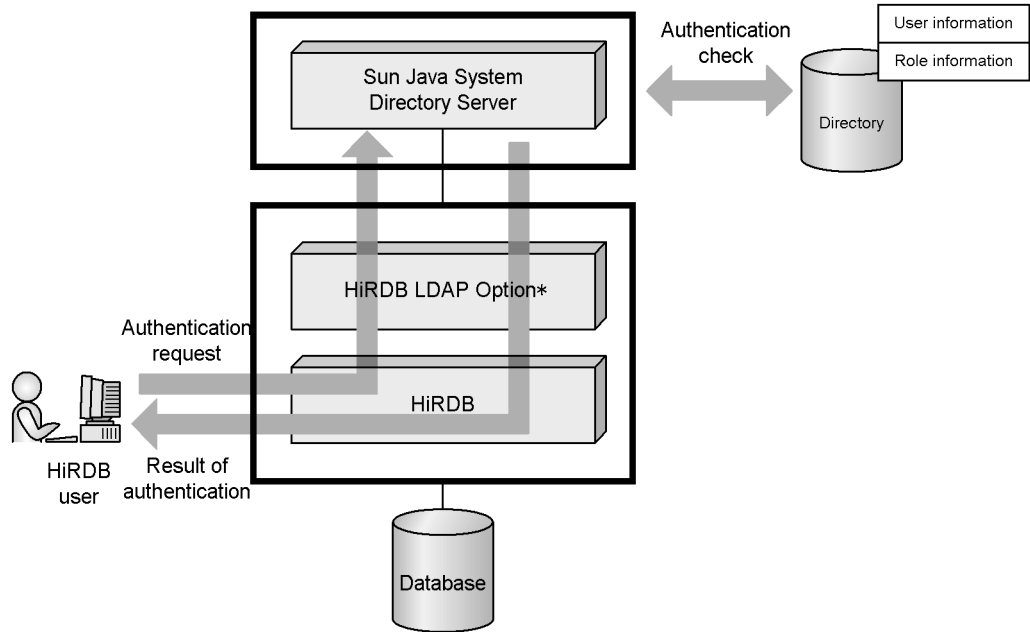
- If you use the Directory Server linkage facility, you do not need to assign the CONNECT privilege to each user individually.
- Each HiRDB authorization identifier is registered as a user ID on the directory server. Once registered, a user is assigned the CONNECT privilege.
- The password corresponding to each HiRDB authorization identifier is registered on the directory server.

Reference note:

The DBA, audit, schema definition, RDAREA usage, and table access privileges are managed by HiRDB.

Figure 2-5 provides an overview of using the Directory Server linkage facility to perform user authentication.

Figure 2-5: Overview of using the Directory Server linkage facility to perform user authentication (Sun Java System Directory Server linkage facility)



Explanation

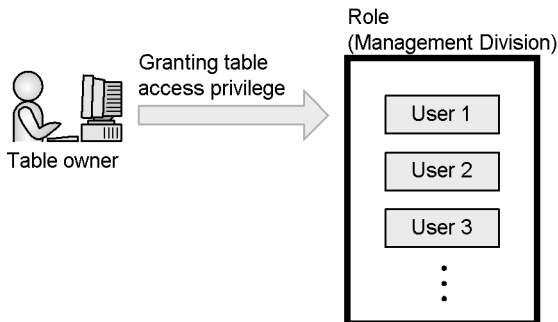
When a user attempts to connect to HiRDB (using CONNECT), Sun Java System Directory Server performs user authentication. If the user ID and password are registered in Sun Java System Directory Server, connection to HiRDB (CONNECT) is permitted.

(2) Granting table access privileges to a role

The concept of roles is supported in a Sun Java System Directory Server. You can register organizations, departments, and other personnel groupings as a single role in a directory server. You can then grant table access privileges to that role, which enables all users belonging to that role to be granted table access privileges. In other words, you can manage table access privileges by role. Figure 2-6 shows granting of table access privilege by role.

To grant table access privileges to a role, you must use the name of a role to which a filter is applied by Sun Java System Directory Server.

Figure 2-6: Granting of table access privilege to a role



Explanation

Once the table owner grants the table access privilege to the Management Division role, all users of the Management Division are able to access that table.

2.3.4 Prerequisite products

The requisite operating system and other products comply with the prerequisites for HiRDB LDAP Option. The following lists the products required to use the Sun Java System Directory Server linkage facility:

- Sun Java System Directory Server
- iPlanet Console (Sun ONE Console)*
- HiRDB LDAP Option

* Needed to register users and other directory-related information via a GUI.

For system configuration examples when the Sun Java System Directory Server linkage facility is being used, see the *HiRDB Version 8 System Operation Guide*.

2.4 Linkage to OLTP products

Transaction processing can be performed when HiRDB is linked to OLTP. This section provides an overview of linking to OLTP products. For details about the OLTP linkage procedures, see the *HiRDB Version 8 Installation and Design Guide*; for information about creating a UAP that uses a linkage to OLTP, see the *HiRDB Version 8 UAP Development Guide*.

2.4.1 OLTP products to which HiRDB can be linked

HiRDB can be linked to the following OLTP products:

- OpenTP1
- TPBroker
- TUXEDO
- WebLogic Server

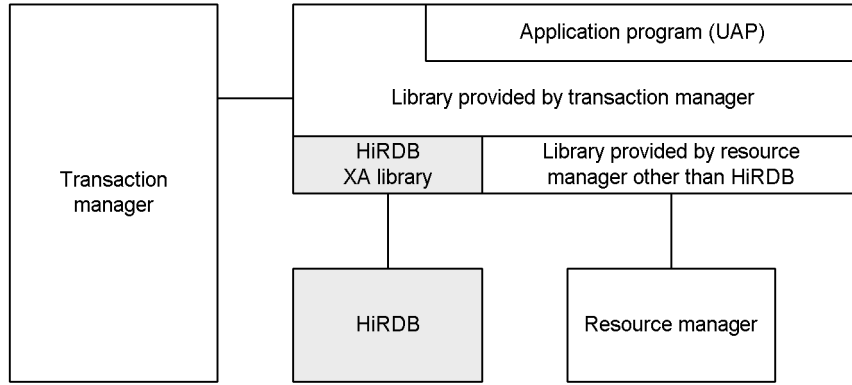
However, you cannot link to Windows (x64) because there is no client library for OLTP products running in the 64-bit mode.

2.4.2 HiRDB XA library

HiRDB uses the X/Open XA interface for linkage to OLTP. This interface, which is based on the X/Open standards, specifies the connection between the transaction manager (TM) and the resource manager (RM) in a distributed transaction processing (DTP) system. Use of the XA interface enables the resource manager's transaction processing to be controlled by the transaction manager. To do this, you must link the library provided by the transaction manager to your UAP.

HiRDB provides the HiRDB XA library so that it can control UAP processing. This library conforms to the XA interface specifications of the X/Open DTP software architecture. Figure 2-7 shows the position of HiRDB in an X/Open DTP model.

Figure 2-7: Position of HiRDB in X/Open DTP model



2.4.3 Functions provided by the HiRDB XA library

Table 2-3 lists the functions provided by the HiRDB XA library.

Table 2-3: Functions provided by the HiRDB XA library

Facility	Explanation
Transaction transfer	Executes transaction commit processing in a process different from the one that was used when the UAP accessed HiRDB. The UAP in this case uses the HiRDB XA library to connect to HiRDB. You specify in the <code>PDXAMODE</code> operand of the client environment definition whether or not the transaction transfer function is to be used. For details on transaction transfer, see the <i>HiRDB Version 8 Installation and Design Guide</i> .
Single-phase optimization	Optimizes two-phase commitment control into a single phase.
Read-only	Transaction manager performs optimization without issuing a commit request in the second phase, if HiRDB resource has not been updated when a prepare request is issued.
Dynamic transaction registration	HiRDB registers dynamically a transaction immediately before executing a UAP.
Multi-connection facility	This function separately executes multiple <code>CONNECTs</code> to a HiRDB server from a single process. For details on the multi-connection facility under the X/Open XA interface environment, see the <i>HiRDB Version 8 UAP Development Guide</i> .

Note

The HiRDB XA library does not provide the asynchronous XA call function (function that allows the transaction manager to issue asynchronous calls to the HiRDB XA library).

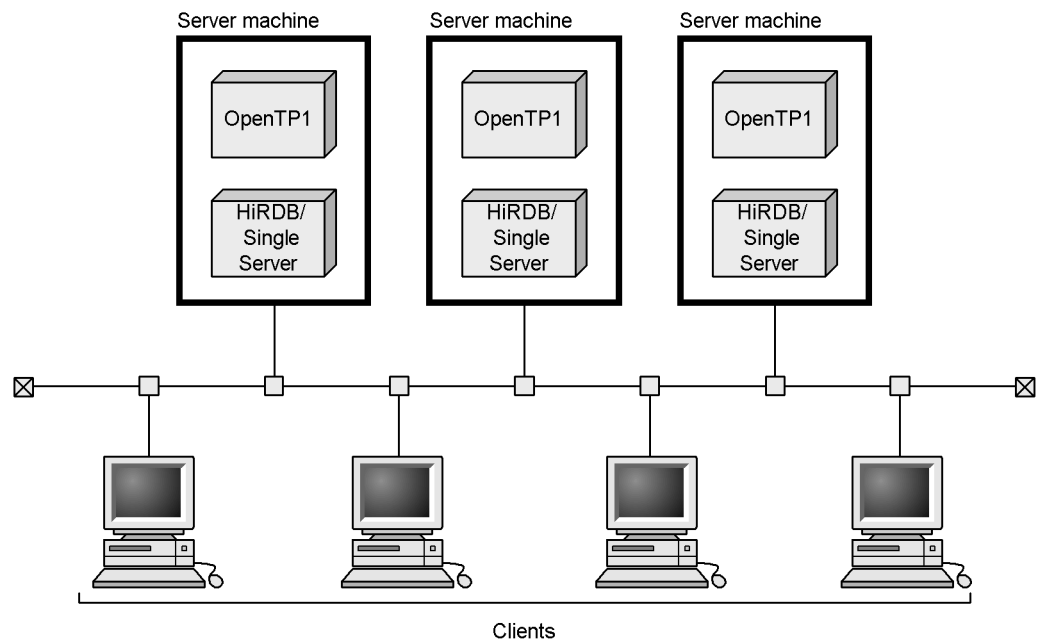
2.4.4 System configuration

This section uses OpenTP1 as an example for explaining the configuration of a system that uses OLTP.

(1) Linkage to HiRDB/Single Server

By linking HiRDB/Single Servers to OpenTP1, processing at multiple HiRDB/Single Servers can be executed centrally. In such a case, the database is partitioned and allocated using a method such as key range partitioning, and OpenTP1 running on each server machine allocates processing to each HiRDB/Single Server. In this way, large-scale transaction processing can be implemented. Linkage with OpenTP1 is the recommended means of integrating multiple HiRDB/Single Servers. Figure 2-8 shows the linkage between HiRDB/Single Servers and OpenTP1.

Figure 2-8: Linkage between HiRDB/Single Servers and OpenTP1



Explanation

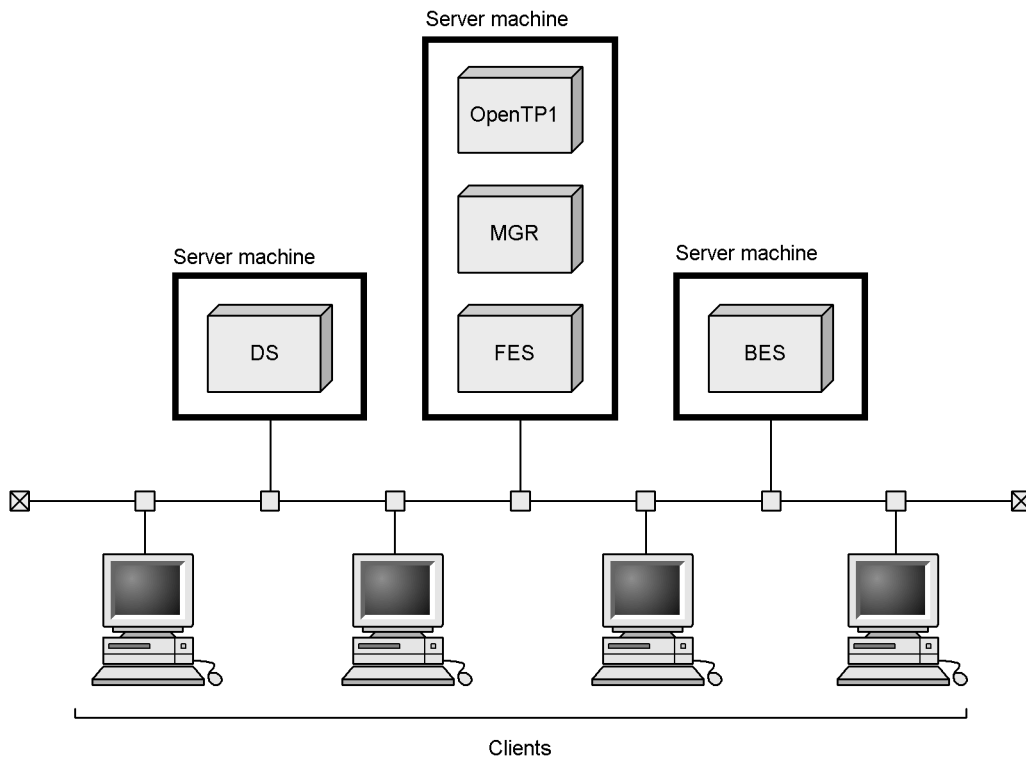
Each of the three server machines is provided with a HiRDB/Single Server and OpenTP1; transaction processing is allocated to each of the three HiRDB/Single Servers.

(2) Linkage to HiRDB/Parallel Server

By linking a HiRDB/Parallel Server to OpenTP1, high-performance, high-load

transaction processing can be achieved. Figure 2-9 shows the linkage between a HiRDB/Parallel Server and OpenTP1.

Figure 2-9: Linkage between HiRDB/Parallel Server and OpenTP1



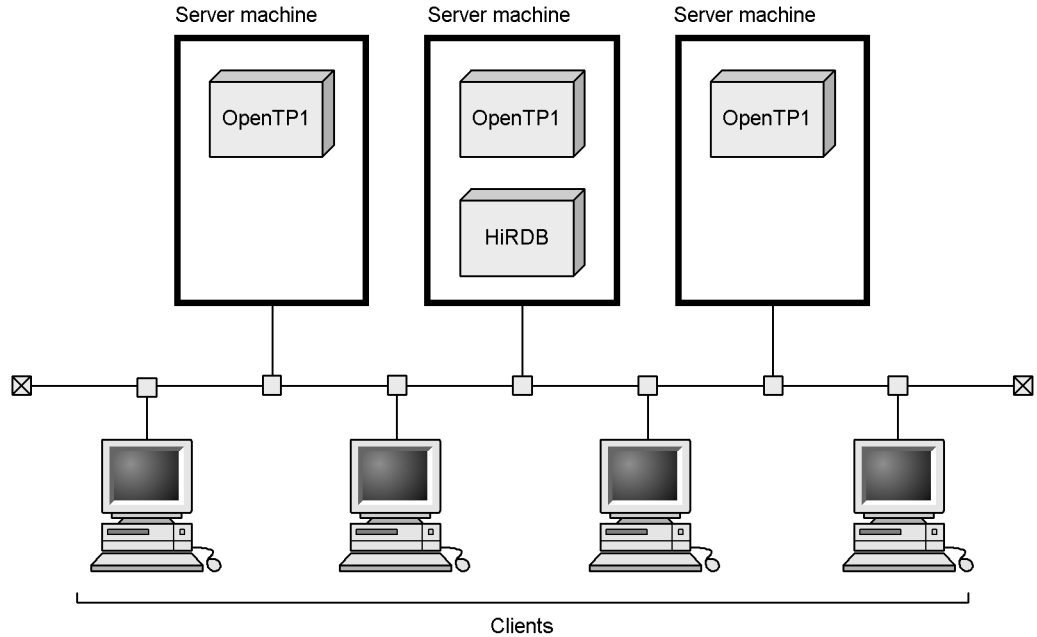
Explanation

OpenTP1 is provided at the server machine where the system manager and front-end server are installed in order to control transaction processing.

(3) Linkage to multiple OpenTP1

Figure 2-10 shows linkage to multiple OpenTP1.

Figure 2-10: Linkage to multiple OpenTP1



Note

The OLTP identifier (PDTMID in the client environment definition) of each OpenTP1 must be unique.

2.4.5 Registration of HiRDB in the transaction manager

The two methods explained as follows are provided for registering HiRDB into the transaction manager as a resource manager.

Dynamic registration

When this method is used, the UAP is placed under the control of the transaction manager when the UAP issues the first SQL statement in a transaction. In the case of a UAP that accesses multiple resource managers including HiRDB, or in the case of a UAP that may not access HiRDB at all, dynamic registration reduces the transaction manager's transaction control overhead with respect to HiRDB.

Static registration

When this method is used, the UAP is placed under the control of the transaction manager, regardless of whether or not the UAP issues SQL statements. If the transaction manager is OpenTP1 and the connection between the UAP and HiRDB is severed (e.g., due to abnormal termination of the unit or the server

process), static registration eliminates the need to restart the UAP, because an OpenTP1 facility will reconnect the UAP when a transaction is started.

For details about registering HiRDB into the transaction manager, see the *HiRDB Version 8 Installation and Design Guide*.

2.5 Linkage to operation support products

This section describes operation support products used with HiRDB.

- HiRDB SQL Executer
- HiRDB Control Manager
- HiRDB SQL Tuning Advisor
- JP1/Performance Management - Agent Option for HiRDB
- JP1/Base
- JP1/Integrated Management
- JP1/Automatic Job Management System 2

2.5.1 HiRDB SQL Executer

HiRDB SQL Executer enables interactive execution of SQL programs. Because it can perform data validation, confirmation, modification, and relatively simple, routine operations, HiRDB SQL Executer is well-suited for database maintenance and simple validation tasks. In order to use HiRDB SQL Executer on a HiRDB client, you must have either HiRDB/Developer's Kit or HiRDB/Run Time. For details on HiRDB SQL Executer, see the README file for HiRDB SQL Executer.

2.5.2 HiRDB Control Manager

HiRDB Control Manager enables you to use GUI-based mouse operations on a PC to perform basic operations, such as displaying and changing HiRDB server statuses, and backing up and recovering databases. You can also centrally operate multiple HiRDB systems from a single graphic user interface, which reduces the system operation workload.

You can use a tool to transfer to the HiRDB Control Manager backup file, unload log file, and catalog file information created with HiRDB Assist.

HiRDB Control Manager supports the following operations:

- Concurrent management of multiple HiRDB systems
- Display of HiRDB statuses
- Starting and stopping HiRDB
- Execution of the database copy utility and database recovery utility
- Operations on RDAREAs
- Operations on tables

- Operations on system log files
- Catalog registration and schedule execution
- Display of XDM/RD activity status together with RDAREA and table structure information

2.5.3 HiRDB SQL Tuning Advisor

HiRDB SQL Tuning Advisor is a product designed to help you locate and resolve SQL performance bottlenecks. It analyzes execution records output by HiRDB servers and clients, and displays the results together with advice in an easy-to-understand format.

HiRDB SQL Tuning Advisor provides the following features:

- Efficiently locates SQL code that may cause performance problems.
- Efficiently tunes SQL code that may cause performance problems.

HiRDB SQL Tuning Advisor provides the following two main facilities:

- SQL trace analysis facility

The SQL trace analysis facility analyzes SQL trace information generated by the HiRDB client, and displays on screen the aggregate time in the units occupied by the UAP or SQL code. This enables you to locate SQL code bottlenecks by summing the SQL processing time and extracting SQL code that is taking a long time to execute.

- Access path analysis facility

The access path analysis facility analyzes and displays on screen access path information generated by the HiRDB client. It can also graphically display the join relationships between tables. It can locate areas that may cause performance problems with respect to the methods used to join the tables (table scan, cross-join, and so on) and display warning advice.

2.5.4 JP1/Performance Management - Agent Option for HiRDB

JP1/Performance Management - Agent Option for HiRDB is a JP1/Performance Management agent product that collects HiRDB performance data. On distributed systems combining various platforms, JP1/Performance Management allows you to centrally manage the performance of the operating systems, server applications, databases, and other resources. By using JP1/Performance Management - Agent Option for HiRDB, you can monitor HiRDB from JP1/Performance Management. This means that you can centrally manage HiRDB performance together with that of the operating systems and other server applications. With JP1/Performance Management - Agent Option for HiRDB, you can collect performance data on the following HiRDB resources:

- Global buffers

- Server processes
- HiRDB file system area for work table files
- HiRDB units and servers
- RDAREAs
- Utilization rate of locked resources management tables

2.5.5 JP1/Base

Events such as HiRDB startup and termination can be reported to *JP1/Base*. *JP1/Base* manages reported HiRDB events as JP1 events. This enables you to use *JP1/Integrated Management* to manage events and to automatically execute jobs by linking with *JP1/Automatic Job Management System 2*. For details about monitoring events with *JP1/Integrated Management*, see 2.5.6 *JP1/Integrated Management*. For details about executing jobs by linking with *JP1/Automatic Job Management System 2*, see 2.5.7 *JP1/Automatic Job Management System 2*.

For details about *JP1/Base*, see the manual that is applicable to the version of JP1 that you are using:

- JP1 Version 8
Job Management Partner 1/Base User's Guide
- JP1 Version 6 and JP1 Version 7i
JP1 V6 JP1/Base or *Job Management Partner 1/Base*

(1) Reporting events

You specify the operands shown below to have HiRDB events reported to *JP1/Base*:

- `pd_jp1_use` operand: Y
- `pd_jp1_event_level` operand: 1 or 2

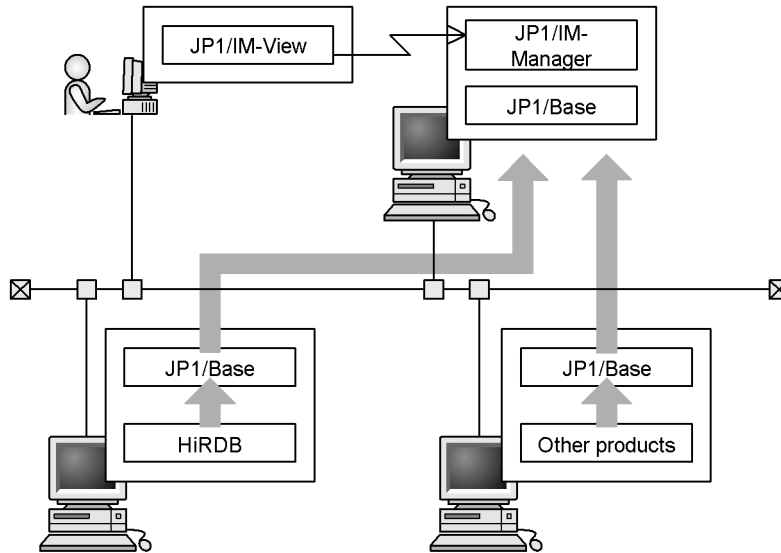
If you specify 1 for the `pd_jp1_event_level` operand, only *basic attributes* are reported. If you specify 2, *extended attributes* are also reported.

For details about the events that can be reported, see the *HiRDB Version 8 Installation and Design Guide*.

2.5.6 JP1/Integrated Management

JP1/Integrated Management optimizes (filters) JP1 events managed by *JP1/Base* and allows events issued by the system to be managed centrally by means of JP1 events. By reporting HiRDB events to *JP1/Base*, you can use *JP1/Integrated Management* to manage HiRDB events as you would those of other products. You can check events on the windows displayed by *JP1/Integrated Management*. Figure 2-11 provides an overview of event monitoring when you use *JP1/Integrated Management*.

Figure 2-11: Overview of event monitoring using JP1/Integrated Management



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

For details about the preparations for displaying HiRDB's native set of extended attributes within an overview of event monitoring using JP1/Integrated Management, see the *HiRDB Version 8 Installation and Design Guide*. For an overview of event monitoring using JP1/Integrated Management, see the manual that is applicable to the version of JP1 that you are using:

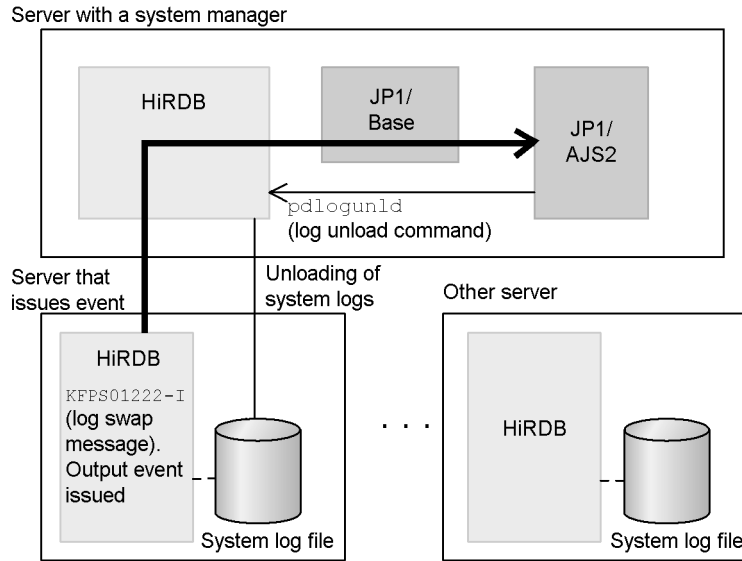
- JP1 Version 8
Job Management Partner 1/Integrated Management - Manager System Configuration and User's Guide
- JP1 Version 7i
Job Management Partner 1/Integrated Manager - Console

2.5.7 JP1/Automatic Job Management System 2



In the case of a HiRDB/Parallel Server, operations such as unloading system log files can become quite complex. For such cases, by reporting HiRDB events to JP1/Base, you can use reported events as triggers for *JP1/Automatic Job Management System 2* to automatically execute jobs, thereby automating HiRDB operations.

Figure 2-12 shows automatic control when system files are being unloaded by linking to JP1.

Figure 2-12: Automatic control when system log files are unloaded by linking with JP1



JP1/AJS2: Job Management Partner 1/Automatic Job Management System 2

 : Flow of event reporting
 : Flow of JP1 control

For details about JP1/Automatic Job Management System 2, see the manual that is applicable to the version of JP1 that you are using:

- JP1 Version 7i and JP1 Version 8
Job Management Partner 1/Automatic Job Management System 2
- JP1 Version 6
JP1/Automatic Job Management System 2 User's Guide

2.6 Linkage to data mining products

DATAFRONT is a data-mining tool for extracting rules/laws from a large volume of data and finding information that is valuable to management strategies.

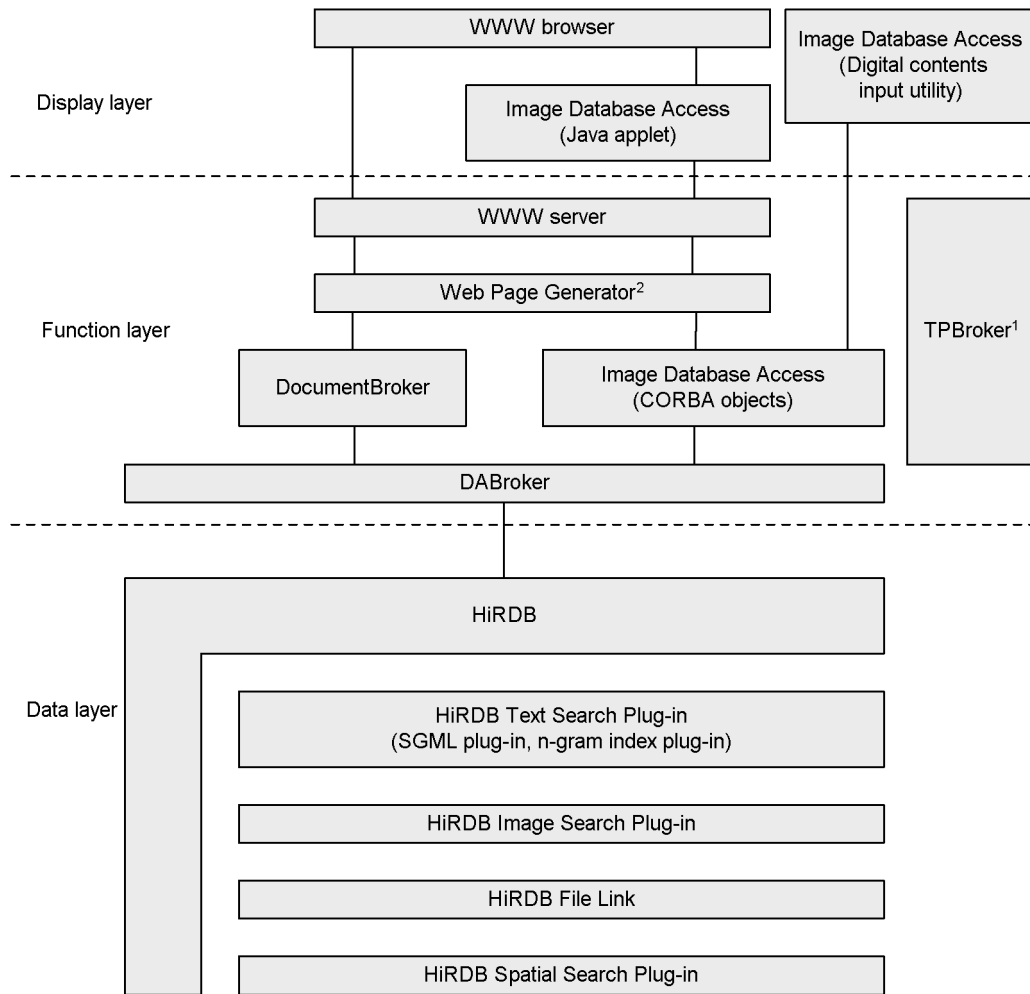
2.7 Linkage to products that handle multimedia information

Linking to the following products allows you to build a system that can support multimedia information, such as text and images, and distributed object environments:

- Plug-ins
- DocumentBroker
- Image Database Access
- Database access tools

Figure 2-13 shows the system configuration of the HiRDB-related products.

Figure 2-13: System configuration of HiRDB-related products



¹ TPBroker is a product that provides communication between objects, transaction control, and operation support functions in a distributed object environment.

² Web Page Generator is a product that provides functions for dynamic HTML generation and modification.

(1) Plug-ins

(a) HiRDB Text Search Plug-in

The HiRDB Text Search Plug-in enables you to use an SQL to search both structured SGML documents and unstructured documents. In the case of searching Japanese text, the HiRDB Text Search Plug-in features an *n-gram index method* that stores the

positions in which n characters (n -gram) occur in a given document so that a high-speed search of the entire document can be performed.

(b) HiRDB Image Search Plug-in

When a keyword search cannot sufficiently identify an image, the HiRDB Image Search Plug-in is used to retrieve similar images using image data colors and shapes as keys. The HiRDB Image Search Plug-in extracts and registers color and shape features of image data in a HiRDB database and retrieves images that have the same features as the image that is used as the key.

(c) HiRDB File Link

HiRDB File Link is used to implement a storage mode that minimizes the database load while managing a large volume of large-scale data by means of a database. In the data storage mode that uses HiRDB File Link, large-scale image data is stored in a file server and only the linkage information to the image data is kept in the database. Therefore, even if the volume of image data increases, the load on the database is kept to a minimum. Moreover, an increasing volume of image data can be handled by increasing the disk capacity or the number of file servers.

(d) HiRDB Spatial Search Plug-in

The HiRDB Spatial Search Plug-in is used to retrieve spatial data (2-dimensional data), such as mapping information.

(2) DocumentBroker

The DocumentBroker product provides a basis for development and execution of applications for constructing systems that efficiently manage large volumes of documents in a distributed object environment and applied in mission-critical operations in business and in government agencies. By storing in a HiRDB database the documents to be handled by DocumentBroker and such attributes as the document creation dates, you can assure scalability, which is a feature of HiRDB, and thus construct high-performance, highly-reliable systems. By also using the HiRDB Text Search Plug-in, you can further increase the speed of document searches. DocumentBroker provides functions such as document history management and management of multiple documents through grouping.

(3) Image Database Access

Image Database Access is a group of programs that support the building of an image management system under a distributed object environment for efficient management of a large volume of image data, such as is used by communications companies and publishing houses. Image Database Access provides CORBA objects that are intended to be executed by the Web Page Generator. Using these CORBA objects with the HiRDB Image Search Plug-in, a system that can retrieve images in the WWW environment can be built. A *digital content input/output utility* for registering image data into a database is also provided. The digital content input/output utility can

generate thumbnails and synthesize mark stamps during image data registration.

(4) Database access tools

(a) DABroker

DABroker is a database access tool for achieving high-speed access to a large volume of data. DABroker also supports a CORBA-based distributed object environment.

Database access in a distributed object environment

DABroker enables you to store multimedia information, such as documents, images, and audio, in a HiRDB database and access that information in a distributed object environment in a manner transparent to the specific location of the database.

Database response improvement

Because a multi-thread environment can be used, the amount of memory required for each connected client can be reduced. Therefore, even when the number of connected clients increases, a uniform response can be expected from the HiRDB server. Additionally, because data transfer from a client and data retrieval from the database can be executed in parallel, response performance can be improved even when the number of clients is small.

Realization of advanced database access

The Simple Interface facility, which consolidates complicated database access processing into eight functions, can implement focused and random searches.

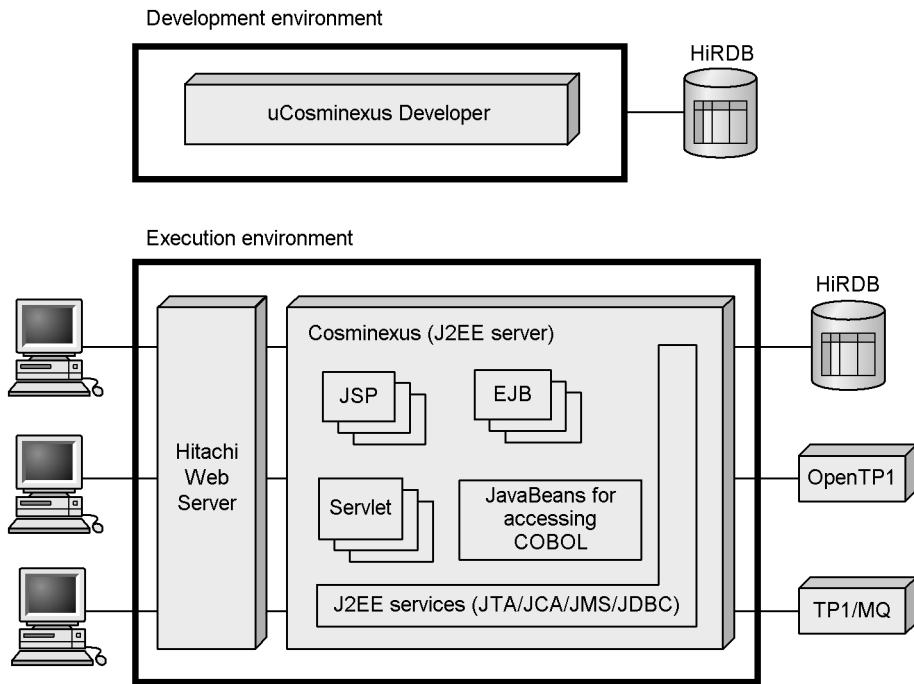
2.8 Linkage to Cosminexus

Linking to Cosminexus enables you to do the following:

- By using standard J2EE interfaces, such as JTA, JCA, or JMS, Cosminexus enables you to extend your legacy core systems onto a highly-reliable Web system.
- Take advantage of your legacy COBOL resources from Java applications via JavaBeans for accessing COBOL
- By linking with various uCosminexus Developer toolsets, you can improve the productivity of application development.

Figure 2-14 shows linking to Cosminexus.

Figure 2-14: Linking to Cosminexus



uCosminexus Developer: uCosminexus Developer Professional, uCosminexus Developer Standard
 EJB: Enterprise Java Beans
 JCA: Java Connector Architecture
 JDBC: Java DataBase Connectivity
 JMS: Java Message Service
 JSP: Java Server Page
 JTA: Java Transaction API

Chapter

3. Database Logical Structure

This chapter explains the logical structure of a database in the following sections:

- 3.1 RDAREAs
- 3.2 Schemas
- 3.3 Tables
- 3.4 Indexes
- 3.5 Expansion into an object relational database

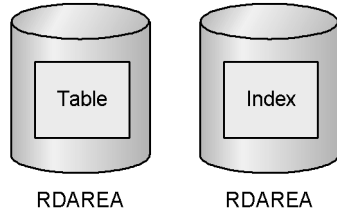
3.1 RDAREAs

This section describes the types of RDAREAs that exist and how to create them.

3.1.1 Types of RDAREAs

RDAREAs are logical areas in which tables and indexes are stored. Figure 3-1 shows the relationship between an RDAREA, a table, and an index.

Figure 3-1: Relationship between RDAREA, table, and index



The types of RDAREAs are listed in Table 3-1.

Table 3-1: Types of RDAREAs

No.	RDAREA type	Number that can be created	Required/optional
1	Master directory RDAREA	1	R
2	Data dictionary RDAREAs	1-41	R
3	Data directory RDAREA	1	R
4	Data dictionary LOB RDAREAs	2	O
5	User RDAREAs	1 to 8,388,589	R
6	User LOB RDAREAs	1 to 8,388,325	O
7	Registry RDAREA	1	O
8	Registry LOB RDAREA	1	O
9	List RDAREAs	1 to 8,388,588	O

R: Required RDAREA

O: Optional RDAREA

Notes:

- The RDAREAs in Nos. 1 through 3 are called *system RDAREAs*.
- The RDAREAs in Nos. 4, 6, and 8 are called *LOB RDAREAs*.

(1) Master directory RDAREA

The master directory RDAREA, which is required, stores internal system information.

(2) Data dictionary RDAREA

A data dictionary RDAREA stores dictionary tables and indexes for dictionary tables. Dictionary tables enable users to conduct HiRDB searches. For details about dictionary tables, see the *HiRDB Version 8 UAP Development Guide*.

There must be at least one data dictionary RDAREA.

(3) Data directory RDAREA

The data directory RDAREA, which is required, stores internal system information.

(4) Data dictionary LOB RDAREA

The data dictionary LOB RDAREAs store definition sources and objects for stored procedures and stored functions. When you define a trigger, the data dictionary LOB RDAREA also stores the internally generated SQL object of that trigger. When you use a stored procedure or a stored function, or when you define a trigger, make sure to create a data dictionary LOB RDAREA.

The two data dictionary LOB RDAREAs are:

- Data dictionary LOB RDAREA that stores the definition sources
- Data dictionary LOB RDAREA that stores the SQL objects

Reference note:

Triggers also include triggers that HiRDB generates internally based on referential constraint actions. Therefore, you must also create a data dictionary LOB RDAREA when you define a referential constraint.

(5) User RDAREA

User RDAREAs, which are required, store tables and indexes. A table can be stored in one or in multiple user RDAREAs. Similarly, a user RDAREA can store one or multiple tables. The same applies to indexes. There are two types of user RDAREAs:

Public RDAREAs

All users who are registered in HiRDB can use the public RDAREAs.

Private RDAREAs

Only users who have the requisite privilege to access a private RDAREA can access that RDAREA.

User RDAREAs that can be referenced from all back-end servers are called *shared RDAREAs*. For details about shared RDAREAs, see the *HiRDB Version 8 Installation and Design Guide*.

(6) User LOB RDAREA

User LOB RDAREAs store long variable-size data, such as documents, images, and audio. A user LOB RDAREA is required in order to use any of the following types of data:

- A column specified as a BLOB-type column (LOB column)
- An attribute specified as BLOB type in an abstract data type
- Plug-in index

If a table contains multiple types of data (LOB column, BLOB-type attribute in an abstract data type, or plug-in for a plug-in index), each type of data must be stored in a separate user LOB RDAREA. There are two types of user LOB RDAREAs:

Public RDAREAs

All users who are registered in HiRDB can use the public RDAREAs.

Private RDAREAs

Only users who have the requisite privilege to access a private RDAREA can access that RDAREA.

(7) Registry RDAREA

The registry RDAREA stores registry information. If plug-ins that use the registry facility are used, it is necessary to create a registry RDAREA. When the registry facility is used, both a registry RDAREA and a registry LOB RDAREA must be created. The registry facility is required in order to use the HiRDB Text Search Plug-in. Some plug-ins do not require the registry facility or a registry RDAREA.

(8) Registry LOB RDAREA

The registry LOB RDAREA stores registry information keys whose key length exceeds 32,000 bytes. If you use plug-ins that use the registry facility, you must create a registry LOB RDAREA. The registry facility is required in order to use the HiRDB Text Search Plug-in. Some plug-ins do not require the registry facility or a registry LOB RDAREA.

(9) List RDAREA

A list RDAREA stores lists created by the `ASSIGN LIST` statement. A list RDAREA is required in order to perform a narrowed search.

3.1.2 RDAREA creation

The following facilities can be used to create RDAREAs:

■ When HiRDB is first installed

If you use either of the following environment setup support tools, RDAREAs can be created when you run the tool:

- Simple setup tool
- Batch file (`SPsetup.bat`)

If you use a command to set up the HiRDB environment, you can use the following utility to create RDAREAs:

- Database initialization utility (`pdinit`)

■ When adding RDAREAs

You can use either of the following utilities to create (add) RDAREAs:

- Database structure modification utility (`pdmod`)
- Registry facility initialization utility (`pdreginit`)

For details about RDAREA creation, see the *HiRDB Version 8 Installation and Design Guide* or *HiRDB Version 8 System Operation Guide*.

Hint:

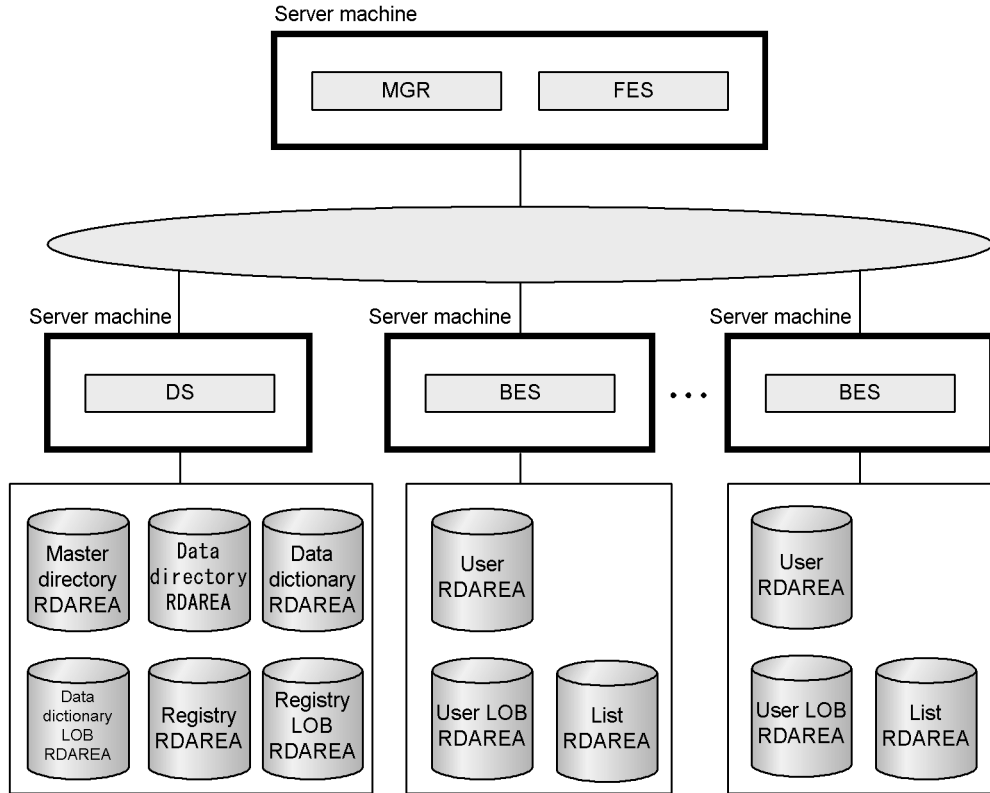
- Pay particular attention to the value of the `pd_max_rdarea_no` operand. This operand specifies the maximum number of RDAREAs. If the number of RDAREAs exceeds the value of this operand, you can no longer start HiRDB normally.
- In the case of a HiRDB/Parallel Server, there are restrictions concerning the servers on which RDAREAs can be created, as shown in Table 3-2. Figure 3-2 shows an example of an RDAREA configuration (for HiRDB/Parallel Server).

Table 3-2: Server machines on which RDAREAs can be created

Type of RDAREA	Server machine on which RDAREA is created
Master directory RDAREA	Server machine on which the dictionary server is running.
Data dictionary RDAREA	
Data directory RDAREA	
Master directory LOB RDAREA	
Registry RDAREA	
Registry LOB RDAREA	

Type of RDAREA	Server machine on which RDAREA is created
User LOB RDAREA	Server machine on which a back-end server is running.
List RDAREA	
User RDAREA	

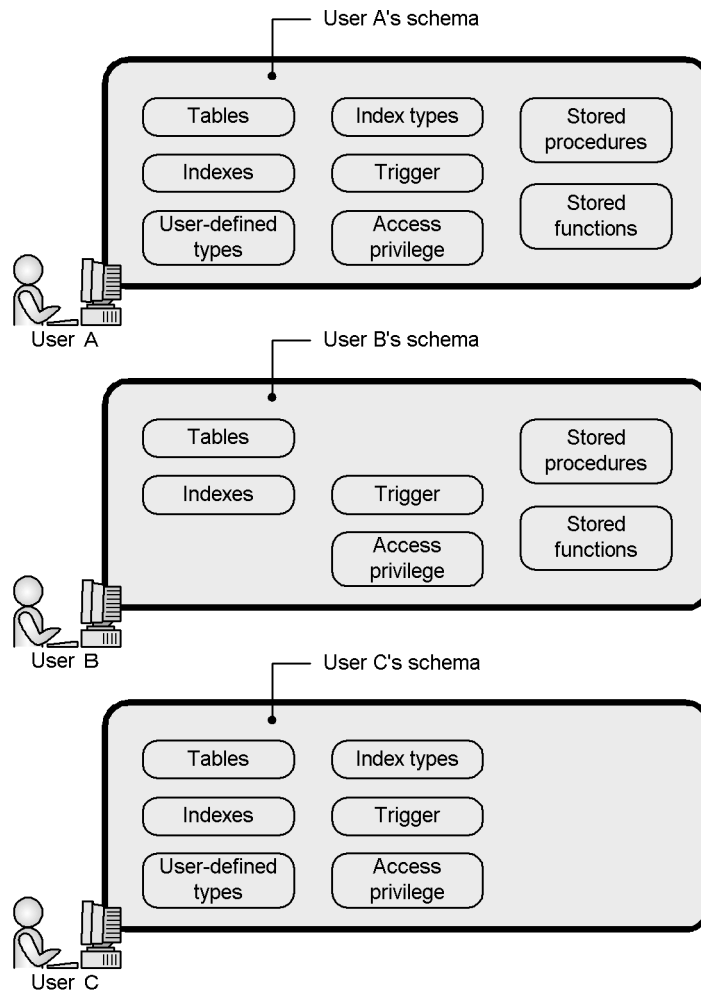
Figure 3-2: RDAREA configuration example (for HiRDB/Parallel Server)



3.2 Schemas

Tables, indexes, abstract data types (user-defined types), index types, stored procedures, stored functions, triggers, and access privileges are subsumed under a concept known as a *schema*. Before you can define any of these resources, you must define a schema. To define a schema, you use the definition SQL `CREATE SCHEMA` statement. One schema can be defined for each user. Figure 3-3 shows the concept of schemas.

Figure 3-3: Concept of schemas



3.3 Tables

A table in HiRDB is a relational database with a tabular structure composed of rows and columns. The following items should be evaluated when a table is being designed:

- What data, organized as data elements, is to be stored in the table?
- How will high-speed access be achieved?
- What security measures should be provided?

3.3.1 Basic table structure

(1) Columns and rows

The cells in a table's horizontal plane constitute *rows*; the cells in the vertical plane constitute *columns*. The row is the unit for performing operations on a table. Each row consists of data entries from one or more columns. Each column is given a *column name*. When an operation is performed on a table, you identify the position of an item in a row by its column name. Figure 3-4 shows an example of a table.

Figure 3-4: Example of a table

Product code	Product name	Color	Price	Stock quantity
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	Sweatsuit	White	45.00	45
671M	Sweatsuit	Blue	45.00	76

← Details of contents of columns

← Column names

← Row

↑ Column

Explanation

This example illustrates the structure of an inventory table (the table name is STOCK). The columns are defined with the `CREATE TABLE` in the definition SQL. Following is the definition of the STOCK table's columns:

```
CREATE TABLE STOCK
(PCODE CHAR(4),
 PNAME NCHAR(8),
 COLOR NCHAR(1)
 PRICE INTEGER,
 SQUANTITY INTEGER);
```

(2) Data types

A data type must be specified for each column and for each attribute comprising an abstract data type. Data types are classified broadly as predefined data types and user-defined data types. A predefined data type is provided by HiRDB. A user-defined data type is defined by a user as needed. Table 3-3 lists the available data types.

Table 3-3: Available data types

Classification		Data type	Data format
Predefined	Numeric data	INTEGER	Integer
		SMALLINT	Integer
		LARGE DECIMAL	Fixed point
		FLOAT	Double-precision floating point
		SMALLFLT	Single-precision floating point
	Character data	CHARACTER	Fixed-size character string
		VARCHAR	Variable-size character string
	National character data	NCHAR	Fixed-size national character string
		NVARCHAR	Variable-size national character string
	Mixed character data	MCHAR	Fixed-size mixed character string
		MVARCHAR	Variable-size mixed character string
	Date data	DATE	Date
	Time data	TIME	Time
	Time-stamp data	TIMESTAMP	Precision for time-stamp data in floating-point seconds
	Date interval data	INTERVAL YEAR TO DAY	Date interval
	Time interval data	INTERVAL HOUR TO SECOND	Time interval
Large object data	BLOB	Binary data string	
Binary data	BINARY	Binary data string	
Boolean data	BOOLEAN	Boolean value	
User-defined	Abstract data type	—	—

— : Not applicable

Data types are specified when a table is created with the `CREATE TABLE` of the

definition SQL. The data type for an attribute comprising an abstract data type is specified with the `CREATE TYPE` of the definition SQL. For details on abstract data types, see Section 3.5 *Expansion into an object relational database*.

3.3.2 Table normalization

The operation of removing redundant data from one table into another table is called *table normalization*. Table normalization should be used to improve the storage efficiency of table data as well as to enhance the capacity for concurrent executions during access processing. By normalizing a given table repeatedly, it becomes possible to transform a complex database into an optimal database. Figure 3-5 illustrates table normalization. For details about table normalization, see the *HiRDB Version 8 Installation and Design Guide*.

Figure 3-5: Table normalization

● Before normalization

STOCK

Product number	Product code	Product name	Price	Stock quantity
PNO	PCODE	PNAME	PRICE	SQUANTITY
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	Sweatsuit	45.00	45
06711	671	Sweatsuit	45.00	76

These two columns always contain corresponding values (1-to-1 correspondence). Thus, these two columns are redundant.

● After normalization

STOCK

PNO	PNAME	PRICE	SQUANTITY
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	Sweatsuit	45.00	45
06711	Sweatsuit	45.00	76

PRODUCT

PCODE	PNAME
101	Blouse
202	Polo shirt
353	Skirt
412	Sweater
591	Socks
671	Sweatsuit

Explanation

Before normalization, the columns `PCODE` and `PNAME` in the `STOCK` table have a 1-to-1 correspondence; the information in these two columns is redundant. In such a case, you can create another table, named `PRODUCT` in this example, composed of the `PCODE` and `PNAME` columns from the `STOCK` table, so that in the `PRODUCT` table there is no duplication of information in the `PCODE` and `PNAME` columns.

3.3.3 FIX attribute

The attribute that is assigned to a table whose row size is fixed is called the *FIX attribute*. Specifying the `FIX` attribute for a fixed-size table that contains no null values speeds up column retrieval processing. Use of this technique improves access performance even when there is a large number of columns. The `FIX` attribute should be specified when all the following conditions are satisfied:

- No columns will be added
- No columns contain the null value
- There are no variable-size columns

The `FIX` attribute is assigned to a table by specifying the `FIX` option in the `CREATE TABLE` of the definition SQL. For details about the `FIX` attribute, see the *HiRDB Version 8 Installation and Design Guide*.

3.3.4 Primary key

A *primary key* makes it possible to uniquely identify the rows in a table. The columns constituting a defined primary key are subject to the *uniqueness constraint* and the *NOT NULL constraint*. The uniqueness constraint means that no value can be duplicated in the key area (column or set of columns)—i.e., there is a constraint that all data entries in the key columns must be unique. The `NOT NULL` constraint means that the null value is not permitted as a value in the key columns.

If there are one or more columns or sets of columns (called *candidate keys*) on the basis of which the rows in a table can be uniquely identified, you can select a primary key from among the candidate keys. You should define this primary key so that it is mnemonically significant and appropriate for being assigned the uniqueness and `NOT NULL` constraints.

The primary key is defined with the `PRIMARY KEY` option in the `CREATE TABLE`. For details about the primary key, see the *HiRDB Version 8 Installation and Design Guide*.

3.3.5 Cluster key

You can store the rows in a table in either ascending or descending order. To do this, you need to define a *cluster key*. Using a cluster key for storing the rows in a table in ascending or descending order reduces the data I/O time for the following types of

operations:

- Searching, updating, or deleting rows by specifying a range
- Searching or updating rows in cluster key sequence

Applicability

You should specify a cluster key when all the following conditions are satisfied:

- Data will be accumulated in either ascending or descending order of the key values and the data will be accessed frequently in key value sequence.
- No key values are to be changed.
- The table's rows have a fixed row size.

A cluster key is defined with the `CLUSTER KEY` option in the `CREATE TABLE`. For details about the cluster key, see the *HiRDB Version 8 Installation and Design Guide*.

3.3.6 Suppress option

The *suppress option* refers to abbreviating some of data in a table so that the data can be stored in a smaller space than its actual size. When the suppress option is specified, the system stores any decimal data in the table by taking only the significant digits (ignoring leading zeros) and a value indicating the actual data size. Because the suppress option stores data by truncating it to less than its actual size, less disk space is used and I/O time for text and other search operations is reduced.

The suppress option is specified with the `SUPPRESS` option in the `CREATE TABLE`. For details about the suppress option, see the *HiRDB Version 8 Installation and Design Guide*.

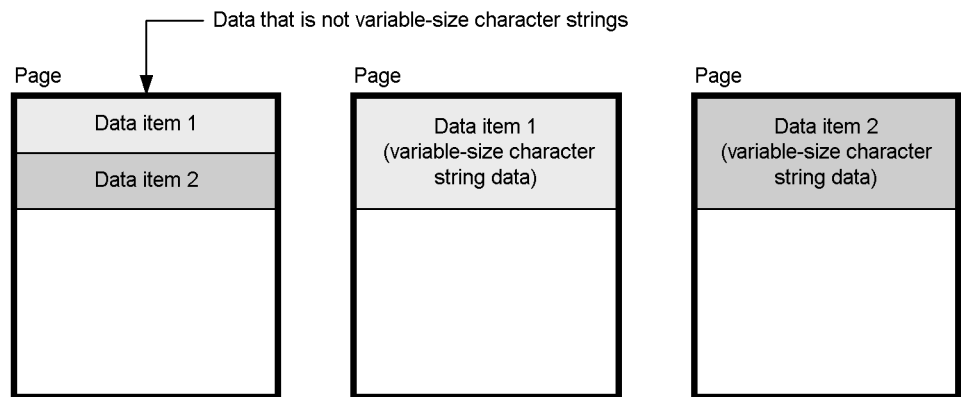
3.3.7 No-split option

When any of the following data types is defined in a table and the actual size of the data of any of these data types is equal to or greater than 256 bytes, the normal procedure results in each row of data being stored on multiple pages:

- `VARCHAR`
- `MVARCHAR`
- `NVARCHAR`

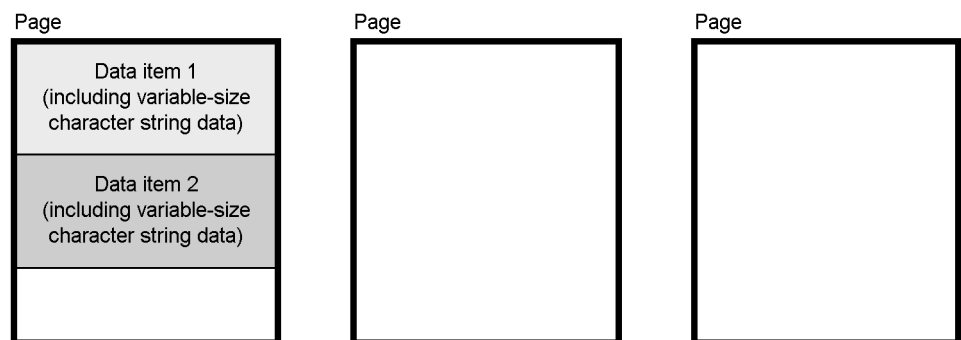
Figure 3-6 shows the data storage method that is used normally.

Figure 3-6: Normal data storage method when the actual data size of a variable-size character string is at least 256 bytes



As illustrated, data that is not part of variable-size character strings and data belonging to variable-size character string are stored on different pages, which reduces the data storage efficiency. In this case, you can improve the data storage efficiency by specifying the no-split option. When the no-split option is specified, the system stores each entire row on the same page, even if the actual data size of a variable-size character string is equal to or greater than 256 bytes. Figure 3-7 shows the data storage method that is used when the no-split option is specified.

Figure 3-7: Data storage method used when the no-split option is specified



Explanation

The no-split option stores all data from a row on the same page, which improves the data storage efficiency as compared to when the no-split option is not specified.

The no-split option is specified with the `NO SPLIT` option in the `CREATE TABLE` or `CREATE TYPE`. For details about the no-split option, see the *HiRDB Version 8*

Installation and Design Guide.

3.3.8 Table row partitioning

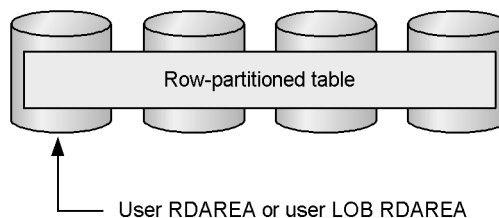
The process by which a single table is split amongst multiple user RDAREAs or user LOB RDAREAs is called *table row partitioning*. A table that is partitioned by this method is called a *row-partitioned table*. When you row-partition a table, you can use user RDAREAs or user LOB RDAREAs as the unit for storing the table's data, reorganizing the table, and making backups.

For example, you could row-partition a table to match the different types of UAPs (job types) that will access the table and then store the sections appropriate to the various UAP types in different RDAREAs. Then, when it becomes necessary to make a backup, you need stop only the UAPs that are accessing the RDAREA that is to be backed up, which facilitates overall system operation.

In the case of a HiRDB/Parallel Server, it is possible to access concurrently a table in user RDAREAs or user LOB RDAREAs under multiple back-end servers, resulting in high-speed table access and better load distribution.

Figure 3-8 shows table row partitioning.

Figure 3-8: Table row partitioning



There are two methods for row-partitioning a table:

- Key range partitioning
- Hash partitioning (flexible hash partitioning and FIX hash partitioning)

(1) Key range partitioning

Key range partitioning is when ranges of values in a specified column are used as the conditions for row-partitioning the table. The specific column that provides the conditions for row partitioning is called the *partitioning key*. Key range partitioning is used when the actual RDAREAs in which table data is stored must be known. There are two ways to specify this type of row partitioning:

- Storage condition specification
- Boundary value specification

(a) Storage condition specification

Comparison operators are used to specify conditions for selecting the data to be stored in specific RDAREAs. Only one range of values, as specified by a storage condition, can be set in each RDAREA. Figure 3-9 shows an example of key range partitioning with a storage condition specified.

Figure 3-9: Key range partitioning: Example of storage condition specification

STOCK				
PCODE	PNAME	COL	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	Sweatsuit	White	45.00	45
671M	Sweatsuit	Blue	45.00	76

↑
Partitioning key

To be stored in RDAREA01

To be stored in RDAREA02

- Row-partitioned table stored in RDAREA01

STOCK (data in rows 101L-353M)

PCODE	PNAME	COL	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-partitioned table stored in RDAREA02

STOCK (data in rows 411M-671M)

PCODE	PNAME	COL	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	Sweatsuit	White	45.00	45
671M	Sweatsuit	Blue	45.00	76

Explanation

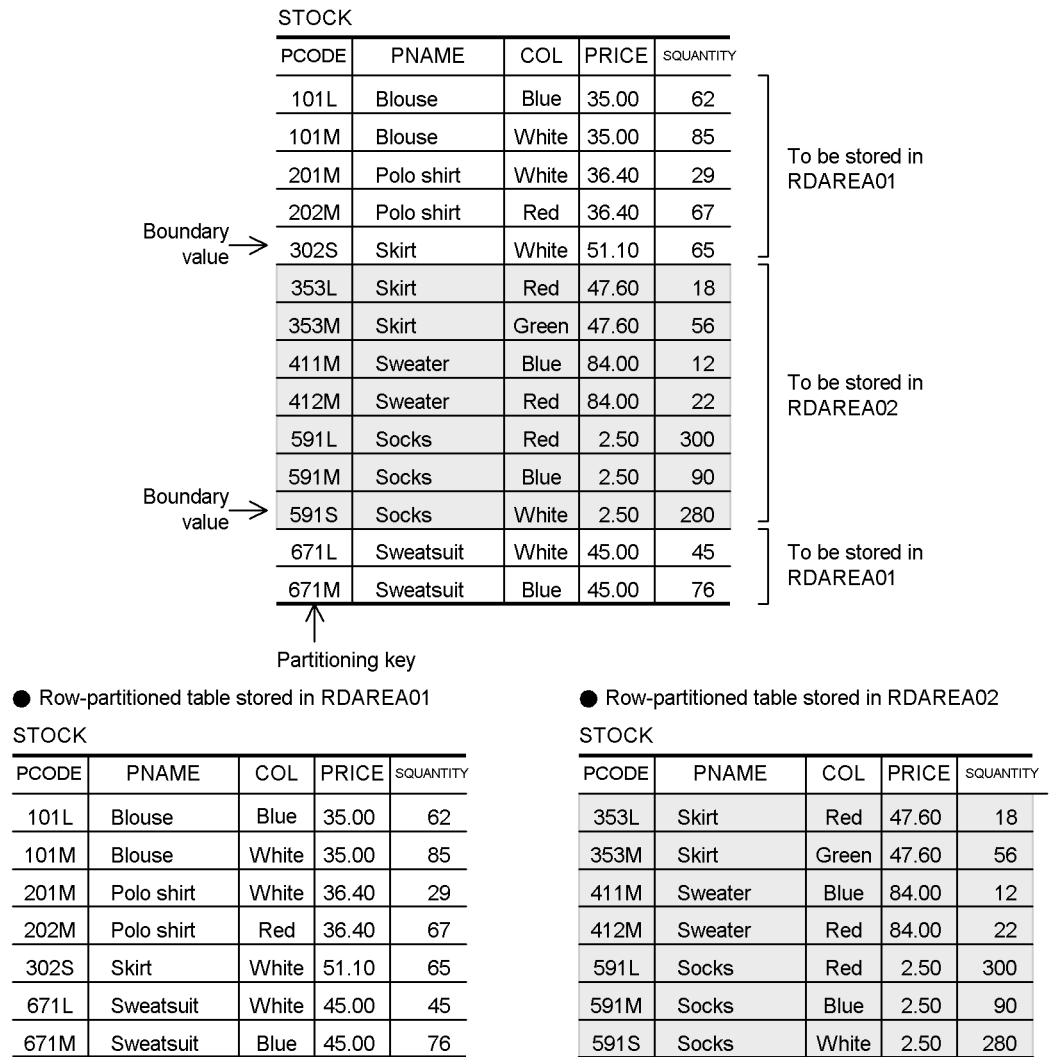
The STOCK table is row-partitioned using PCODE as the partitioning key; the results of row-partitioning are stored in RDAREA01 and RDAREA02:

```
CREATE TABLE STOCK
  (PCODE CHAR(4) NOT NULL, PNAME NCHAR(8),
   COLOR NCHAR(1), PRICE INTEGER, SQUANTITY INTEGER
  ) IN ((RDAREA01) PCODE<='353M', (RDAREA02));
```

(b) Boundary value specification

Boundary values for the data to be stored in each RDAREA are specified with literals in ascending order. Multiple ranges delimited by boundary values can be specified for each RDAREA. Figure 3-10 shows an example of key range partitioning with boundary values specified.

Figure 3-10: Key range partitioning: Example of boundary value specification



Explanation

The STOCK table is row-partitioned using PCODE as the partitioning key; the results of row-partitioning are stored in RDAREA01 and RDAREA02:

```
CREATE TABLE STOCK
  (PCODE CHAR(4) NOT NULL, PNAME NCHAR(8),
  COLOR NCHAR(1), PRICE INTEGER, SQUANTITY INTEGER
  ) PARTITIONED BY PCODE
  IN ((RDAREA01) '302S', (RDAREA02) '591S', (RDAREA01));
```

(2) Hash partitioning

Hash partitioning is when a table is row-partitioned by using a hash function to store evenly among the storage RDAREAs the values contained in the columns that make up the table. The specific column that provides the conditions for row-partitioning is called the *partitioning key*. Hash partitioning is used when it is necessary to store the same amount of table data in each of the RDAREAs. Table 3-4 provides an overview of the two types of hash partitioning.

Table 3-4: Types of hash partitioning

Hash partitioning type	Explanation
Flexible hash partitioning	The RDAREAs in which the table is stored cannot be determined. Therefore, a search process must check all back-end servers that may contain parts of the table.
FIX hash partitioning	HiRDB keeps track of the RDAREAs in which the table is stored. Consequently, a search process has to check only the back-end servers that are expected to contain the table's the data.

Figure 3-11 shows an example of hash partitioning.

Figure 3-11: Example of hash partitioning

STOCK				
PCODE	PNAME	COL	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	Sweatsuit	White	45.00	45
671M	Sweatsuit	Blue	45.00	76

↑
Partitioning key

● Row-partitioned table stored in RDAREA01

STOCK				
PCODE	PNAME	COL	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	Sweatsuit	Blue	45.00	76

● Row-partitioned table stored in RDAREA02

STOCK				
PCODE	PNAME	COL	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	Sweatsuit	White	45.00	45

Explanation

The `STOCK` table is row-partitioned using `PCODE` as the partitioning key; the results of row-partitioning are stored in `RDAREA01` and `RDAREA02`:

Note that the actual `RDAREAs` in which the data is stored may differ from those in the example.

```
CREATE TABLE STOCK
```

```
(PCODE CHAR(4) NOT NULL, PNAME NCHAR(8),
COLOR NCHAR(1), PRICE INTEGER, SQUANTITY INTEGER
) [FIX]* HASH HASH6 BY PCODE
IN (RDAREA01, RDAREA02);
```

* This is specified for FIX hash partitioning.

(3) Examples of table row partitioning

The RDAREAs into which a row-partitioned table is stored should be allocated for different disks. If they are allocated on the same disk, contention for access to these RDAREAs can occur, resulting in decreased performance.

The concepts of *row partitioning among servers* and *row partitioning within a server* apply to a HiRDB/Parallel Server. Row-partitioning among servers is the mode in which a table is row-partitioned over multiple back-end servers. Row-partitioning within a server is the mode in which a table is row-partitioned within a single back-end server. In the case of a HiRDB/Single Server, row-partitioning is always within the server.

Figure 3-12 shows an example of table row partitioning for a HiRDB/Single Server, and Figure 3-13 shows an example of table row partitioning for a HiRDB/Parallel Server.

Figure 3-12: Example of table row partitioning: HiRDB/Single Server

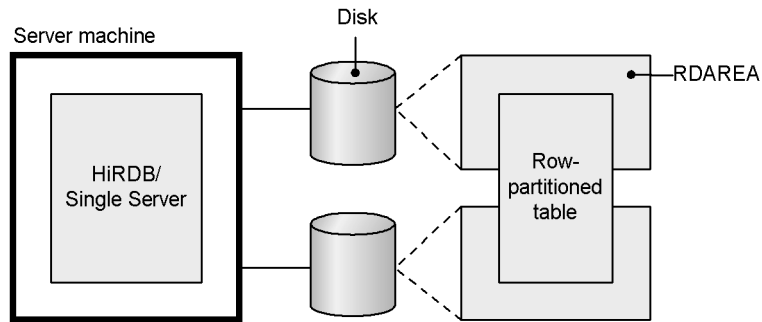
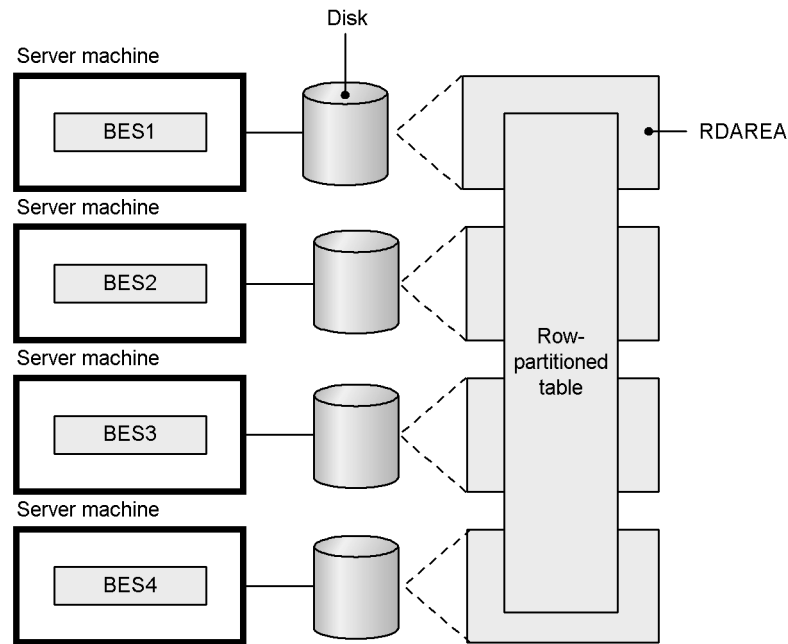


Figure 3-13: Example of table row partitioning: HiRDB/Parallel Server



Explanation

The table is row-partitioned among back-end servers BES1 to BES4.

(4) Table row partitioning definition

To row-partition a table, the elements listed below must be specified in the `CREATE TABLE` definition SQL:

- Allocation of the table to RDAREAs
- Row-partitioning method (specification of storage conditions, boundary values, or hash partitioning)

For details about the row-partitioning design considerations for improving processing performance, see the *HiRDB Version 8 Installation and Design Guide*.

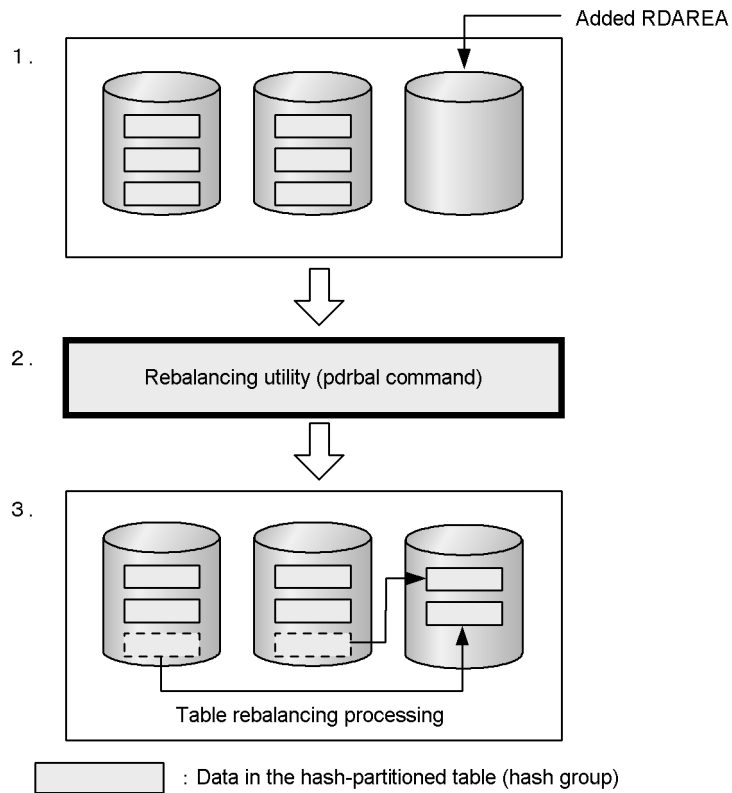
(5) Hash facility for hash row partitioning

When new RDAREAs are added to accommodate an increase in the volume of data in a hash-partitioned table (increase in the number of table partitions), there may be significant differences in the amount of data stored in the existing RDAREAs and in the new RDAREAs. The *hash facility for hash row partitioning* corrects this sort of imbalance in the amount of data in RDAREAs as a result of increasing the number of table partitions. Figure 3-14 illustrates the hash facility for hash row partitioning. This

facility can be applied to both FIX hashing and flexible hashing.

For details about the hash facility for hash row partitioning, see the *HiRDB Version 8 System Operation Guide*.

Figure 3-14: Hash facility for hash row partitioning



Explanation

1. Because the hash-partitioned table became filled with data, an additional RDAREA for storing table data was provided (the number of table partitions was increased). No data is placed in the new RDAREA, which creates a data volume imbalance.
2. The *rebalancing utility* (pdrbal command) is executed in order to correct the data volume imbalance.
3. Execution of the rebalancing utility moves and re-allocates data in units of hash groups in a process called *table rebalancing*. The hash facility for hash row partitioning causes HiRDB to divide data into 1024 groups (called *hash groups*) based upon the result of hashing the primary key. For each group, an

RDAREA segment is allocated, and the data is stored. This reallocation of data is also performed in units of hash groups.

Application criteria

- An increase in the amount of data to be stored is anticipated, so an RDAREA is needed for potential future use.*
- Because of the large amount of data in the table, it would be prohibitively difficult to re-create the table.

* Although normally an RDAREA cannot be added to a FIX hash-partitioned table in which data is already stored, you can use the hash facility for hash row partitioning to add an RDAREA.

Operating procedure

Following is a summary of the operating procedure for using the hash facility for hash row partitioning:

1. When a hash partitioned table is defined, use one of the hash functions A to F to define the table as a *rebalancing table*.
2. Add a table storage RDAREA in order to increase the number of table partitions for the table.
3. Execute the rebalancing utility to rebalance the table.

3.3.9 Table matrix partitioning

Partitioning a table by a combination of partitioning methods using two table columns as partitioning keys is called *matrix partitioning*. The first column used as a partitioning key is called the *first dimension partitioning column*, and the second column used as a 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

A table that has been matrix-partitioned is called a *matrix-partitioned table*. You can also perform matrix partitioning on indexes that have been mapped to a matrix-partitioned table. Note that you must have HiRDB Advanced Partitioning Option to perform matrix partitioning on a table.

(1) Benefits of table matrix partitioning

The following describes the benefits that are provided by using multiple columns as partitioning keys to split a table:

- Increase in SQL processing speed

You can execute SQL processes in parallel and more quickly perform searches by using multiple keys to narrow the searched range.

- Decrease in operation time

Matrix partitioning creates smaller partitions, which decreases the size of any single RDAREA, and reduces the time required to perform operations such as reorganization, backup, and failure recovery.

(2) Application 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. Or, you wish to limit the RDAREAs that are accessed only by column *n* specified in the SQL statement.

When the following conditions are met, we recommend that you combine key range partitioning with boundary values specified and hash partitioning:

- 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) How matrix partitioning is defined

To define matrix partitioning, specify the following in the `PARTITIONED BY MULTIDIM` operand of the `CREATE TABLE` definition SQL:

- Allocation of a table to an RDAREA
- The matrix partitioning method (partitioning key, partitioning method)

(4) Examples of matrix partitioning

(a) Key range partitioning with boundary values specified is used for the second dimension partitioning column

In this example, boundary values are specified for the registration date (`RDATE`) and the store number (`SNUM`) in a customer data table. The table is then matrix partitioned

such that the user data is stored by registration date and store number in the user RDAREAs USR01 to USR06. In this example, the number of user RDAREAs needed to store this data is (the number of boundary values for RDATE +1) × (the number of boundary values for SNUM + 1), so the number of user RDAREAs needed is 3 × 2 = 6.

RDATE	SNUM	
	100 or less	101 or greater
2000 and earlier	USR01	USR02
2001	USR03	USR04
2002 and later	USR05	USR06

The following shows the SQL code used to matrix partition the table:

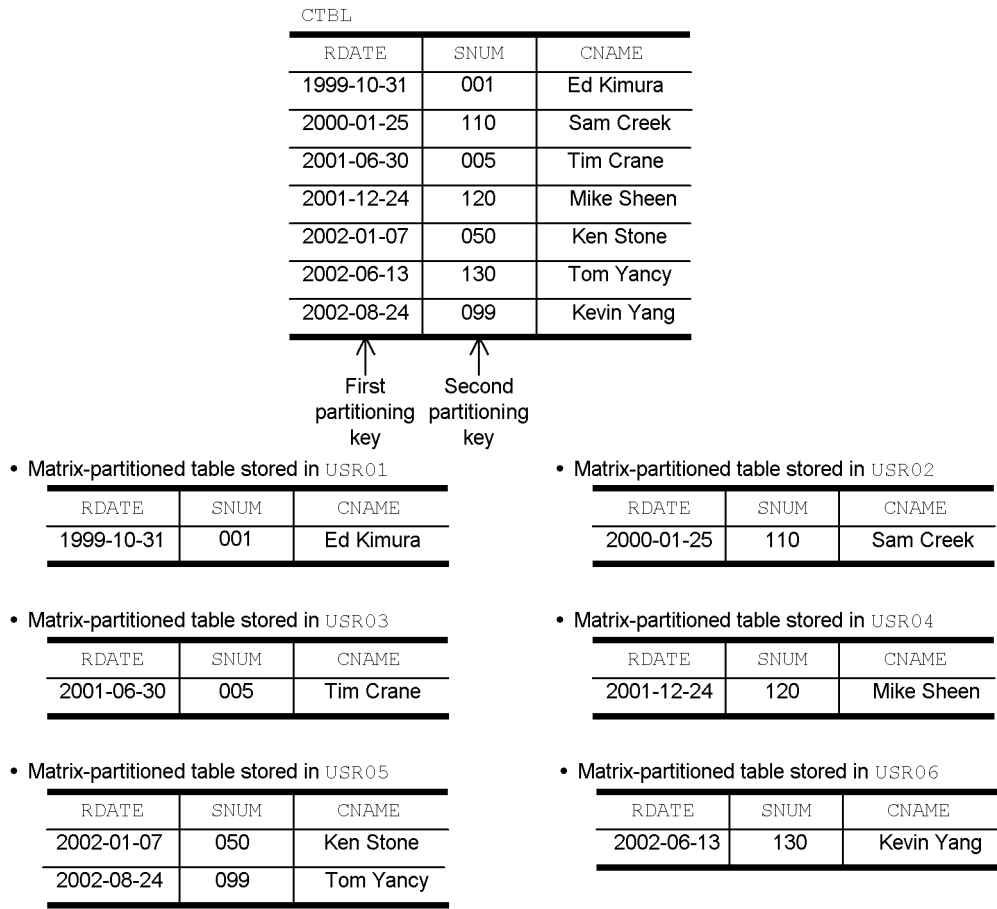
```
CREATE FIX TABLE CTBL
  (RDATE DATE, SNUM INT, CNAME NCHAR(10))
  PARTITIONED BY MULTIDIM(
    RDATE (('2000-12-31'), ('2001-12-31')), ...1.
    STORE_NO ((100)) ...2.
  ) IN ((USR01,USR02), (USR03,USR04), (USR05,USR06))
```

Explanation

1. Specifies the name of the first dimension partitioning column (name of the column that is used as the first partitioning key), and its list of boundary values.
2. Specifies the name of the second dimension partitioning column (name of the column that is used as the second partitioning key), and its list of boundary values.

Figure 3-15 shows an example of matrix partitioning.

Figure 3-15: Example of matrix partitioning (combining with key range partitioning with boundary values specified)



(b) Hash partitioning is used for the second dimension partitioning column

This example uses FIX hash partitioning for the second dimension partitioning column.

In this example, boundary values are specified for the registration dates in a customer table, and then a hash function is used to separate the data by store number and region code into three segments. This results in a table that is matrix partitioned such that each segment of customer data is stored in one of the user RDAREAs (USR01 to USR09) shown below. The number of user RDAREAs needed to store the resulting data is (number of boundary values + 1) × (desired partitions to be obtained by hash function). In this example, the number is 3 × 3 = 9.

Registration date	Store number and region code (divided into 3 partitions by hash function)		
2002 and earlier	USR01	USR02	USR03
2003	USR04	USR05	USR06
2004 and later	USR07	USR08	USR09

The following code shows the SQL statement used to define 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 column to be used as the first partitioning key) and its list of boundary values.
2. Specifies the name of the second dimension partitioning column (name of the column to be used as the second partitioning key) and the name of the hash function.

Figure 3-16 shows an example of matrix partitioning.

Figure 3-16: Example of matrix partitioning (combination of key range partitioning with boundary values specified and hash partitioning)



- Matrix-partitioned table stored in `USR03`

RDATE	SNUM	RCODE	CNAME
2002-06-30	005	01	Tim Crane

- Matrix-partitioned table stored in `USR04`

RDATE	SNUM	RCODE	CNAME
2003-01-24	120	03	Mike Sheen
2003-08-24	099	02	Kevin Yang

- Matrix-partitioned table stored in `USR05`

RDATE	SNUM	RCODE	CNAME
2003-03-07	050	01	Ken Stone
2003-11-15	010	01	Kim Nun

- Matrix-partitioned table stored in `USR06`

RDATE	SNUM	RCODE	CNAME
2003-04-13	130	03	Tom Yancy
2003-12-24	020	01	Ted Little

- Matrix-partitioned table stored in `USR07`

RDATE	SNUM	RCODE	CNAME
2004-01-27	080	02	Lynn Felt

- Matrix-partitioned table stored in `USR08`

RDATE	SNUM	RCODE	CNAME
2004-03-24	170	04	Fawn Smith

- Matrix-partitioned table stored in `USR09`

RDATE	SNUM	RCODE	CNAME
2004-07-24	015	01	Bill Nunez

3.3.10 Changing the partitioning storage conditions of a table

You can use `ALTER TABLE` to change the partitioning storage conditions of tables that have been row-partitioned by key range partitioning. * Changing the partitioning storage conditions of a table can reduce your workload by enabling you to reuse `RDAREAs` in which outdated data is stored. `ALTER TABLE` deletes a table whose partitioning storage conditions are to be changed, eliminating the need to re-create the table.

*

`ALTER TABLE` can be used to change a table's partitioning storage condition when either of the following partitioning methods was used:

- Boundary value specification
- Storage condition specification (and the equal sign (=) was used as the comparison operator for the storage condition)

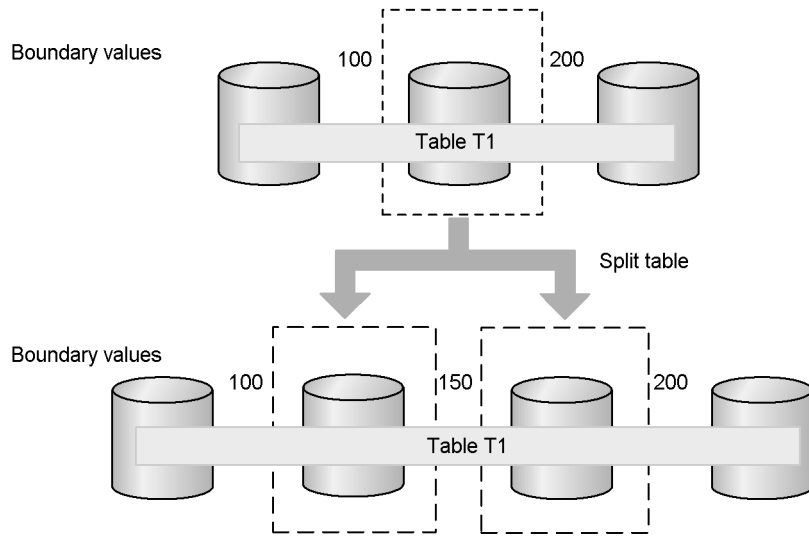
To change the partitioning storage conditions of a table, you must have *HiRDB Advanced Partitioning Option*.

The following two facilities are provided for changing the partitioning storage conditions of tables.

(1) Split facility

The split facility changes the table partitioning storage conditions, splitting data stored in a single RDAREA into multiple RDAREAs. Figure 3-17 provides an example of the split facility (using boundary value specification).

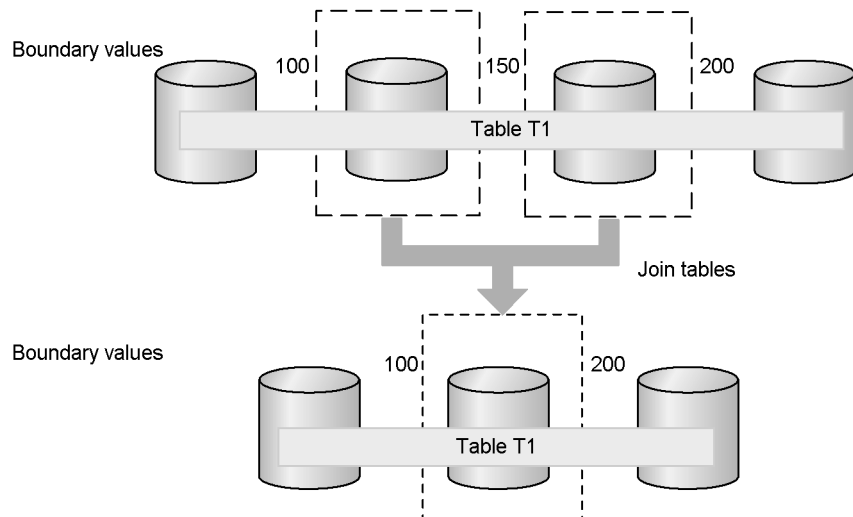
Figure 3-17: Example of the split facility (boundary value specification)



(2) Combine facility

The combine facility changes the table partitioning storage conditions, combining data stored in multiple RDAREAs into a single RDAREA. Figure 3-18 provides an example of the combine facility (using boundary value specification).

Figure 3-18: Example of the combine facility (boundary value specification)



3.3.11 Falsification-prevention table

To prevent falsification of critical data due to human error or fraudulence, you can now define a falsification-prevented table, which prevents all users, including the table owner, from updating data in the table. Table 3-5 indicates the executability of operations on falsification-prevented tables.

Table 3-5: Executability of operations on falsification-prevented tables

Operation	Falsification-prevention table	
	With deletion-prevention duration specification	Without deletion-prevention duration specification
INSERT	Y	Y
SELECT	Y	Y
Update by column (UPDATE)	Y ¹	Y ¹
Update by row (UPDATE)	N	N
DELETE	Y ²	N
PURGE TABLE	N	N
Other data manipulation SQLs	Y	Y

Legend:

Y: Can be executed.

N: Cannot be executed.

¹ Only updatable columns can be updated.

² Only data for which the deletion-prevention duration has elapsed can be deleted. If a deletion-prevention duration is not specified, the data in the table cannot be deleted.

(1) **Specification method**

To prevent falsification, you specify the `INSERT ONLY` option (falsification prevention option) in the `CREATE TABLE` definition SQL. You can also use the `INSERT ONLY` option of `ALTER TABLE` to change the definition of an existing table so that it becomes a falsification-prevented table.

You can define the following types of columns when you create or change a table definition:

- Updatable column

If you define an updatable column, you can use either of the following methods to update the data by column:

- Always update (specify `UPDATE`)
- Update only once from the null value to a non-null value (specify `UPDATE ONLY FROM NULL`)

You can define updatable columns at the following times:

- When you execute `CREATE TABLE`
- Before you execute `ALTER TABLE CHANGE INSERT ONLY`
- When you execute `ALTER TABLE ADD column-name` or `ALTER TABLE CHANGE column-name*`

* You cannot execute `ALTER TABLE CHANGE column-name` on a falsification-prevented table.

- Insert history maintenance column

By defining an insert history maintenance column, you can specify a *deletion-prevented duration*. If you do not specify the deletion-prevented duration, you will not be able to delete any of the data in the table. In addition, because you cannot execute `DROP TABLE` on such a table if it contains any data, you will not be able to delete either the table itself or the data in the table. To avoid this situation, specify the deletion-prevention duration if the data hold time has been determined, or if the data hold time can be determined.

(2) Limitations

The limitations of falsification-prevented tables and the RDAREAs in which falsification-prevented tables are stored are listed as follows. For details, see the *HiRDB Version 8 Installation and Design Guide*.

- You cannot specify the updatable column attribute for all columns and then specify the falsification prevention option.
- You cannot change existing columns to become updatable columns or change updatable columns to become regular columns.
- You must define updatable columns before you apply the falsification prevention facility.
- You cannot apply the falsification prevention facility to an existing table that already contains data. Instead, you must first unload the data from the existing table, change the table to a falsification-prevented table, and then load the data back into the table.
- You cannot delete a falsification-prevented table that contains data. The fact that you cannot delete data in a falsification-prevented table for which no deletion-prevention duration has been specified means that you cannot delete the table itself.
- You cannot use the database structure reconfiguration utility (`pdmod`) to initialize an RDAREA (`initialize rdarea`) in which a falsification-prevented table is stored.
- Do not use a replication facility (HiRDB Dataextractor or HiRDB Datareplicator) to duplicate data in or to reflect update results to a falsification-prevented table. If you do so, the data in the copy source and copy target may become inconsistent, resulting in errors.

3.3.12 Tables used in numbering

Numbering means the capability to perform other tasks using report numbers and document numbers that have been created primarily for one task. The table used for numbering is dedicated to number management only. Because multiple users need to access to this table simultaneously, it is necessary to minimize the lock processing to conduct speedy updating.

In this table, a function is provided that releases row locking as soon as the table update is complete without waiting for transaction confirmation, and that prevents subsequent rollbacks.

To define a table used for numbering, the `WITHOUT ROLLBACK` option must be specified in the `CREATE TABLE` definition SQL.

For details about numbering, see the *HiRDB Version 8 UAP Development Guide*.

3.3.13 Repetition columns

A repetition column is a column whose data values may consist of multiple elements. An *element* means each item that is stored in the same row of the repetition column. A column is defined as a repetition column in `CREATE TABLE`; the number of elements must also be defined (however, the number of elements can be increased later with the `ALTER TABLE`).

Defining a table to contain repetition columns offers the following benefits:

- If multiple tables contain identical information, the overlapping information can be eliminated, thus reducing the amount of disk space that is required.
- The need for joining multiple tables is eliminated.
- Because related data (repetition data) is stored nearby, access performance is better than when separate tables are accessed.

Figure 3-19 shows an example of a table containing repetition columns. For details about repetition columns, see the *HiRDB Version 8 Installation and Design Guide*.

Figure 3-19: Example of a table containing repetition columns

STAFF_TABLE

NAME	QUALIFICATION	SEX	FAMILY	RELATIONSHIP	SUPPORT
John Smith	CAD I	Male	Gene	Father	1
	Design		Grace	Mother	1
	CAD II		Mary	Wife	1
			Pat	Eldest son	1
			Sue	Second daughter	1
Tom Jones	CAD II	Male	Jim	Father	0
	Accounting I		Carol	Wife	1
Mark Peters	Systems analysis	Male	Helen	Mother	1
Pete Johnson		Male			

Diagram annotations: A bracket on the right side of the table groups the 'QUALIFICATION' column and the 'FAMILY'/'RELATIONSHIP'/'SUPPORT' columns. An arrow labeled 'Repetition column element' points to the 'CAD II' row in the 'QUALIFICATION' column. An arrow labeled 'Column' points to the 'Support' column.

Note: A blank cell means the null value.

(1) Example of defining a repetition column

Following is the `CREATE TABLE` SQL statement that defines the table containing the repetition column in Figure 3-19:

```
CREATE TABLE employee list
  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]);
```

(2) Operations for repetition columns

The operations listed below can be performed on a table containing repetition columns.

Retrieval using a structured repetition predicate

A structured repetition predicate is used to perform a search on multiple repetition columns in a table, where the subscripts correspond directly with the elements.

Example 1

```
SELECT NAME FROM STAFF_TABLE WHERE
    ARRAY(RELATIONSHIP, SUPPORT) [ANY]
    (RELATIONSHIP='Father' AND SUPPORT=1)
```

Updating a repetition column

The following three repetition column updating methods are provided:

- Updating an existing element (SET clause of UPDATE statement)
- Adding a new element (ADD clause of UPDATE statement)
- Deleting an existing element (DELETE clause of UPDATE statement)

To update a table containing a repetition column, *repetition-column-name* [*subscript* | *] is used to specify the element of the repetition column that is to be updated.

Example 2: Updating an existing element

```
UPDATE STAFF_TABLE SET QUALIFICATION[2]=N'Accounting-II'
WHERE NAME=N'Tom Jones'
```

Example 3: Adding a new element

```
UPDATE STAFF_TABLE ADD QUALIFICATION[*]=ARRAY{N'Systems
analysis'}
WHERE NAME=N'Tom Jones'
```

Example 4: Deleting an existing element

```
UPDATE STAFF_TABLE DELETE QUALIFICATION[2]
WHERE NAME=N'Tom Jones'
```

Other operations for repetition columns

To specify a repetition column in an SQL statement, *repetition-column-name* [*subscript*] must be used.

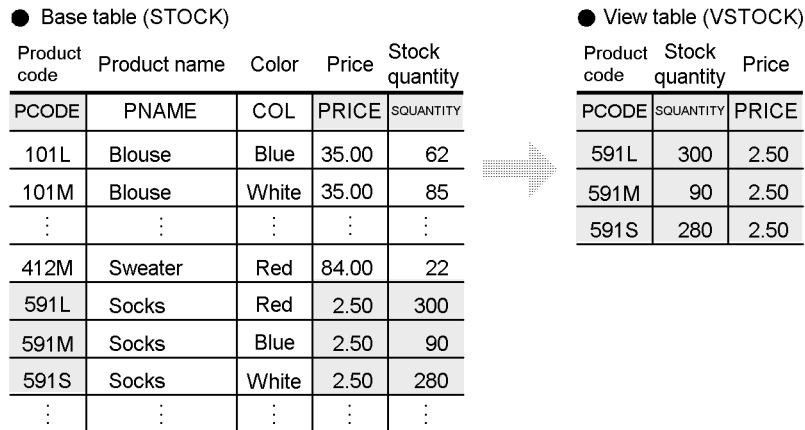
For details about the operations for tables containing repetition columns, see the

3.3.14 View table

You can create a virtual table by selecting specific rows and columns from a table that is actually stored in a database (called a *base table*). Such a virtual table is called a *view table*. By creating a view table that makes only specified columns available to the public, you can effectively protect the remainder of the data. In addition, the resulting reduction in the number of columns to be manipulated improves the operability of the table.

To create a view table, you must execute the `CREATE VIEW` of the definition SQL. Figure 3-20 shows an example of a view table.

Figure 3-20: Example of a view table

**Explanation**

The `VSTOCK` view table is created from the `STOCK` base table, and is composed of the product code (`PCODE`), stock quantity (`SQUANTITY`), and unit price (`PRICE`) columns and only those rows with `Socks` in the product name (`PNAME`) column. The columns are arranged in the order of product code, stock quantity, and unit price.

```
CREATE VIEW VSTOCK
AS SELECT PCODE, SQUANTITY, PRICE
FROM STOCK
WHERE PNAME = N'Socks'
```

If the HiRDB External Data Access facility is being used

You can create a view table from a foreign table. However, you cannot update such a view table.

3.3.15 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 the 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 the 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*. For details about shared tables, see the *HiRDB Version 8 Installation and Design Guide*.

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. This section describes the use of shared tables with a HiRDB/Parallel Server. For details about using shared tables with a HiRDB/Single Server, see (6) *Using shared tables with a HiRDB/Single Server* below.

(1) Advantages 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) Application 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.

(3) Notes on updating shared tables

When you update a shared table, all back-end servers in which RDAREAs are allocated are locked. If you execute any statement other than the `UPDATE` statement or `PURGE TABLE` statement, which do not update index key values, you must specify `IN EXCLUSIVE MODE` with the `LOCK TABLE` statement and lock all back-end servers to update the shared table. This means that any jobs that access a locked RDAREA may result in deadlock, or that global deadlock may occur among the servers.

When you use a local buffer to update a shared table, issue the `LOCK` statement before you update. If you update without issuing the `LOCK` statement and the server process terminates abnormally, the abort code `Phb3008` is output (which indicates that the unit may have terminated abnormally).

(4) Definition method

To define a shared table, specify `SHARE` with `CREATE TABLE` in the definition SQL (specify `CREATE SHARE FIX TABLE`). Note that shared tables must satisfy the following conditions:

- The shared table is a non-partitioned `FIX` table.
- The `RDAREA` storing the shared table or shared index is a shared `RDAREA` (specify `SDB` for the `-k` option of the `pdfmkfs` command).
- The `WITHOUT ROLLBACK` option is not specified.
- The table is not a referencing table defined with a referential constraint.

(5) Limitations

The following limitations apply to using a shared table:

- You cannot search a shared table while the `LOCK TABLE` statement is being executed with `IN EXCLUSIVE MODE` specified.
- You cannot use the `ASSIGN LIST` statement to create a list of shared tables.
- You cannot specify a shared table as a replication copy destination

(6) Using shared tables with a HiRDB/Single Server

- You cannot define shared `RDAREAs` in a HiRDB/Single Server; therefore, you must store shared tables and shared indexes in regular user `RDAREAs`. When you do this, you must provide separate user `RDAREAs` for shared tables and shared indexes; you must not store shared tables in user `RDAREAs` that store regular tables and indexes that are not shared.
- When you migrate a HiRDB/Single Server to a HiRDB/Parallel Server, do not use the database structure modification utility (`pdmod`) while shared tables or shared indexes are defined. For details about the migration procedure, see the *HiRDB Version 8 Installation and Design Guide*.

For details about the differences with a HiRDB/Parallel Server, see the *HiRDB Version 8 Installation and Design Guide*.

3.4 Indexes

An index is a search-basis key that is assigned to one or more columns of a table for the purpose of making retrievals from that table. An index can be either a B-tree index provided by HiRDB or a plug-in index provided by a plug-in. B-tree indexes are explained here; for an explanation of plug-in indexes, see Section *10.4 Preparations for using plug-ins in HiRDB*.

When an index is defined, the table columns on which the index is to be based must be defined; columns that will improve retrieval performance should be selected.

3.4.1 Basic structure of an index

An index consists of a key and key values. A *key* is the column name of an indexed column. The values in the column are called the *key values*. Creating an index for a column that will be used as the basis for retrievals from the table will improve the table's retrieval performance.

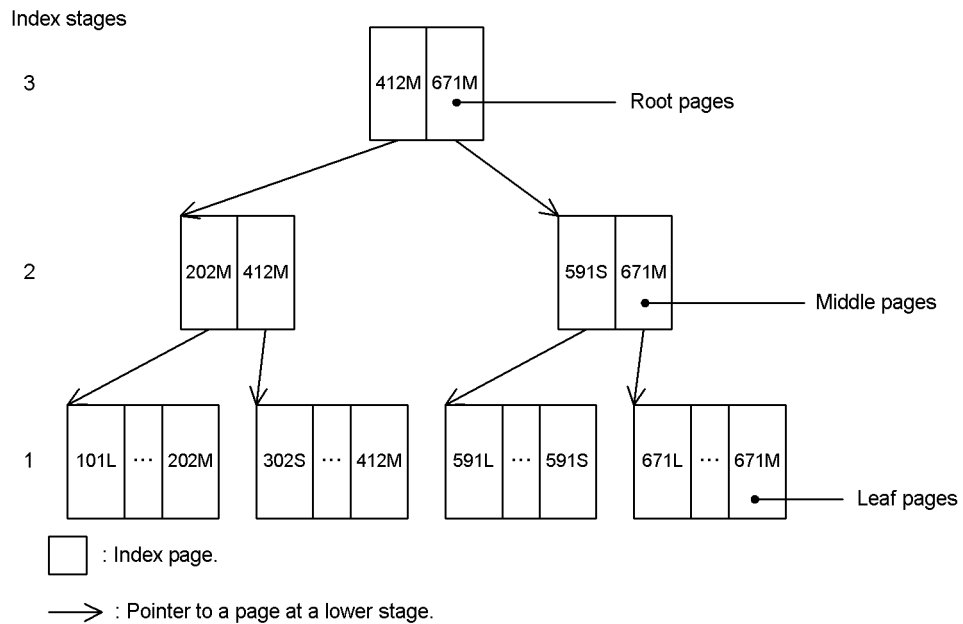
It is recommended that indexes be created for the following types of columns:

- Column used as a condition for narrowing the range of data to be searched
- Column used as a condition for joining tables
- Column used as a condition for sorting or grouping data
- Component column that defines a referential constraint (foreign key)

Indexes provided by HiRDB have a B-tree structure. The highest stage of an index in a B-tree structure is called a *root page*; the lowest stage is called a *leaf page*; a page in the middle is called a *middle page*. Root pages and middle pages point to pages at lower stages. A leaf page has a key value for each index page and its address.

Figure 3-21 shows the B-tree structure of an index.

Figure 3-21: B-tree structure of an index



Explanation

This is a B-tree structure index. The index is created in Step 3 on the basis of the *product code* column, PCODE is the key, and 101L to 671M are key values.

(1) *Single column index and multicolumn index*

An index can be based either on a single column or on multiple columns (multicolumn index). A single column index is created by indexing the values in a single column of a table. A single column index is appropriate for retrievals that require a single column as the key. A multicolumn index is created by indexing the values in more than one column of a table. A multicolumn index is appropriate for the following purposes:

- To narrow the data that is to be retrieved to data that satisfies multiple conditions
- To group or sort the data that has been narrowed using a retrieval condition
- When some of the same columns are used in multiple multicolumn indexes created for a table

(2) *Optimizing based on cost*

When multiple indexes have been created for a table, HiRDB selects and uses the index that has the least access cost and that it evaluates as being optimal for the conditions specified for the retrieval. This process in which HiRDB selects an index

based on an evaluation of optimality is called *optimizing based on cost*. HiRDB evaluates the following access costs:

- Hit rate based on the specified retrieval conditions
- Number of input/output processes required for SQL processing
- CPU load required for SQL processing

HiRDB performs optimizing based on cost in order to improve table retrieval performance. Even when an SQL with retrieval conditions specified is executed, table retrieval performance does not deteriorate. This means that it is possible to create UAPs without being concerned about index access priority. However, in order for HiRDB to use the optimal index, indexes must have been defined for those columns for which retrieval conditions are specified.

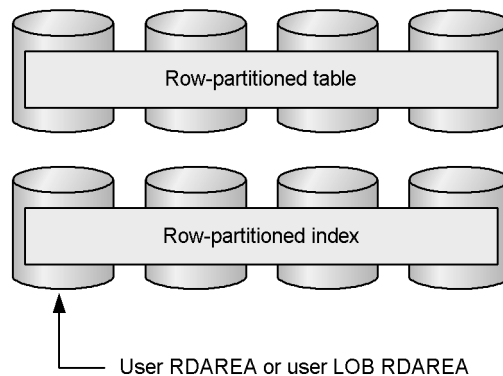
(3) Index definition

The `CREATE INDEX` definition SQL is used to specify the columns in a table that are to be indexed. For details about design considerations related to indexes, such as index definition for retrieval performance improvement, see the *HiRDB Version 8 Installation and Design Guide*.

3.4.2 Index row partitioning

The process of row-partitioning an index for a table that has been row-partitioned is called *index row partitioning*. The index must be partitioned so that it corresponds to the row-partitioned table in terms of the multiple RDAREAs in which the table is stored. Such an index is called a *row-partitioned index*. When an index is row-partitioned, each user RDAREA or user LOB RDAREA can be handled independently during index batch creation or re-creation. Figure 3-22 illustrates index row partitioning.

Figure 3-22: Index row partitioning



Explanation

You use the `CREATE INDEX` of the definition SQL to specify the RDAREAs in which a row-partitioned index is to be stored.

(1) Example of index row partitioning: HiRDB/Single Server

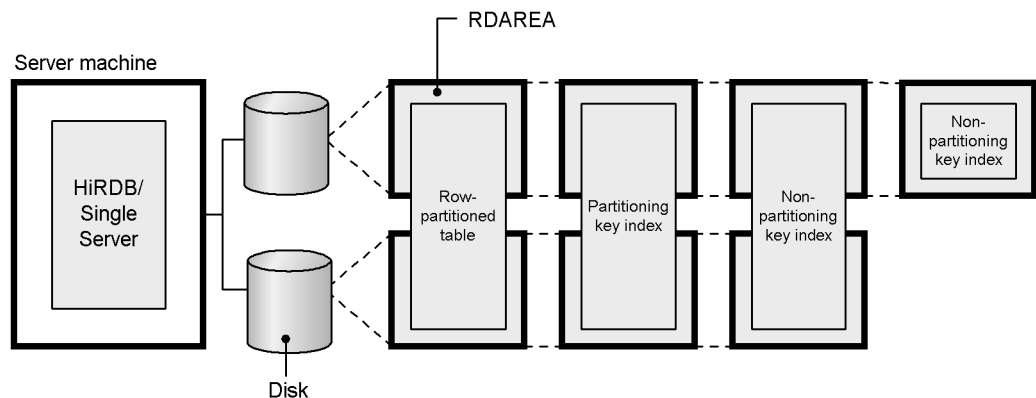
When you have a row-partitioned table, you need to know whether an index being defined is a partitioning key index or a non-partitioning key index. For details about partitioning key indexes and non-partitioning key indexes, see Section (3) *Partitioning key index and non-partitioning key index* as follows. Table 3-6 shows guidelines for row-partitioning an index by index type.

Table 3-6: Guidelines for row-partitioning an index: HiRDB/Single Server

Index type	Partitioning guidelines
Partitioning key index	Row-partition the index so that it corresponds to the row-partitioned table.
Non-partitioning key index	We recommend that the index not be row-partitioned, because row-partitioning the index can reduce the performance of index-based searches. However, if the table contains an extremely large amount of data, you may wish to consider row-partitioning it. Row-partitioning an index allows the system to manage the table storage RDAREAs and the index storage RDAREAs on a 1-to-1 basis, which will improve the operability of the utilities. For example, if data is loaded by RDAREA, or if RDAREAs are reorganized without the index having been row-partitioned, you will have to batch-create an index after you have loaded the data or reorganized the RDAREAs. If the index is row-partitioned, you will not have to batch-create an index after data loading or RDAREA reorganization.

Figure 3-23 shows an example of row-partitioning an index for a HiRDB/Single Server.

Figure 3-23: Example of row-partitioning an index: HiRDB/Single Server



Explanation

- To prevent disk access contention, the RDAREAs storing the partitioned table and the index are allocated to different disks.
- The partitioning key index is row-partitioned.
- If better performance is important, you would not row-partition non-partitioning key indexes.
- If better operability is important, you would row-partition non-partitioning key indexes.

(2) Example of index row partitioning: HiRDB/Parallel Server

In the case of a HiRDB/Parallel Server, the index row partitioning guidelines depend on whether the table is partitioned within a server or among servers.

(a) Table is row-partitioned within a server

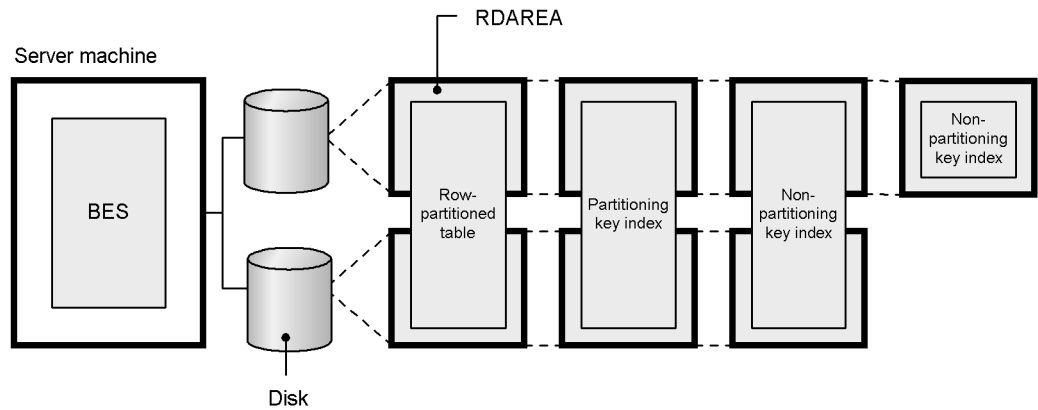
You need to know whether the index being defined is a partitioning key index or a non-partitioning key index. For details on partitioning key indexes and non-partitioning key indexes, see Section (3) *Partitioning key index and non-partitioning key index* below. Table 3-7 shows guidelines for row-partitioning an index by index type.

Table 3-7: Guidelines for row-partitioning an index: HiRDB/Parallel Server)

Index type	Partitioning guidelines
Partitioning key index	Row-partition the index so that it corresponds to the row-partitioned table.
Non-partitioning key index	We recommend that the index not be row-partitioned, because row-partitioning the index can reduce the performance of index-based searches. However, if the table contains an extremely large amount of data, you may wish to consider row-partitioning it. Row-partitioning an index allows the system to manage the table storage RDAREAs and the index storage RDAREAs on a 1-to-1 basis, which will improve the operability of the utilities. For example, if data is loaded by RDAREA, or if RDAREAs are reorganized without the index having been row-partitioned, you will have to batch-create an index after you have loaded the data or reorganized the RDAREAs. If the index is row-partitioned, you will not have to batch-create an index after data loading or RDAREA reorganization.

Figure 3-24 shows an example of row-partitioning an index within a server.

Figure 3-24: Example of row-partitioning an index within the server



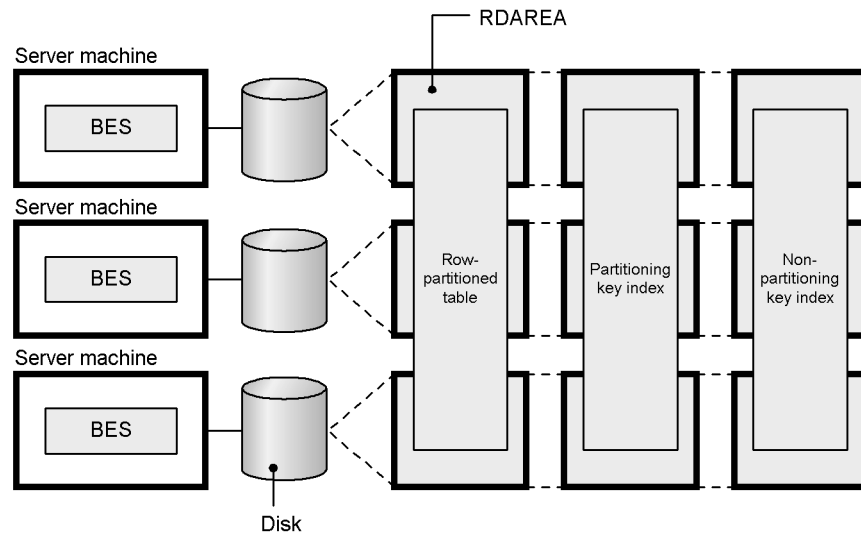
Explanation

- To prevent disk access contention, allocate the RDAREAs storing the partitioned table and the index to different disks.
- Row-partition a partitioning key index.
- If better performance is important, do not row-partition non-partitioning key indexes.
- If better operability is important, row-partition non-partitioning key indexes.

(b) Table is row-partitioned among servers

You must row-partition the index so that it corresponds to the row-partitioned table. You need not know whether the index being defined is a partitioning key index or a non-partitioning key index. Figure 3-25 shows an example of row-partitioning an index among servers.

Figure 3-25: Example of row-partitioning an index among servers



Explanation

- To prevent disk access contention, the RDAREAs storing the partitioned table and the index are allocated on different disks.
- Both partitioning key indexes and non-partitioning key indexes are row-partitioned.

(3) Partitioning key index and non-partitioning key index

An index that satisfies a particular condition becomes a partitioning key index. An index that does not satisfy the condition is called a non-partitioning key index. The condition is explained below. The condition is whether the table involved is partitioned on the basis of single-column partitioning or multicolumn partitioning. When only one column is used in the table partitioning condition, the partitioning is said to be *single-column partitioning*; when multiple columns are used in the table partitioning condition, the partitioning is said to be *multicolumn partitioning*.

(a) Single-column partitioning

An index satisfying one of the following conditions is a partitioning key index:

Conditions

- Single-column index defined in the column (partitioning key) specifying the storage conditions when the table is row-partitioned.
- Multicolumn index with the column (partitioning key), defined as the first member column, specifying the storage conditions when the table is

row-partitioned.

Figure 3-26 shows an example (based on the `STOCK` table) of an index that becomes a partitioning key index.

Figure 3-26: Partitioning key index: Single-column partitioning

STOCK				
PCODE	PNAME	COL	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29

↑
Column specified for the partitioning condition (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 `PCODE` column, which is the partitioning key, is used as an index, the index becomes a *partitioning key index*. If any other column is used as an index, the resulting index becomes a non-partitioning key index.
2. Specifying the `PCODE` column, which is the partitioning key, as the first constituent column of the index makes the resulting multicolumn index a *partitioning key index*.
3. Specifying the `PCODE` column, which is the partitioning key, as a column other than the first constituent column of the index makes the resulting multicolumn index a *non-partitioning key index*.

(b) Multicolumn partitioning

An index satisfying the following condition is a partitioning key index:

Condition

- The index is created on the basis of multiple columns, beginning with the partitioning key and containing all the columns specified for partitioning from the beginning and without any change in their order.

Figure 3-27 shows an example (based on the `STOCK` table) of an index that becomes a partitioning key index.

Figure 3-27: Partitioning key index: Multicolumn partitioning

STOCK				
PCODE	PNAME	COL	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29

Columns specified for the partitioning condition (partitioning key)

```

CREATE TABLE STOCK...
HASH HASH1 BY PCODE,PRICE...
  
```

Explanation

```

CREATE INDEX A12 ON STOCK (PCODE ASC, PRICE DESC) 1
CREATE INDEX A12 ON STOCK
                (PCODE ASC, PRICE DESC, 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. All partitioning keys (`PCODE` and `PRICE` columns) are specified, and these keys are specified in the same order as in the table definition. Therefore, this multicolumn index is a *partitioning key index*.
2. All partitioning keys (`PCODE` and `PRICE` columns) are specified, and these keys are specified in the same order as in the table definition. Therefore, this multicolumn index is a *partitioning key index*.
3. All partitioning keys (`PCODE` and `PRICE` columns) are specified, but these keys are specified in an order that differs from the table definition order. Therefore, this multicolumn index is a *non-partitioning key index*.
4. All partitioning keys (`PCODE` and `PRICE` columns) are specified, but these keys are specified in an order that differs from the table definition order. Therefore, this multicolumn index is a *non-partitioning key index*.

3.4.3 Index page splitting

When an attempt is made to add a key to an index page that does not contain any more free space, HiRDB allocates space for a new index page and splits the index information on the existing page. It then transfers the second portion of the index information to the new page. This is called *index page splitting*.

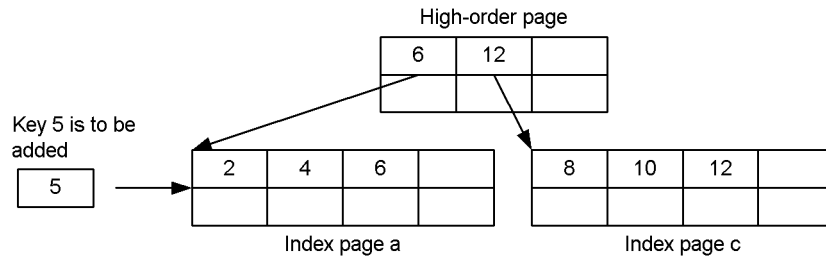
An index page can be split so that the existing index information is divided evenly in half, or it can be split unevenly (unbalanced index splitting).

(1) Even-split index page splitting

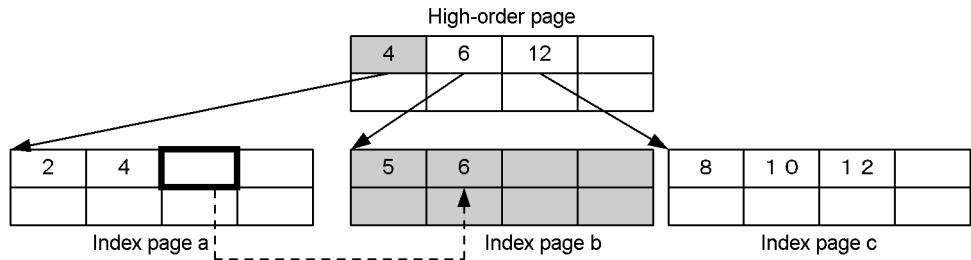
Figure 3-28 shows an example of even-split index page splitting.

Figure 3-28: Example of even-split index page splitting

1. B-tree structure of index before index page splitting



2. Index page splitting



: Locations where the index structure was changed by index page splitting.

: Key that was moved to another page by index page splitting. (Data is divided evenly between index pages a and b.)

Explanation

The addition of key 5 to index page a causes index page splitting. Key 5, as well as key 6 that follows, are moved to a new index page b so that the keys are distributed evenly.

(2) Unbalanced index split

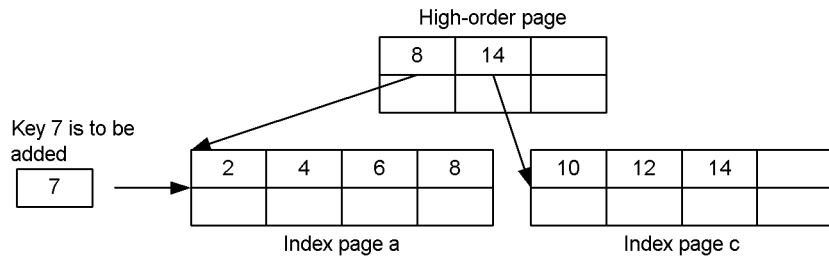
Adding many contiguous key values can degrade an index page's data storage efficiency. In such a case, it is advisable to split the index information on the index page unevenly; this is called *unbalanced index splitting*.

When unbalanced index splitting is used, the place in the index information at which

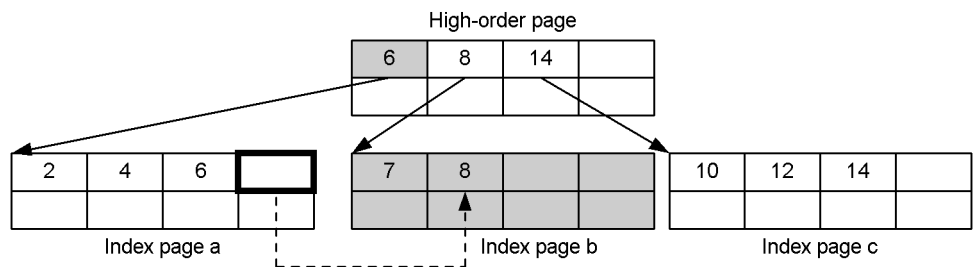
the index page is split is determined by the location of the key value that is being added. If the location of the new key value is in the top part of the index page, the system assumes the subsequent keys will also be added in the top part. Therefore, it stores the top part in the left-hand page, using the key value that is one greater than the added key value as the partitioning position. If the location of the new key value is in the bottom part of the index page, the system assumes the subsequent keys will also be added in the bottom part. Therefore, it stores the bottom part in the right-hand page, using the key value of the added key as the partitioning position. Figure 3-29 shows an example of an unbalanced index split in which a key is added to the bottom part of an index page.

Figure 3-29: Example of unbalanced index split

1. B-tree structure of index before unbalanced index split



2. Unbalanced index split



: Locations where the index structure was changed by unbalanced index split.



: Key that was moved to another page by unbalanced index split.
(Because the key was added to the second half of index page a, data is stored to allocate more free space in index page b.)

Explanation

Key 7 is added to the second half of index page a, so key 7, as well as key 8 that follows, are moved to index page b on the right side.

Application criteria

Use of unbalanced index splitting is advisable in the following cases, because it can improve the data storage efficiency and reduce the number of times index splitting is necessary:

- The index has a uniform degree of duplication of key values
- The index's key values are roughly uniform in size
- Index is one to which consecutive intermediate key values are added frequently

Specifying an unbalanced index split

For an unbalanced index split, you specify the `UNBALANCED SPLIT` option in the `CREATE INDEX` or `CREATE TABLE` of the definition SQL. For details on the unbalanced index split option, see the *HiRDB Version 8 System Operation Guide*.

3.4.4 Exception key value

All data values in a column for which an index is defined, even null values, are incorporated into the index as index key values. However, null-value keys in an index serve no purpose, so their presence in the index is wasteful of space. Therefore, if an entire column in an index is redundant null values, the null value can be specified as an exception key value for the index. Assigning an exception key value to an index has the following benefits:

Expected benefits

1. Because null-value keys are not used in the index, the size of the index is reduced.
2. Overhead for index maintenance (CPU time, number of I/O operations, number of lock requests, frequency of deadlock, etc.) during row insertion, deletion, and updating operations is reduced.
3. If the only retrieval condition for a column of an index that has the null value as an exception key value is `IS NULL`, that index will not be used. Consequently, retrieval performance would be improved in the following situation:
 - When input/output processes have occurred on the same page because data pages were accessed randomly using an index that contained many redundant null values.

Specifying an exception key value

To set an exception key value, you specify the `EXCEPT VALUES` option in the `CREATE INDEX` of the definition SQL. For details on index exception key values, see the *HiRDB Version 8 Installation and Design Guide*.

3.4.5 Defining an index for a table that contains data

Creation of an index for a table that contains a large amount of data (execution of the `CREATE INDEX`) is a time-consuming process that delays execution of other definition SQL statements.

The `EMPTY` option in the `CREATE INDEX` enables you to define an index without actually creating the index entries. Such an index is called an *unfinished index*. Because no index entries are created, execution of the `CREATE INDEX` terminates immediately and other definition SQL statements can be executed.

Because the index entries are not created, it is not possible to perform searches using an unfinished index or to update the columns in the table that define the unfinished index (an SQL error may result). The index entries can be created by executing the index re-creation facility (`-k ixrc`) of the database reorganization utility (`pdroorg`). When the index entries are created, the index is released from unfinished index status. Deleting all data in the table with the `PURGE TABLE` statement also resets indexes to the table from unfinished index status.

For details on the use of the `EMPTY` option, see the *HiRDB Version 8 System Operation Guide*.

3.4.6 Index key value no-lock

Index key value no-lock is the process by which a table is accessed by locking only the table data without locking its index key values.

The index key value no-lock option prevents the following problems:

- Deadlock between data-updating and index-search processes
- Unreasonable delayed access to data with the same key
- Unreasonable delayed access to data with different keys

When the index key value no-lock option is used, search processing based on an index does not lock the index key values. In the case of table updating (inserting rows, deleting rows, updating column values), the index key values for any index that is defined on the basis of a column being updated are not locked.

(1) Application criteria

The index key value no-lock option can be used for all operations. However, whether or not it is to be used should be determined by taking into consideration the operation of unique indexes, the presence of any residual entries, and the size of the index log. For an explanation of uniqueness constraint assurance processing for unique indexes and residual entries, see Section (4) *Notes*. For the implications of the size of the index log when the index key value no-lock option is used, see the explanation on estimating the size of the index log and the number of locked resources in the *HiRDB Version 8 Installation and Design Guide*.

(2) Specification

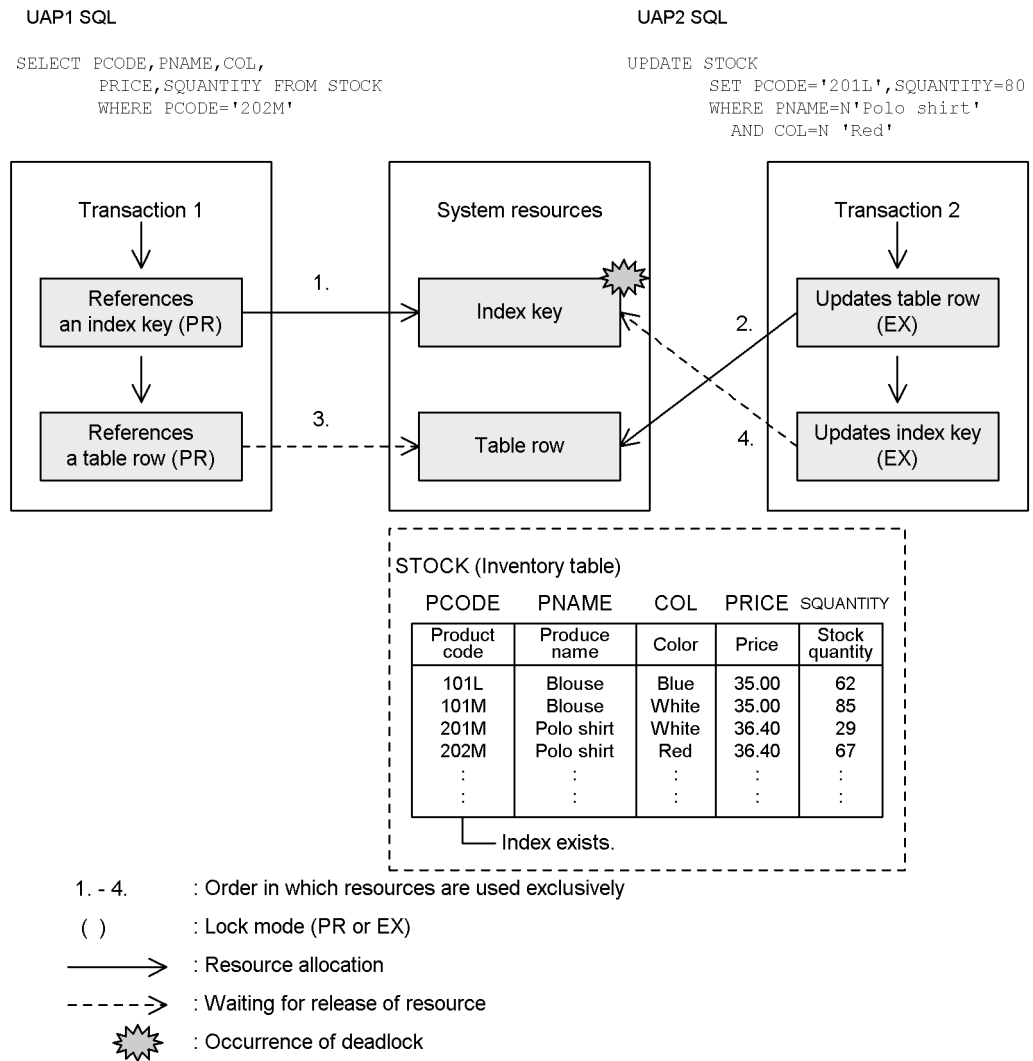
To use the index key value no-lock option, specify `NONE` in the `pd_indexlock_mode` operand in the system common definition.

If the value specified for the `pd_inner_replica_control` system definition operand is greater than 1, `NONE` is assumed for the `pd_indexlock_mode` system definition operand, regardless of the value specified for `pd_indexlock_mode`.

(3) Example of preventing deadlock by using the index key value no-lock option

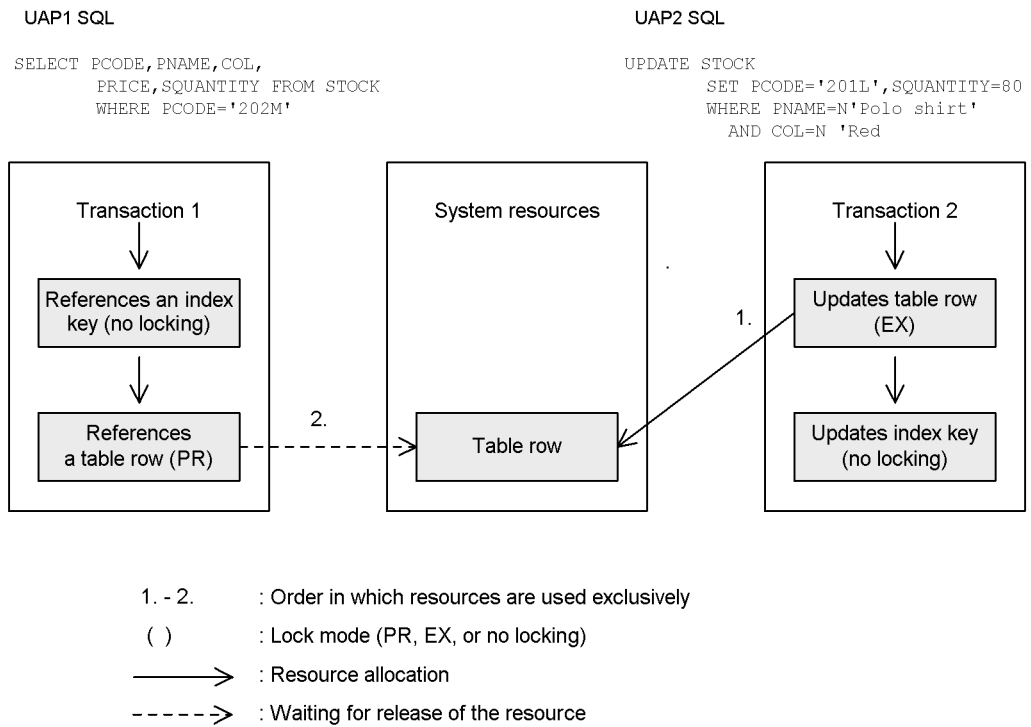
This section explains how to prevent deadlock by using the index key value no-lock option. Figure 3-30 shows an example of the deadlock described below. For a description of the lock mode, see Section 6.10.2 *Lock modes*.

Figure 3-30: Example of deadlock when the index key value no-lock option is not used



Specifying the index key value no-lock option prevents the type of deadlock shown in Figure 3-31. Figure 3-31 shows an example in which deadlock is prevented because the index key value no-lock option is being used.

Figure 3-31: Example of preventing deadlock by using the index key value no-lock option



(4) Notes

(a) Operation of uniqueness constraint assurance processing for unique indexes

In the case of a table for which the uniqueness constraint is specified, the operation of the uniqueness constraint assurance processing that is performed during addition or updating of rows depends on whether or not the index key value no-lock option is used. Uniqueness constraint assurance processing is the processing by which locking is applied during insertion of row data or updating of column values in order to ensure the uniqueness of a key value being added; key value uniqueness is determined by checking the index (unique index) to see whether or not the keyed data to be added already exists. If uniqueness constraint assurance processing locates an index key entry that uses the same key, an error occurs. Even if another party's transaction that is manipulating the index key is incomplete, and there is a possibility of a rollback being performed, an error occurs without a lock check being performed.

If you are performing table data insertion or updating processing with the uniqueness

constraint specified and you want continuation of processing to take priority over waiting, you should apply the index key value no-lock option. If the insertion or updating processing must take priority, even if doing so will create delay, you should not apply the index key value no-lock option.

(b) Residual entries in a unique index

When the index key value no-lock option is used, either lock-wait or deadlock can arise in the unique index. With a unique index in the index key value no-lock option, any index key prior to the execution of the `DELETE` or `UPDATE` statement is left intact, instead of being deleted from the index. This is to maintain the uniqueness of the index. The remaining index key is called a residual entry. Although residual entries are deleted at an appropriate time when the transaction has been settled, if the `INSERT` or `UPDATE` statement is executed on the same key as a residual entry, it is possible that an unexpected delay or deadlock will result.

These problems can be prevented by creating the UAP so that it will not update any columns that are subject to the uniqueness constraint.

(5) *Deadlocks that cannot be prevented even with the index key value no-lock option*

A deadlock between index keys can arise depending on the order in which accesses are made by a UAP. To prevent this, you must create UAPs so that they will not update columns that are subject to the uniqueness constraint.

3.5 Expansion into an object relational database

By applying object oriented concepts to a relational database model, you can build an *object relational database management system*.

HiRDB is capable of handling data with a complex structure, such as multimedia data, as well as operations performed on such data, by integrating this type of data as objects and managing them in a database. This means that SQL statements can be used to manipulate multimedia data, as in the case of a conventional relational database. For example, you can manage and manipulate the following types of multimedia data:

- SGML/XML structured text data

You can perform operations such as text searches and highlighting of retrieval hit positions by means of structure specifications.

- Image data

You can perform operations such as calculation of levels of similarities (the extent to which a given image is similar to another image) by means of quantifying image features.

The SGML/XML structured text data and image data manipulation functions are provided as plug-ins. For details on using plug-ins, see Chapter 10. *Plug-ins*.

3.5.1 Abstract data types

By using *abstract data types*, which are user-defined types, together with various routines, you can uniquely define and use data with complex structures and perform operations on such data. When you define a column as having an abstract data type, you can conceptualize and model its data based on object-oriented concepts. In addition, by applying object-oriented software development techniques, you can reduce the workload for database design, UAP development, and maintenance.

HiRDB makes it possible to use definition SQL statements to define unique abstract data types and their structures. An abstract data type can be handled as a column data type in the same way as HiRDB's predefined data types, such as the numeric type and the character type. Operations on the values of abstract data types can also be defined as routines by using definition SQL statements. In a UAP, you can use a routine to code complex operations on abstract data types in SQL.

Abstract data types, routines, and their characteristic concepts are explained below by means of examples.

(1) Defining an abstract data type

An example of managing and manipulating employee information in a database using an abstract data type is explained below.

Let's assume that employee information consists of such items as name, sex, date of employment, position, and salary. Let's also assume that image information such as an ID photo is also part of the employee information. Calculation of service years is one possible operation on employee information.

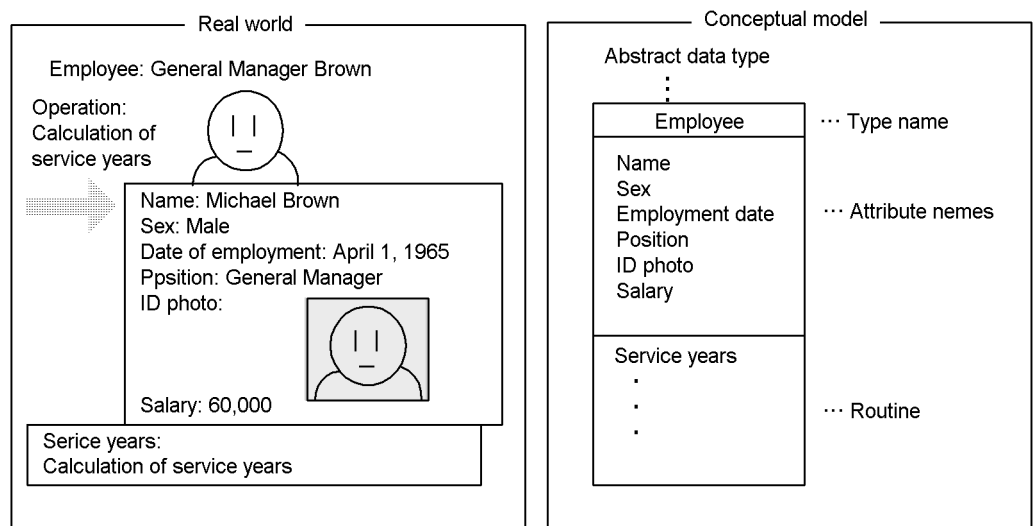
When this information is handled in a database, the abstract concept of *employee* in a data model can be considered to consist of attributes such as *name*, *sex*, *employment date*, *position*, *ID photo*, and *salary*, all of which are common to this concept. An operation such as *calculation of service years* for an employee can also be considered to be an operation that indicates an employee characteristic.

Employee can capture all these characteristics (attributes and operations) in a single concept.

In HiRDB, you can address as abstract concepts and handle in a database objects that exist in the real world, and then use abstract data types to define such concepts as data types.

Figure 3-32 shows a conceptual model based on real-world information and an abstract data type.

Figure 3-32: Conceptual model based on real-world information and an abstract data type



An abstract data type can be defined in a database using the `CREATE TYPE` definition SQL shown as follows.

```
CREATE TYPE t_employee (
    Name          CHAR(16),
    Sex           CHAR(1),
    employment_date DATE,
```

3. Database Logical Structure

```
position      CHAR(10),
id_photo      BLOB(64K),
salary        INTEGER,
.....
FUNCTION      service-years ( p t_employee )
  RETURNS     INTEGER
  BEGIN
    DECLARE service_years INTERVAL YEAR TO DAY;
    SET service_years = CURRENT_DATE -
p..employment_date;
    RETURN YEAR(service_years);
  END,
  .....
)
```

In this way, the user can use an abstract data type to define new data types by specifying attributes and operations.

(2) Abstract data type as a data type

An abstract data type can be handled in the same way as the HiRDB system default data types, such as the numeric type and the character type. For example, the table STAFF_TABLE can be defined by the definition SQL shown below using the abstract data type t_employee as a column data type:

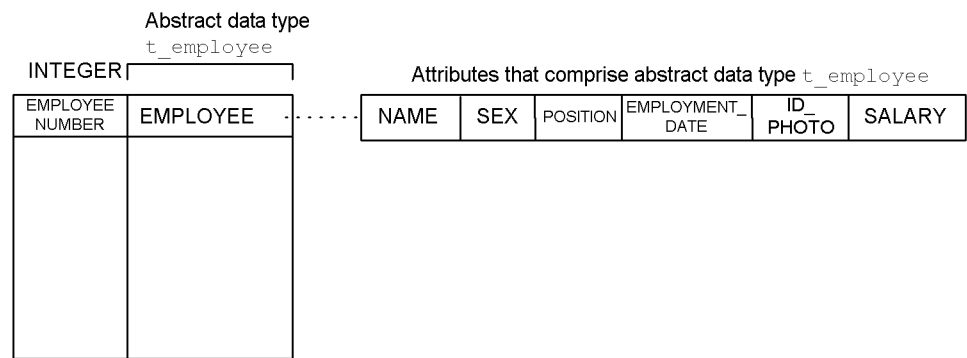
```
CREATE TABLE STAFF_TABLE (
  employee_number  INTEGER,
  employee         t_employee
                  ALLOCATE(id_photo IN(lobarea))
)
```

When the BLOB type is to be used as one of the attributes of an abstract data type, the user LOB RDAREA in which the data is to be stored must be specified with the ALLOCATE option in CREATE TABLE. In the example above, id_photo, which is an attribute of t_employee, is of the BLOB type, so ALLOCATE is used to store it in user LOB RDAREĀ lobarea.

Figure 3-33 shows a table called STAFF_TABLE, for which an abstract data type is defined.

Figure 3-33: STAFF_TABLE for which an abstract data type is defined

Table: STAFF_TABLE

**(3) Encapsulation**

When an abstract data type is used, an application can handle the abstract data type values without knowing the detailed configuration of the individual attributes or the installed routines; it does this by using routines declared in that abstract data.

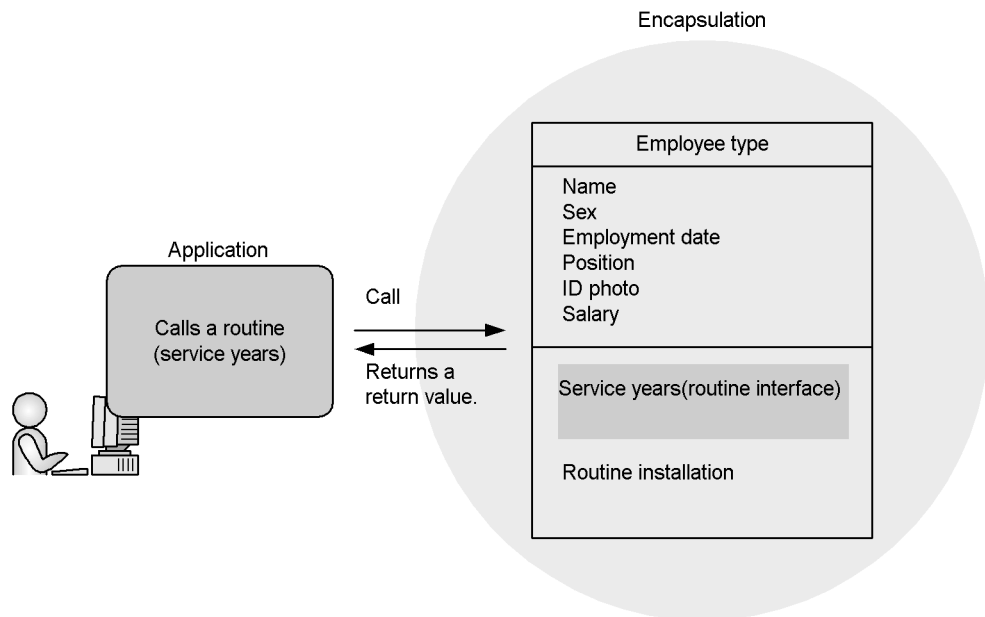
For example, it is possible to manipulate t_employee type values using the following data manipulation SQL:

```
SELECT employee_number, employee_name,
       service-years(employee)
FROM   STAFF_TABLE
```

Handling of values based on an abstract data type and using only an external interface without being concerned about the information in the values is called encapsulation.

Figure 3-34 shows encapsulation.

Figure 3-34: Encapsulation



(4) Abstract data type values

(a) Value generation

By executing a function without arguments that is recognized by the same name as an abstract data type, HiRDB can generate values for that abstract data type.

For example, for the `t_employee` type, `t_employee` type values can be generated using the function `t_employee()`.

A function that generates abstract data type values is called a constructor function.

```
BEGIN
    DECLARE p t_employee; ... Start of SQL procedure.
    SET p = t_employee(); ... Declares a t_employee type variable.
    SET p..name = 'Michael Brown' ... Generates a t_employee type value
    ... and substitutes in the variable.
    RETURN p; ... Setting of attribute value
    ... through component
    ... specification.
END ... Returns the function's return value
... (returns a t_employee type value).
... End of SQL procedure.
```

When `CREATE TYPE` is used to define an abstract data type in the database, HiRDB automatically defines a function such as `t_employee()` that has the same name as the data type but has no argument. Such a function is called the default constructor

function.

(b) User-defined constructor function

The user may also define a constructor function.

A constructor function is defined by defining in a `CREATE TYPE` routine declaration a function that has the same name as the abstract data type to be defined and that uses the abstract data type as the return value type.

```
CREATE TYPE t_employee (
    name          CHAR(16),
    sex           CHAR(1),
    employment_date DATE,
    position      CHAR(10),
    id_photo      BLOB(64K),
    salary        INTEGER,

    FUNCTION t_employee(
        p_name      CHAR(16),
        p_sex       CHAR(1),
        p_employment_date DATE,
        p_position  CHAR(10),
        p_id_photo  BLOB(64K),
        p_salary    INTEGER)
        RETURNS    t_employee

    BEGIN
        DECLARE    d_employee t_employee;
        SET        d_employee = t_employee();
        SET        d_employee..name      = p_name;
        SET        d_employee..sex       = p_sex;
        SET        d_employee..employment_date
                = p_employment_date;
        SET        d_employee..position  = p_position;
        SET        d_employee..id_photo  = p_id_photo;
        SET        d_employee..salary    = p_salary;

        RETURN    d_employee;
    END,
    ...
)
```

For example, using the user-defined constructor function `t_employee()` and the following data manipulation SQL, values can be generated and stored in a database:

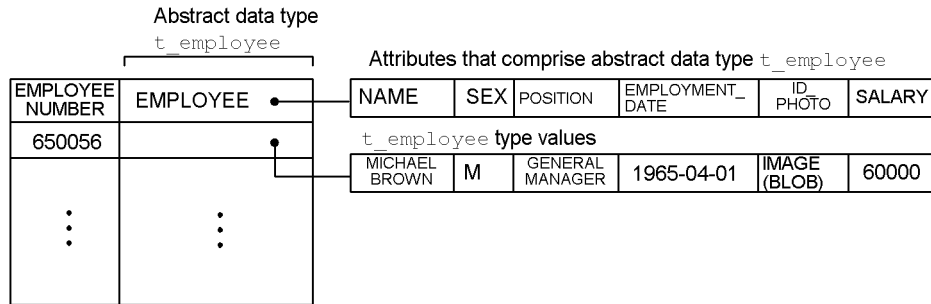
```
INSERT INTO STAFF_TABLE
VALUES (
    650056, t_employee(:name AS CHAR(16),
                      :sex AS CHAR(1),
                      :yrs AS DATE,
                      :post AS CHAR(10),
```

```
:picture AS BLOB(64K),
:salary AS INTEGER)
)
```

Figure 3-35 shows a table called STAFF_TABLE for which values are generated by constructor function t_employee() and inserted as column values.

Figure 3-35: STAFF_TABLE for which values are generated using a constructor function

Table: STAFF_TABLE



(5) Abstract data type null values

Null values can also be applied to an abstract data type, in the same manner as with the HiRDB system default types. For example, executing the following data manipulation SQL for the aforementioned STAFF_TABLE places null values in the EMPLOYEE column:

```
INSERT INTO STAFF_TABLE(employee_number)
VALUES (650056)
```

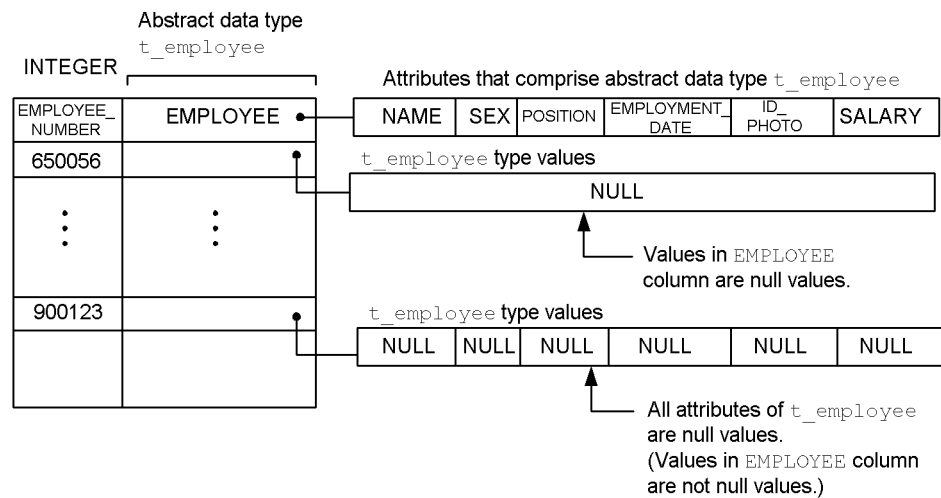
On the other hand, executing the following data manipulation SQL changes all t_employee type attribute values to null values (the column values of the abstract data type in which all attribute values are null values are regarded as values other than null values):

```
INSERT INTO STAFF_TABLE (900123, t_employee ())
```

Figure 3-36 shows the handling of null values in STAFF_TABLE for which an abstract data type is defined.

Figure 3-36: Handling of null values in STAFF_TABLE for which an abstract data type is defined

Table: STAFF_TABLE



For example, executing the following data manipulation SQL retrieves the employee numbers of those employees whose values in the EMPLOYEE column are not null values (therefore, the employee numbers of those employees who have all of the t_employee type attributes as null values will not be retrieved):

```
SELECT employee_number FROM STAFF_TABLE
WHERE employee IS NOT NULL
```

Retrieval result:

900123

(6) Manipulation of abstract data type values

Let's plan an operation for calculating an employee's service years with the company.

Operations for abstract data type values can be defined in HiRDB by using a routine declaration in CREATE TYPE. For example, an operation such as for calculating service years and compensation rate can be defined with the following definition SQL:

```
CREATE TYPE t_employee (
    name          CHAR(16),
    sex           CHAR(1),
    employment_date DATE,
    position      CHAR(10),
    id-photo     BLOB(64K),
    salary        INTEGER,
    .....
```

3. Database Logical Structure

```
FUNCTION service_years ( p t_employee )
  RETURNS INTEGER
  BEGIN
    DECLARE service_years INTERVAL YEAR TO DAY;
    SET service_years
      = CURRENT_DATE - p..employment date;
    RETURN YEAR(service_years);
  END,
  .....
)
```

In this way, routines defined for an abstract data type can be used for abstract data type values. For example, an SQL for retrieving the employee number and employee name of each employee with a service years value of 10 or more years can be described as follows:

```
SELECT employee number, employee..name, service_years (employee)
  FROM STAFF_TABLE
  WHERE service_years(employee) >= 10
```

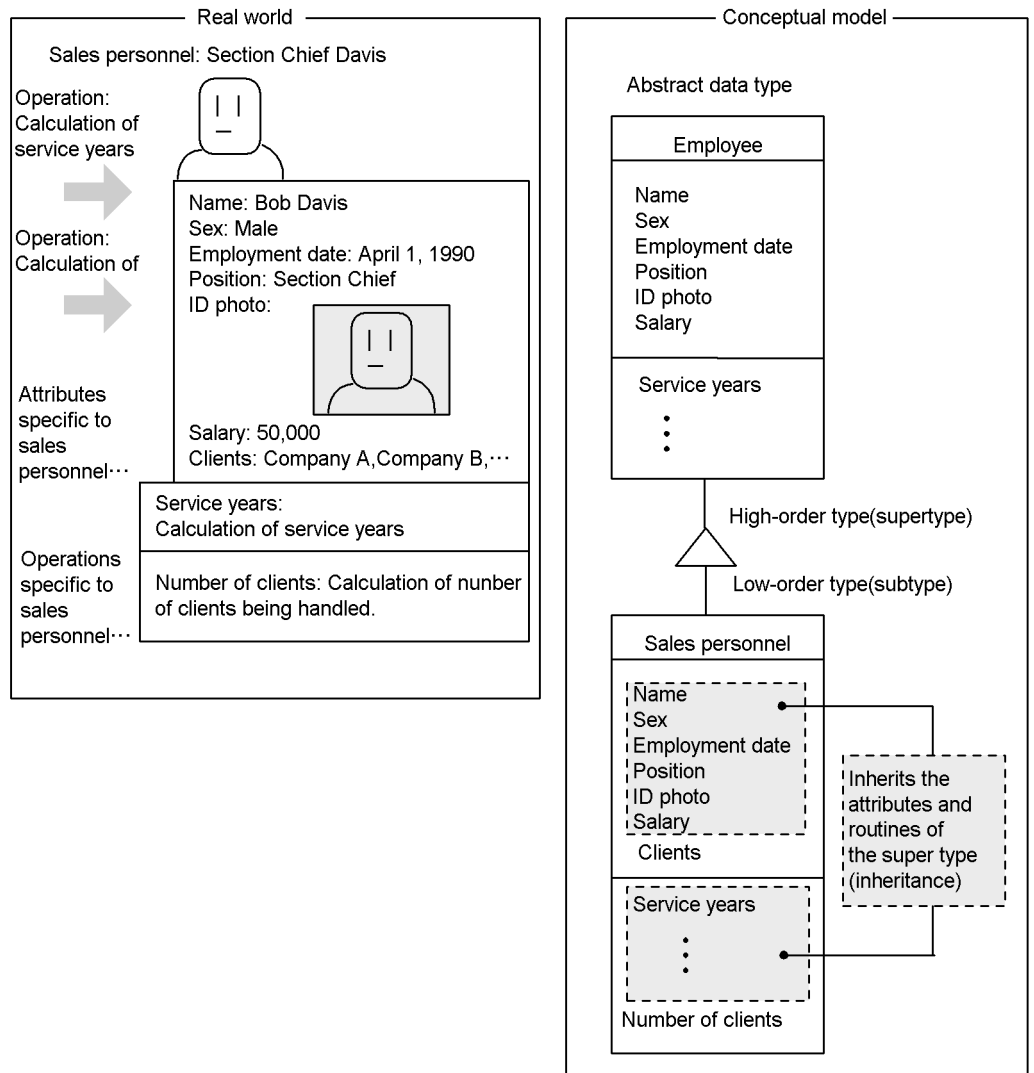
3.5.2 Subtypes and inheritance

(1) Subtype

Let's consider managing the information for employees who are in sales. *Salesperson* can be considered a more specific and specialized (particular) concept in comparison to the abstract concept of *employee*. The information on a salesperson may contain information related to sales activities in addition to the information common to all employees. Therefore, attributes such as *charge client* and *number of clients* are assigned to a salesperson, in addition to being an employee.

Figure 3-37 shows a conceptual model for a salesperson based on both real-world information and an abstract data type.

Figure 3-37: Conceptual model based on real-world information and an abstract data type (for a salesperson)



HiRDB makes it possible to take a particular data type and tailor it so as to define an abstract data type as a subtype.

For example, the subtype clause of a CREATE TYPE definition SQL statement can be used to define t_salesperson based on t_employee, as shown below:

```
CREATE TYPE t_salesperson UNDER t_employee(
    charge client VARCHAR(3000),
```

```

FUNCTION number of clients (.....)
RETURNS INTEGER
    .....
)

```

Note that a high-order abstract data type, such as `t_employee`, is called a super type.

(2) Substitutability

Salesperson is also an *employee*. HiRDB can handle an abstract data type value that was specialized (low-order) in a subtype also as a high-order abstract data type value. For example, the SQL shown below can substitute the `t_salesperson` values as `t_employee` column values in `STAFF_TABLE`.

Note

To handle `t_salesperson` values in the same manner as `t_employee` values, it is necessary to re-create SQL objects by executing `ALTER ROUTINE` before executing this SQL.

```

INSERT INTO STAFF_TABLE VALUES ( 51, t_salesperson(:name AS
CHAR(16),.....) )

```

The capability of a low-order abstract data type value to be considered as a high-order abstract data type value in this way is called substitutability.

(3) Inheritance

Because *salesperson* is also an *employee*, it also has attributes, such as *name* and *sex*, just like *employee*, and it should be possible to calculate *service years* for a salesperson.

Figure 3-37 shows that a lower-order abstract data type in a subtype inherits the attributes and routines defined for the higher-order abstract data type.

This manner in which a lower-order abstract data type inherits the attributes and routines of the higher-order abstract data type is called inheritance.

For example, the attribute `name` and the operation `service years` become available for `t_salesperson` values through inheritance. Consequently, as shown below, the SQL shown above can be executed without any changes for `STAFF_TABLE` into whose columns `t_salesperson` values have been inserted.

Note

Before this SQL can be executed, it is necessary to re-create SQL objects by executing `ALTER ROUTINE`.

```

SELECT employee number, employee..name, service years
(employee)
FROM STAFF_TABLE

```

```
WHERE service years(employee) >= 10
```

Subtype and inheritance can provide the following benefits:

- The concept of substitutability (e.g., a salesperson is also an employee) can be expressed quickly and clearly.
- Attributes and routine definitions can be shared, and new definitions can be added based on existing definitions. Consequently, the overhead for database and application development can be reduced, and a system with expandability can be constructed.

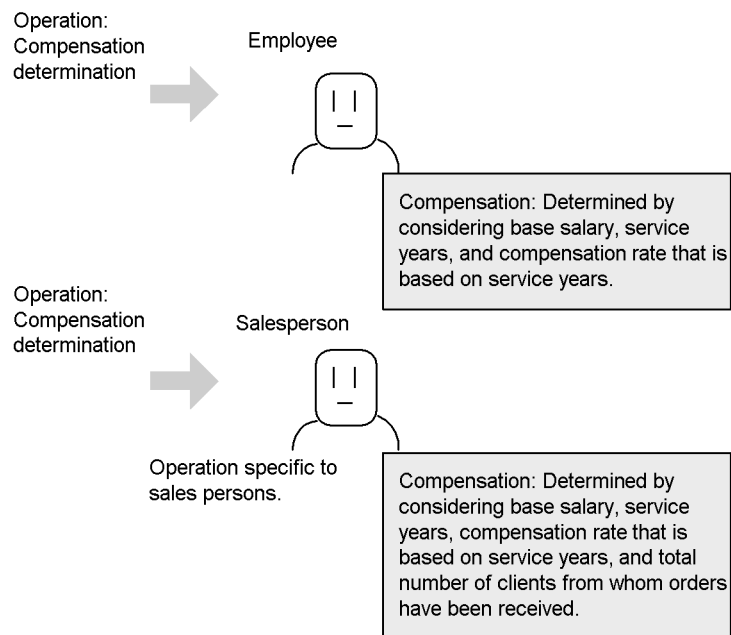
(4) Override

Let's consider an operation for determining compensation for an employee. For example, compensation can be determined based on the information that is set up by salary, service years, and compensation rate based on service years and on work performance.

Meanwhile, the operation *compensation* to be defined for *employee* is inherited by *salesperson*, as explained above. However, the method of determining compensation for salespersons may be different from that used for ordinary employees, and may include such attributes as *number of clients* that are specific to salespersons.

Figure 3-38 shows operations related to employees and salespersons in the real world.

Figure 3-38: Operations related to employees and salespersons in the real world



Here, let's define routines, such as `employee compensation` and `salesperson compensation`, that have different names for each of the abstract data types.

In this case, despite the fact that *employee* values and *salesperson* values can be handled together because of substitutability, it will be necessary to have a different name for the routine to be called for each value type. Consequently, applications will not be able to execute the data manipulation SQL described as follows:

```
SELECT  employee number, employee..name,  
        employee compensation(employee)  
FROM    STAFF_TABLE
```

... *Determination of compensation specific to salesperson cannot be executed.*

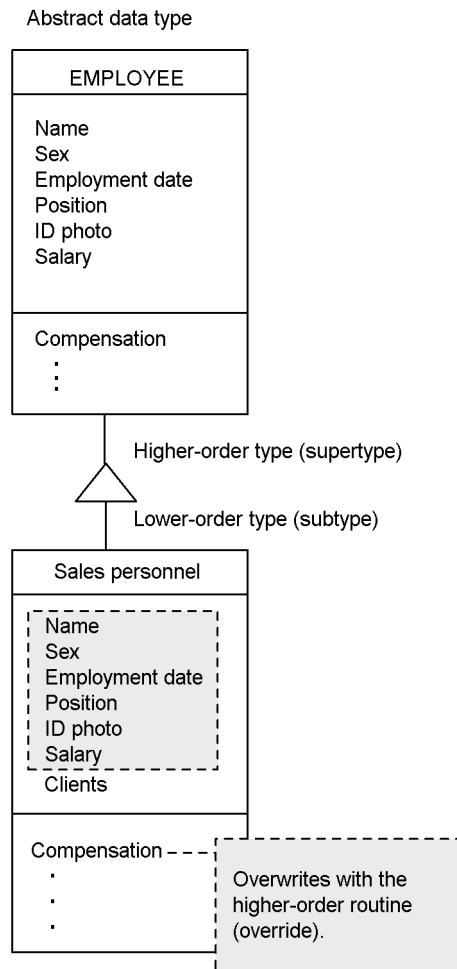
```
SELECT  employee number, employee..name,  
        salesperson compensation (employee)  
FROM    STAFF_TABLE
```

... *Determination of compensation specific to salesperson will be executed also for employees who are not salespersons.*

In HiRDB, a routine that has the same name as a routine defined in a higher-order abstract data type can be overwritten during the definition of a lower-order abstract data type. This process of overwriting is called *override*.

Figure 3-39 shows an example of *override*.

Figure 3-39: Override



HiRDB automatically executes an overridden routine based on the definition appropriate to the type of value that is used as the argument.

Override eliminates the need to change the name of the routine to be called, according to the value type. Therefore, it is possible to execute the data manipulation SQL shown as follows.

Note

Before this SQL can be executed, it is necessary to re-create SQL objects by executing `ALTER ROUTINE`.

```
SELECT  employee_number, employee..name,
```

```
        compensation (employee)
FROM      STAFF_TABLE
```

... *The overriding routine compensation executes a routine that is appropriate to each argument value.*

During execution of this SQL, the routine `compensation` defined for the `t_employee` type is executed for the `t_employee` type values, and the routine `compensation` defined for the `t_salesperson` type is executed for the `t_salesperson` type values.

An application can call a routine without being concerned with whether or not that routine has been overridden, as in the above SQL. Even when routines are added through `override`, it is possible to execute an SQL without changing the application.

3.5.3 Encapsulation

It may be desirable to prevent applications from directly accessing some of the information related to employees, such as personal information (e.g., information to be processed in the `compensation` operation and the value of the `employment_date` attribute). The reasons for this are listed as follows:

- Direct access from an application is not desirable in order to protect confidentiality.

Example

The value of the `salary` attribute may be referenced in internal processing of routines such as `compensation` but should not be directly accessible from the outside.

- Direct access from an application is meaningless.

Example

If job performance evaluation results are encoded and held as attribute values, accessing these values from an application will be of no use.

- It is necessary to prevent an application from directly altering internal information.

Example

It is necessary to prevent an application from changing the value of the `employment_date` attribute to a different date.

(1) *Encapsulation level*

In HiRDB, it is possible to specify an encapsulation level during the declaration of abstract data types and routines.

The following three encapsulation levels are available:

PRIVATE

Attribute values can be accessed and routines can be used only within the definition of an abstract data type.

For example, `PRIVATE` may be specified for the `t_employee` type attribute `employment date`.

When `PRIVATE` is specified for an attribute, it will not be possible to access this attribute or to use a routine from inside a subtype definition or an application.

PROTECTED

Attribute values can be accessed and routines can be used only within the definition of an abstract data type and within the definition of subtypes of that abstract data type.

For example, `PROTECTED` may be specified for the `t_employee` type routine `compensation`. This will allow `t_salesperson` (i.e., a subtype of `t_employee` type) to execute the `compensation rate` routine in order to calculate compensation rate, without directly referencing the attribute `employment date` or knowing the processing details of `compensation rate` (calculation of service years based on the employment date, and calculation of a compensation rate based on this value).

PUBLIC

No restrictions are placed, as with `PRIVATE` or `PROTECTED`, and attribute values can be accessed and routines can be used from within the definitions of other abstract data types and subtypes, as well as from applications.

For example, `PUBLIC` may be specified for the `service years` routine for which there is no access restriction.

Table 3-8 shows the encapsulation levels and their degrees of accessibility to abstract data type values and routines.

Table 3-8: Encapsulation levels and access types

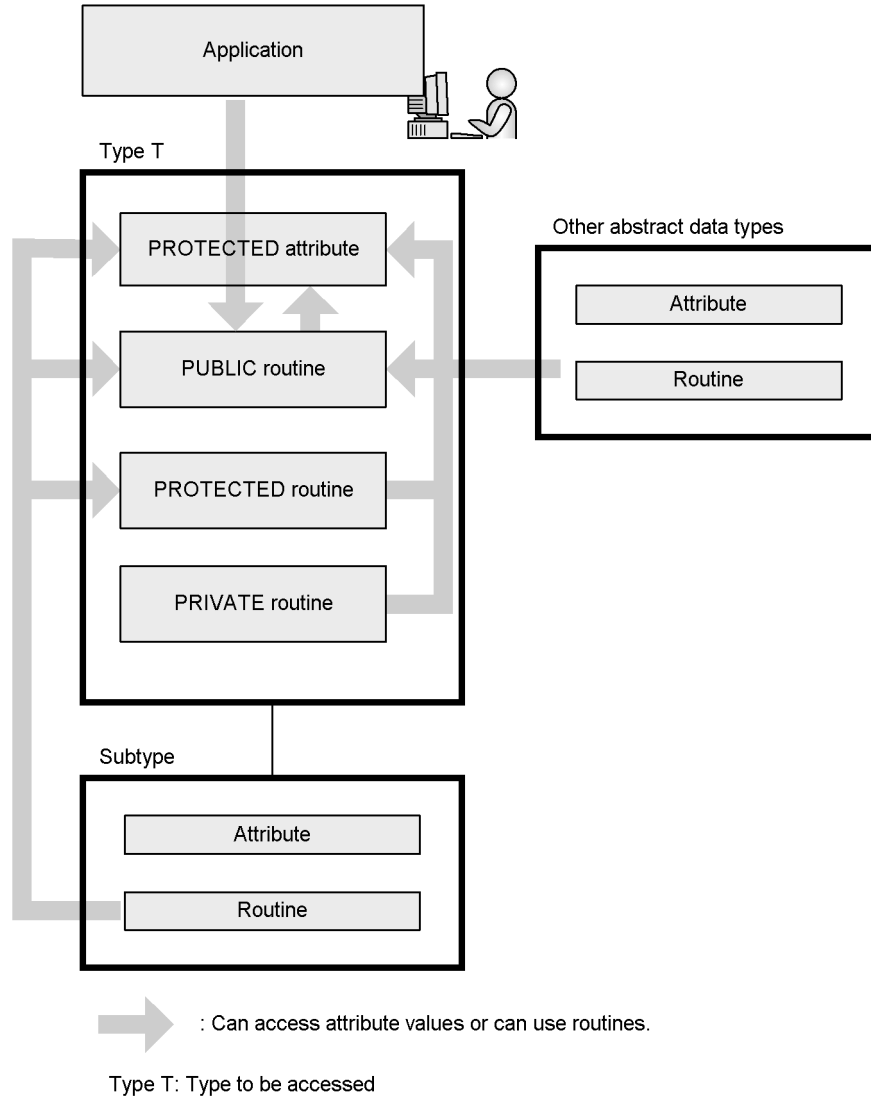
Encapsulation level	Access source			
	Within the definition of an abstract data type	Within the definition of a subtype abstract data type	Within the definitions of abstract data types other than those listed on the left	Applications
PUBLIC	Y	Y	Y	Y
PROTECTED	Y	Y	N	N
PRIVATE	Y	N	N	N

Y: Can access abstract data type values and can use routines.

N: Cannot access abstract data type values and cannot use routines (an SQL error occurs).

Figure 3-40 shows an example of the relationship between encapsulation levels and access types for abstract data type T.

Figure 3-40: Encapsulation levels and access types



For details about the procedures for designing and creating a table for which an

abstract data type is defined, see the *HiRDB Version 8 Installation and Design Guide*. For details about the procedures for manipulating a table for which an abstract data type is defined, see the *HiRDB Version 8 UAP Development Guide*.

Chapter

4. Database Physical Structure

This chapter explains the physical structure (segments and pages) of a database.

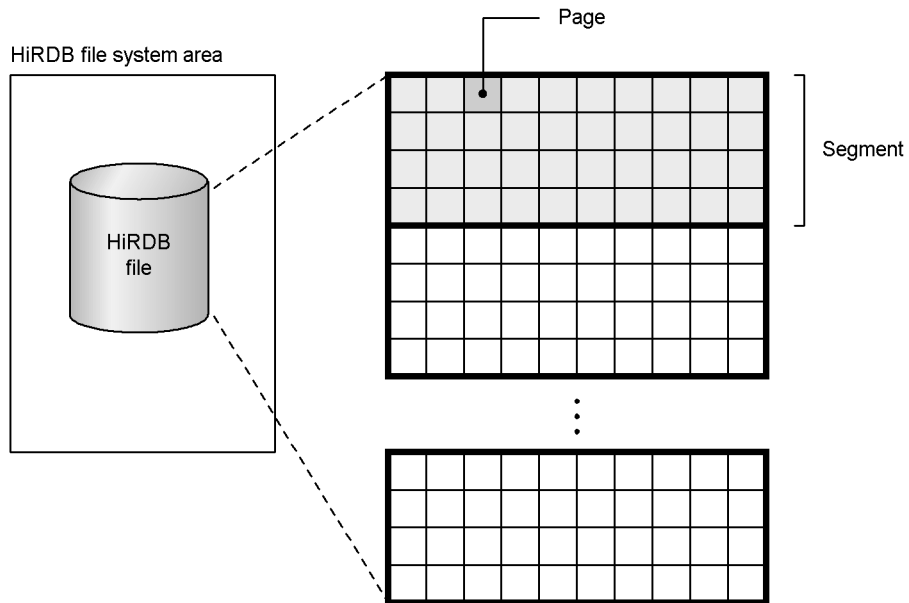
This chapter contains the following sections:

- 4.1 Database physical structure
- 4.2 Segment design
- 4.3 Page design

4.1 Database physical structure

Figure 4-1 shows the physical structure of a database.

Figure 4-1: Database physical structure



Explanation

- **HiRDB file system area**
This is an area in which HiRDB files are created.
- **HiRDB file**
This is a type of file unique to HiRDB that is used to store table and index data.
- **Segment**
This is the smallest unit of table and index data storage. One segment stores data from only one table or one index. A segment is composed of multiple, consecutive pages.
- **Page**
This is the smallest unit of database I/O operations. If the page size is large, one page can store several consecutive rows, thus reducing the number of I/

O operations in situations where the data is processed in the same order in which it is stored. Table 4-1 lists the available types of pages.

Table 4-1: Types of pages

Page type	Explanation
Data page	Stores row data for a table.
Index page	Stores index key values.
Directory page	Stores management information on the status of RDAREAs.

Specifying the physical structure of a database

The physical structure of a database is specified when its RDAREAs are defined. Specifically, the physical structure is specified in the following control statements:

- `create rdarea` statement of the database initialization utility (`pdinit`)
- `create rdarea` statement of the database structure modification utility (`pdmod`)

Following is an example of specifying the `create rdarea` statement:

```

create rdarea USRRD01 for user used by PUBLIC      1
  page 4096 characters                             2
  storage control segment 20 pages                3
  file name "C:\rdarea01\file01"                  4
  initial 150 segments ;                           5

```

Explanation

1. Specifies a name (`USRRD01`) for an RDAREA.
2. Specifies 4096 bytes as the page size.
3. Specifies 20 pages as the segment size.
4. Specifies the name of the HiRDB file system area where the RDAREA will be stored and a HiRDB file name:
`C:\rdarea01`: Name of HiRDB file system area
`file01`: HiRDB file name
5. Specifies the number of segments.

4.2 Segment design

(1) Segment statuses

Table 4-2 lists and describes the statuses that are assigned to segments.

Table 4-2: Segment statuses

Segment status	Description
Used segment*	A segment in which table or index data is stored. A used segment that is completely filled with data such that no more can be added is called a <i>full segment</i> , and a used segment from which data has been deleted so that only free pages remain (used free pages or unused pages) is called a <i>used free segment</i> .
Unused segment	A segment that is not being used. This segment can be used by any table or index in the RDAREA.
Free segment	A segment in which no data is stored. Both used free segments and unused segments are called free segments.

* A used segment can only be used by a table or an index that has stored data in that segment. Other tables or indexes cannot use the segment.

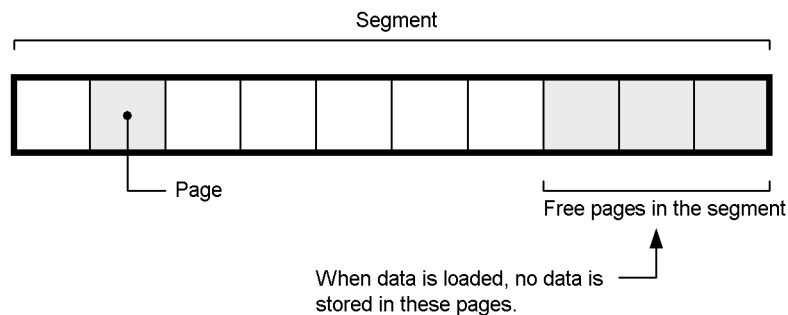
(2) Segment design policy

Segments must be designed with care, because the segment size (number of pages per segment) affects I/O times and disk space requirements. The recommended segment size for most purposes is 10-20 pages. For details about segment design, see the *HiRDB Version 8 Installation and Design Guide*.

(3) Ratio of free pages in a segment

When you define a table, you can specify the ratio of free pages in a segment. This concept is illustrated in Figure 4-2.

Figure 4-2: Ratio of free pages in a segment



Explanation

- In this example, the segment free page ratio is set to 30%. Therefore, three out of every 10 pages are free pages.
- The segment free page ratio is specified with the `PCTFREE` option of the `CREATE TABLE`. This ratio can be specified in the 0-50% range; the default is 10%.
- When data is loaded (including when reloading and reorganizing), it is not stored in the free pages specified here.

The data storage efficiency improves as the segment free page ratio becomes smaller.

Increasing the segment free page ratio sometimes improves performance. For example, when data is added to a table for which a cluster key is defined and a segment free page ratio has been set, you can cause data to be stored in positions that are near a cluster key value, which minimizes the number of data I/O operations that will be needed.

For details about determining the segment free page ratio, see the *HiRDB Version 8 Installation and Design Guide*.

(4) Allocating and releasing segments

Segments are not allocated when the table is defined. Instead, segments are allocated as needed when data is saved to the table. Once a segment is allocated (once it becomes a used segment), no other table or index can use that segment until it is released. Consequently, the RDAREA may run out of free space if data is added and deleted repeatedly, even though the amount of data has not increased. To prevent this from occurring, periodically perform the following operations to release segments:

- Reorganize tables or indexes using the database reorganization utility (`pdrorg` command).
- Release used free segments using the free page release utility (`pdreclaim` command).

For details about reorganizing tables and indexes, and about releasing used free segments, see the *HiRDB Version 8 System Operation Guide*. In addition to these operations, the following operations also release segments:

- Executing the `PURGE TABLE` statement
- Reinitializing the RDAREA
- Deleting the table definition
- Deleting the index definition
- Loading data in the creation mode (`-d` option specification)

(5) Reusing free space

The free space reuse facility enables you to efficiently utilize free space made available by deleting data.

(a) How searching is performed when data is saved

When data is saved to a table, there are two page search modes that are used to search for a storage area:

- New page allocate mode

In this mode, when the last page in a used segment becomes full, a new unused segment is allocated. If the RDAREA runs out of unused pages, a search is then conducted to find free space in a used page, starting at the beginning of the first used segment. Once free space is found, the data is saved to that free space.

- Free page reuse mode

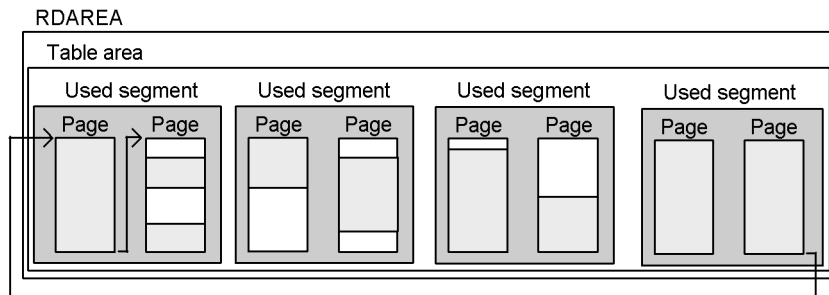
In this mode, when the last page in a used segment becomes full, a search is conducted to find free space in used pages starting in the first used segment. If no free space is found, an unused segment is allocated. The next search start position is also remembered, so that, the next time, searching for free space begins at this position.

(b) Overview of the free space reuse facility

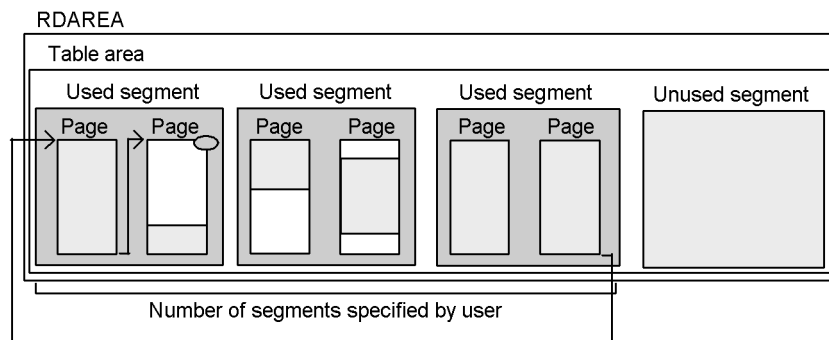
The *free space reuse facility* is designed to enable the free space in used pages to be utilized by activating the free page reuse mode. This occurs when the number of used segments in a table reaches the number of segments specified by the user, and the last segment becomes full. Once the free space in every segment of the specified number of segments runs out, the free space reuse facility activates the new page allocate mode, and a new unused segment is allocated. If the number of segments is not specified, free space is not reused until there are no more unused pages in the RDAREA. If the free space reuse facility is not used, the system searches for free space from the beginning of the used segment every time a search is performed. If the free space reuse facility is used, the free page reuse mode is activated, the next search start position is remembered, and the next search is performed from there. As such, use of this facility enables searches to be performed more efficiently. Figure 4-3 provides an overview of the free space reuse facility.

Figure 4-3: Overview of the free space reuse facility

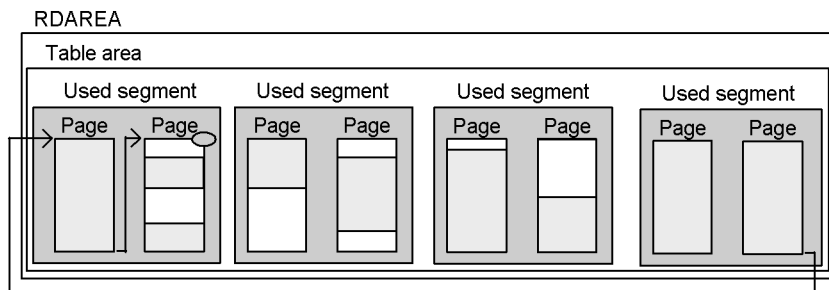
- If the free space reuse facility is not used



- If the free space reuse facility is used (with the number of segments specified)



- If the free space reuse facility is used (without the number of segments specified)



- : Free space on used page
- : Data storage area
- : Search when data is inserted
- : Start position stored for next search

Explanation

- If the free space reuse facility is not used
Every time data is to be inserted after the RDAREA has run out of unused pages, the system searches for free space in a used page, starting at the beginning of the first used segment. Once the space is found, the data is saved to that free space.
- If the free space utility is used (with the number of segments specified)
When an attempt is made to insert data into a table after the specified number of segments is reached, this facility searches for free space in a used page, starting at the beginning of the first used segment. Once the space is found, it saves the data to that free space. In addition, the facility remembers that location as the next search start position and, the next time, begins searching for free space from this position.
- If the free space utility is used (without the number of segments specified)
When an attempt is made to insert data after the RDAREA has run out of unused pages, this facility searches for free space in a used page, starting at the beginning of the first used segment. Once the space is found, it saves the data to that free space. In addition, the facility remembers that location as the next search start position and, the next time, begins searching for free space from this position.

For details about the free space reuse utility, see the *HiRDB Version 8 System Operation Guide*.

(c) Application criteria

For operations requiring frequent reorganization, use the free space reuse facility to absolutely minimize the number of reorganizations you need to perform, especially when repeated deletions and insertions of data are using up large numbers of segments.

(d) Environment settings

The following environment settings are used for the free space reuse facility. For details, see the *HiRDB Version 8 Installation and Design Guide*.

1. In the `pd_assurance_table_no` operand, specify the number of tables that can use the free space reuse facility.
2. Specify the number of segments in the `SEGMENT REUSE` option of the definition `SQL CREATE TABLE` statement. For tables that have already been created, specify this value in the `SEGMENT REUSE` option of the `ALTER TABLE` statement.

(e) Notes

The free space reuse facility does not operate in the following cases:

- When data is being stored with the hash facility for hash row partitioning
- When a data dictionary table is being stored
- When data is being stored using the data load or database reorganization utility (pdorg)

4.3 Page design

This section describes page statuses and explains page design policy.

(1) Page statuses

Table 4-3 lists and describes the statuses that are assigned to pages.

Table 4-3: Page statuses

Page status	Description
Used page	A page in which table or index data is stored. A used page that is completely filled with data such that no more can be added is called a <i>full page</i> , and a used page from which data has been deleted so that it no longer contains data is called a <i>used free page</i> .
Unused page	A page that is not being used.
Free page	A page in which no data is stored. Both used free pages and unused pages are called free pages.

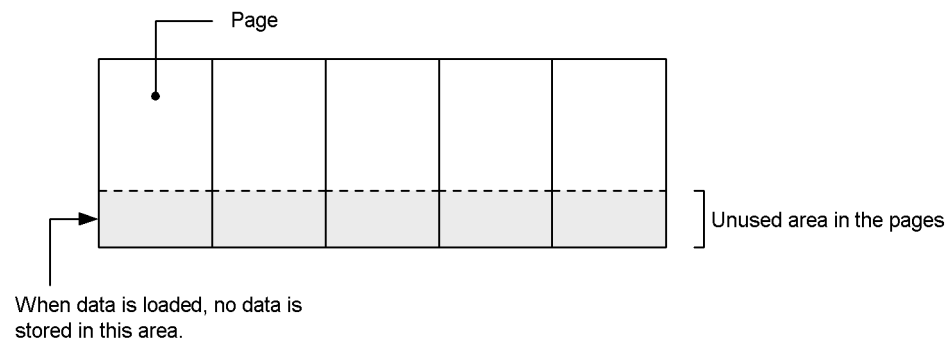
(2) Page design policy

Pages must be designed with care, because the page size affects data I/O times. For details about page design, see the *HiRDB Version 8 Installation and Design Guide*.

(3) Ratio of unused area in a page

When you define a table or index, you can specify the ratio of unused area in a page. This concept is illustrated in Figure 4-4.

Figure 4-4: Ratio of unused area in a page



Explanation

- In this example, the page unused area ratio is set to 30%.

- The page unused area ratio is specified with the `PCTFREE` option of the `CREATE TABLE` or `CREATE INDEX`. This ratio can be specified in the 0-99% range; the default is 30%.
- When data is loaded into the table, no data is stored in the unused area.

The data storage efficiency improves as the page unused area ratio becomes smaller. Increasing the page unused area ratio sometimes improves performance. For example, when data is updated, you can reduce the number of data I/O operations, provided that either of the following occurs:

- The row size is increased as a result of the updating.
- A row is added to a table for which a cluster key is specified.

For details about determining the page unused area ratio, see the *HiRDB Version 8 Installation and Design Guide*.

(4) Allocating and releasing pages

(a) Allocating pages

Pages are not allocated when the table is defined. Instead, pages are allocated as needed when data is saved to the table. Once a page is allocated (once it becomes a used page), it cannot be reused until it is released.

When an index is defined, the system allocates pages according to the number of data items to be included in the index. If there are no data items, only one page (*root page*) is allocated. However, if the `EMPTY` option is specified in the `CREATE INDEX` (index entries not to be created), no pages will be allocated.

Reference note:

- If data in a non-FIX table is updated and as a result there is a change in the row size, any area freed up by a reduction in the row size cannot be reused.
- 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.

Note:

Reusing a page freed up by deletion of data is subject to the following restrictions:

- The page cannot be used by VARCHARs of 256 bytes or greater, BINARY types, abstract data types, and branch rows of repetition columns.
- The page cannot be used when data is inserted until the segment utilization rate reaches 100%.
- A transaction that executes DELETE cannot use the space freed by the DELETE until COMMIT is issued.

(b) Releasing pages

- When a segment is released, the pages in that segment are released.
- When a UAP deletes all rows in a page in a table that is locked with the LOCK statement in the EXCLUSIVE specification, that page is released.
- Because the used segments in a table or index are released when PURGE TABLE is executed, all pages in that segment other than the index root page are also released.
- You can use the free page release utility (pdreclaim command) to release used pages. For details about releasing used free pages, see the *HiRDB Version 8 System Operation Guide*.

Chapter

5. Database Access Using SQL

This chapter explains the use of the SQL database manipulation language to manipulate a database.

This chapter contains the following sections:

- 5.1 Use of SQL in HiRDB
- 5.2 Basic data manipulation
- 5.3 Stored procedures and stored functions
- 5.4 Java stored procedures and Java stored functions
- 5.5 Triggers
- 5.6 Integrity constraints
- 5.7 Referential constraints
- 5.8 Check constraints
- 5.9 Check pending status
- 5.10 Improving database access performance
- 5.11 Narrowed search
- 5.12 Accessing databases using DB access products

5.1 Use of SQL in HiRDB

This section provides an overview of using SQL for database access. For details about the SQL syntax, see the *HiRDB Version 8 SQL Reference*; For details about creating and designing a UAP, see the *HiRDB Version 8 UAP Development Guide*.

5.1.1 HiRDB SQL functions

Database manipulation language SQL statements are used to manipulate data in tables. The following functions are provided in the HiRDB SQL:

Basic data manipulation

- Data retrieval
- Data updating
- Data deletion
- Data insertion
- Search for specific data
- Data operations
- Data processing
- Manipulation of data in a table containing an abstract data type

Reduction in number of steps required for UAP development and reduction of communication and analysis overhead

- Stored procedures
- Stored functions

Improvement in database access performance

- Block transfer facility
- Rapid grouping facility
- FETCH facility using arrays
- Holdable cursor
- SQL optimization option

5.1.2 SQL execution methods

There are three methods of using SQL to access a HiRDB database:

- By coding SQL statements in a UAP and executing the executable-format file of the UAP (a UAP in which SQL statements are directly coded is called an

embedded SQL UAP)

- By using the HiRDB SQL Executer to execute SQL statements interactively
- By using the *database definition utility* (`pddef`) to execute SQL statements

You can create a UAP to reference table data when complex operations or computational processing are required, or you can execute SQL statements interactively in order to perform simple data operations.

The following lists the high-level languages that can be used to code embedded UAPs:

- C
- C++
- COBOL
- OOCOBOL

5.2 Basic data manipulation

5.2.1 Cursor

Table retrieval results usually span multiple rows. A cursor is used by a UAP to indicate the position from which the newest result is to be extracted one row at a time out of retrieval results that span multiple rows. A cursor can be used for retrieving, updating, and deleting data.

`DECLARE CURSOR` is specified in order to use a cursor. The `OPEN` statement is used to open a cursor, and the `CLOSE` statement is used to close a cursor. The `FETCH` statement advances the position of the cursor.

5.2.2 Data retrieval

Data retrieval is the process of selecting the rows from a table that satisfy conditions that are specified with respect to a column. The data retrieval methods and an example of specifying SQL statements are shown as follows.

Data retrieval methods

The `SELECT` statement is used to retrieve data. The following three methods are available for data retrieval:

- Retrieval using a cursor
- Retrieval of multiple tables (specify the `FROM` clause in the `SELECT` statement)
- Retrieval by row (specify `ROW` in the selection expression of the `SELECT` statement)

Data retrieval SQL specification example

An example of retrieval using a cursor is explained as follows.

Example

1. Define cursor.

In this example, a cursor named `CUR1` is used to retrieve from the stock table (`STOCK`) the product names (`PNAME`), colors (`COLOR`), and prices (`PRICE`) of products whose product name (`PNAME`) is `skirt`:

```
DECLARE CUR1 CURSOR FOR
SELECT PNAME,COLOR,PRICE FROM STOCK
WHERE PNAME='skirt'
```

2. Open cursor.

Cursor CUR1 is opened:

```
OPEN CUR1
```

3. Extract data.

While cursor CUR1 is open, it is advanced by one row and the contents of that row are stored in the UAP in specified areas (:XPNAME, :XCOLOR, :XPRICE):

```
FETCH CUR1 INTO
  :XPNAME,
  :XCOLOR,
  :XPRICE
```

4. Close cursor.

Cursor CUR1 is closed:

```
CLOSE CUR1
```

By specifying `LIMIT` following the `ORDER BY` clause, you can retrieve search results from the first n rows. Specifying `LIMIT` may also improve SQL search performance. For details about retrieving search results from the first n rows, see the *HiRDB Version 8 UAP Development Guide*.

5.2.3 Data updating

Data updating is the process of changing information in a table. The data updating methods and an example of specifying SQL statements are shown as follows.

Data updating methods

The `UPDATE` statement is used to update data. The following three methods are available for data updating:

- Updating the row being pointed to by the cursor
- Updating only those rows that satisfy a condition (specify the `WHERE` clause in the `UPDATE` statement)
- Updating by row (specify `ROW` in the `SET` clause)

Data updating SQL specification example

An example of updating only those rows that satisfy a condition is explained as follows.

Example

In this example, the `UPDATE` statement updates to 20 the stock quantity

(SQUANTITY) of products whose product code (PCODE) in the stock table (STOCK) is 411M:

```
UPDATE STOCK
  SET SQUANTITY=20
  WHERE PCODE='411M'
```

5.2.4 Data deletion

Data deletion is the process of removing either the rows from a table that satisfy conditions that are specified with respect to a column or all rows that constitute the table. The data deletion methods and an example of specifying SQL statements are shown as follows.

Data deletion methods

The DELETE or PURGE TABLE statement is used to delete data. The following three methods are available for deleting data:

- Deleting the row being pointed to by the cursor
- Deleting only those rows that satisfy a condition (specify the WHERE clause in the DELETE statement)
- Deleting all rows (PURGE TABLE statement)

Data deletion SQL specification example

An example of deleting only those rows that satisfy a condition is explained as follows.

Example

In this example, the DELETE statement deletes from the stock table (STOCK) only the data whose product name (PNAME) is skirt:

```
DELETE FROM STOCK
  WHERE PNAME='skirt'
```

5.2.5 Data insertion

Data insertion is the process of inserting rows into a table. The data insertion methods and an example of specifying SQL statements are shown as follows.

Data insertion methods

The INSERT statement is used to insert rows. The following two methods are available for inserting rows into a table:

- Inserting rows by column
- Inserting rows by row (specify ROW in the INSERT statement)

Data insertion SQL specification example

An example of inserting rows by column is explained as follows.

Example

In this example, the `INSERT` statement inserts in each column of the stock table (`STOCK`) the values set in embedded variables (`:ZPCODE`, `:ZPNAME`, `:ZCOLOR`, `:ZPRICE`, and `:ZSQUANTITY`) that are used for transferring values between a table and UAPs:

```
INSERT INTO STOCK (PCODE, PNAME, COLOR, PRICE, SQUANTITY)
VALUES (:ZPCODE, :ZPNAME, :ZCOLOR, :ZPRICE, :ZSQUANTITY)
```

5.2.6 Searching for specific data

A search condition is specified to manipulate specific data in a table. A search condition is a condition for selecting rows. For example, a search condition can specify a specific range of data or all data that is not the null value. It is also possible to use Boolean operators to combine multiple conditions. The methods of searching for specific data and SQL specification examples follow.

(1) *Methods of searching for specific data*

The following methods can be used to search for data in a table:

- Searching for data within a specific range
- Searching for a specific character string
- Searching for data that is not the null value
- Searching for data that satisfies multiple conditions
- Search using a subquery

(2) *SQL examples for searching for specific data*

(a) **SQL specification example for searching for data within a specific range**

The following three methods are available for searching for data within a specific range:

- Comparison predicate (used for equivalence and size comparison)
- `BETWEEN` predicate (used for extracting data within a specific range)
- `IN` predicate (used for extracting only the data that matches multiple values that are specified)

An example of using a comparison predicate for a data search is explained as follows.

Example

In this example, the `SELECT` statement searches the stock table (`STOCK`) for the product codes (`PCODE`) and product names (`PNAME`) of products whose stock quantity (`SQUANTITY`) is 50 or less:

```
SELECT PCODE, PNAME FROM STOCK
WHERE SQUANTITY<=50
```

(b) SQL specification example for searching for a specific character string

An example of conducting a search for rows in which there is a column that contains a specific character string is explained as follows.

Example

The `LIKE` predicate is used in this example. The `SELECT` statement searches the stock table (`STOCK`) for the product names (`PNAME`) and stock quantities (`SQUANTITY`) of products whose product code (`PCODE`) has `L` as its second character:

```
SELECT PNAME, SQUANTITY FROM STOCK
WHERE PCODE LIKE '_L%'
```

(c) SQL specification example for searching for data that is not the null value

An example of conducting a search for rows in which a specified column does not contain the null value is explained as follows.

Example

`NOT` of the `NULL` predicate is used in combination in this example. The `SELECT` statement searches the stock table (`STOCK`) for the product codes (`PCODE`) of products whose product name (`PNAME`) is not empty (not the null value):

```
SELECT PCODE FROM STOCK
WHERE PNAME IS NOT NULL
```

(d) SQL specification example for searching for data that satisfies multiple conditions

An example of conducting a search for rows that contain data that satisfies a combination of multiple conditions is explained as follows.

Example

Boolean operators (`AND`, `OR`, and `NOT`) are used in this example. The `SELECT` statement searches the stock table (`STOCK`) for the product codes (`PCODE`) of products whose product name (`PNAME`) is `blouse` or `polo shirt` and whose stock quantity (`SQUANTITY`) is 50 or greater:

```
SELECT PCODE, SQUANTITY FROM STOCK
```

```
WHERE (PNAME='blouse'
      OR PNAME='polo shirt')
      AND SQUANTITY=>50
```

(e) SQL specification example for a search using a subquery

You can code a complex query by specifying a retrieval result of a search as a condition in the `SELECT` statement. This is called a *subquery*. The following two subquery methods are available:

- Quantified predicate

A quantified predicate is used to narrow the results of a subquery by determining whether the results of the subquery satisfy a specified comparison condition.

- EXISTS predicate

The EXISTS predicate is used to determine whether the results of a subquery constitute an empty set.

An example of conducting a search that uses a subquery that contains a quantified predicate is explained as follows:

Example

In this example that uses a quantified predicate, the `SELECT` statement searches the stock table (`STOCK`) for the product codes (`PCODE`) and product names (`PNAME`) of products that have a greater stock quantity value than the stock quantity (`SQUANTITY`) value for `blouse`:

```
SELECT PCODE, PNAME FROM STOCK
      WHERE SQUANTITY>ALL
            (SELECT SQUANTITY FROM STOCK
             WHERE PNAME='blouse')
```

5.2.7 Data operations

It is possible to retrieve numerical values or dates from a table's columns, to perform operations on such values, and to extract the results. The data operation methods and an SQL specification example follow.

Data operation methods

The following methods can be used to perform data operations:

- Concatenation operations on character string data
- Arithmetic operations on numeric data
- Operations on date and time data
- Operations using scalar functions

- Specification of conditional values using the `CASE` expression
- Explicit type conversion using a `CAST` specification

Data operation SQL specification example

An example of performing an arithmetic operation on numeric data is explained as follows.

Example

In this example, the `SELECT` statement uses the price (`PRICE`) and stock quantity (`SQUANTITY`) to compute the projected revenue for products with `sweat pants` as the product name (`PNAME`) and extracts the product codes (`PCODE`) and the computation results (in units of \$10):

```
SELECT PCODE, PRICE*SQUANTITY/10 FROM STOCK
WHERE PNAME='sweat pants'
```

5.2.8 Data processing

When data is extracted from a table, it is possible to process that data by grouping it or by sorting it in ascending or descending order. The data processing methods and an SQL specification example follow.

Data processing methods

The following operations can be used to process data in a table:

- Data grouping (`GROUP BY` clause specification and the set function)
- Sorting data in ascending or descending order (`ORDER BY` clause specification)
- Excluding duplicate data (`DISTINCT` specification)
- Set operations between sets of rows (derived tables) (`UNION` or `EXCEPT` specification)

Data processing SQL specification example

An example of rearranging (sorting) data into ascending order is explained as follows.

Example

In this example, the `SELECT` statement retrieves from the stock table (`STOCK`) the product codes (`PCODE`) and stock quantities (`SQUANTITY`) of products and sorts so that the retrieved data is in ascending order of the product codes (`PCODE`):

```
SELECT PCODE, SQUANTITY FROM STOCK
ORDER BY PCODE
```

5.2.9 Manipulation of data in a table containing an abstract data type

This section describes how to manipulate data in a table containing abstract data types.

(1) Abstract data type provided by a plug-in

When a plug-in is used, creation of a UAP by specifying the facility provided by the plug-in makes it possible to quickly and easily manipulate multimedia data such as documents and images.

The following examples use the HiRDB Text Search Plug-in.

(a) Data retrieval

An example of using a Boolean predicate for data retrieval is explained as follows.

Example

In this example, the `SELECT` statement retrieves medicine IDs that contain the keyword `headache` in the `efficacy` section of the `operation manual` column of the `MEDICINE_MANAGEMENT_TABLE` (the SQL uses the `contains` in order to extract those documents that match the text search condition provided by the plug-in):

```
SELECT medicine-ID FROM MEDICINE_MANAGEMENT_TABLE
WHERE
contains(operation-manual, 'attached-document-data[efficacy{
"headache"}]') IS TRUE
```

(b) Data updating

An example of data updating is explained as follows.

Example

In this example, the `UPDATE` statement updates the data in the `operation manual` for those columns in the `MEDICINE_MANAGEMENT_TABLE` that have `medicine 2` as the `medicine ID` (the SQL uses the `SGMLTEXT` facility provided by the plug-in):

```
UPDATE MEDICINE_MANAGEMENT_TABLE SET operation-manual =
SGMLTEXT(:sgml)
WHERE medicine-ID = 'medicine-2'
```

The following `sgml` BLOB type embedded variable must be defined before the `UPDATE` statement:

```
EXEC SQL BEGIN DECLARE SECTION;                               /
    SQL TYPE IS BLOB(300K) sgml;                               /
EXEC SQL END DECLARE SECTION;                                   /
```

```
strcpy (sgml.sgml_data, char_ptr_pointing_to_a_sgml_text);    2
sgml.sgml_length =
strlen (char_ptr_pointing_to_a_sgml_text);                  3
```

Explanation

1. Defines the `sgml` BLOB type embedded variable.
2. Stores the new data for updating in the `sgml` embedded variable.
3. Sets the attribute value `sgml_length` of the created BLOB data to the length of the stored data.

(c) Data deletion

An example of data deletion is explained as follows:

Example

In this example, the `DELETE` statement deletes from the `MEDICINE_MANAGEMENT_TABLE` those rows that have `medicine 2` in the `medicine ID` column:

```
DELETE FROM MEDICINE_MANAGEMENT_TABLE
WHERE medicine-ID = 'medicine-2'
```

(d) Data insertion

An example of data insertion is explained as follows.

Example

In this example, the `INSERT` statement inserts into the `MEDICINE_MANAGEMENT_TABLE` rows that have `medicine 25` in the `medicine ID` column (the `SGMLTEXT` facility provided by a plug-in is used in the SQL):

```
INSERT INTO MEDICINE_MANAGEMENT_TABLE
(medicine-ID, operation-manual)
VALUES ('medicine-25', SGMLTEXT (:sgml))
```

The following `sgml` BLOB type embedded variable must be defined before the `UPDATE` statement:

```
EXEC SQL BEGIN DECLARE SECTION;                               1
SQL TYPE IS BLOB(300K) sgml;                                   1
EXEC SQL END DECLARE SECTION;                                  1
strcpy (sgml.sgml_data, char_ptr_pointing_to_a_sgml_text);    2
sgml.sgml_length =
strlen (char_ptr_pointing_to_a_sgml_text);                    3
```

Explanation

1. Defines the `sgml` BLOB type embedded variable.
2. Stores the new data for updating in the `sgml` embedded variable.
3. Sets the attribute value `sgml_length` of the created BLOB data to the length of the stored data.

(2) User-defined abstract data type

A routine or a component specification is used to manipulate data of a table containing a user-defined abstract data type. A component specification is used to manipulate the attribute of a column that comprises an abstract data type. The following examples manipulate the data of a table containing a user-defined abstract data type.

(a) Retrieving an abstract data type column

An example of retrieving a column of a table containing a user-defined abstract data type is explained as follows.

Example

In this example, the `SELECT` statement uses the user-defined facility `service-years` to retrieve from `STAFF_TABLE` the employee numbers of employees whose service years are 20 years or more:

```
SELECT employee-no
FROM STAFF_TABLE
WHERE service-years (employee) >= 20
```

(b) Updating an abstract data type column

An example of updating a column of a table containing a user-defined abstract data type is explained as follows.

Example

In this example, the `UPDATE` statement updates to `CHIEF` the `position` employee attribute of the employee whose `EMPLOYEE_NO` column in `STAFF_TABLE` is 9001230; this `UPDATE` statement uses the `employee..position` component specification for this updating:

```
UPDATE STAFF_TABLE
SET employee.position = 'CHIEF'
WHERE employee_no = '900123'
```

(c) Deleting an abstract data type column

An example of deleting a column of a table containing a user-defined abstract data type is explained as follows:

Example

In this example, the DELETE statement uses the `employee..position` component specification to delete the data in which the `position` attribute of the `employee` column in `STAFF_TABLE` is `GENERAL`:

```
DELETE FROM STAFF_TABLE
WHERE employee..position='general'
```

(d) Data insertion

An example of insertion of data into a table containing a user-defined abstract data type is explained as follows.

Example

In this example, the INSERT statement uses the `t_employee` constructor facility to insert a row whose `EMPLOYEE_NO` column is `990070` into `STAFF_TABLE` (`:xidphoto` is a BLOB type embedded variable in which the `ID_photo` image has been set):

```
INSERT INTO STAFF_TABLE
VALUES ('990070', t_employee('Mary Moore',
                             'F',
                             'GENERAL',
                             '1999-04-01',
                             :xidphoto AS BLOB,
                             140000
                             ))
```

5.3 Stored procedures and stored functions

When a set of operations on a database is defined as a *procedure*, it is called a *stored procedure*; when a set of operations on a database is defined as a *function*, it is called a *stored function*.

Defining a stored procedure or a stored function generates an SQL object that codes an access procedure. The resulting stored procedure or stored function, together with its definition information, is stored in the database. Processing procedures for stored procedures or stored functions can be coded in either SQL or Java. A procedure that is coded in SQL is called an *SQL stored procedure* or an *SQL stored function*; a procedure that is coded in Java is called a *Java stored procedure* or a *Java stored function*. For details about Java stored procedures and Java stored functions, see Section 5.4 *Java stored procedures and Java stored functions*.

A stored procedure may or may not have input, output, or input/output parameters; it is called by an SQL `CALL` statement. A stored function may or may not have input parameters; it is able to return a return value, and thus can be called as a value expression in an SQL. It should be noted, however, that a stored function can be used only for processing data; it cannot be used for accessing tables in the database.

(1) Application of a stored procedure to a job

Explained as follows are the types of jobs for which a stored procedure may be useful. For example, a product management job might involve the processing described as follows for analyzing the sales status of a product.

- For each product, the total number of units ordered each month is computed and the result is incorporated into a table that shows the cumulative number of units ordered since the product was introduced.

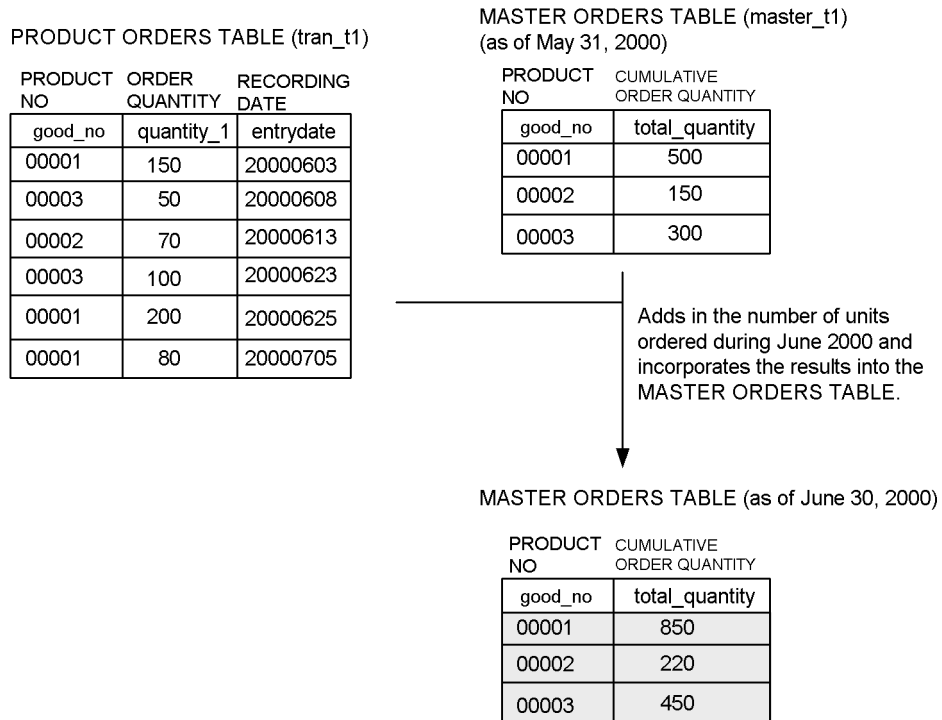
This is accomplished through the multiple database access processes described as follows.

1. Use a cursor to retrieve from the product orders table the product numbers, order quantities, and order recording dates for products for which orders were received during the month (`SELECT` statement).
2. Use a cursor to retrieve from the master order table the cumulative total number of units ordered for each product whose product number is contained in both the master orders table and the product orders table (`SELECT` statement).
3. For each applicable product, compute the sum of the total number of units ordered during the month and the cumulative total number of units ordered from the time the product was introduced through the end of the previous month, and update the master orders table with the result (`INSERT` and `UPDATE` statements).

When a series of database accessing processes such as the job shown here is registered

at the database side rather than at the UAP side, the processes can be called for use from multiple UAPs. Figure 5-1 shows jobs to which a stored procedure can be applied.

Figure 5-1: Jobs to which a stored procedure can be applied



In the database access process shown in Figure 5-1, the total number of units ordered during June 2000 is determined for each product from the **PRODUCT ORDERS TABLE**, which has **PRODUCT NO**, **ORDER QUANTITY**, and **RECORDING DATE** as its columns. The results are incorporated into the **MASTER ORDERS TABLE**, which has **PRODUCT NO** and **CUMULATIVE ORDER QUANTITY** as its columns. In this case, the database access process described as follows can be defined as a stored procedure and registered in the database.

Processing contents of stored procedure

The number of units ordered for a specified period is determined, and the result is incorporated into the master orders table.

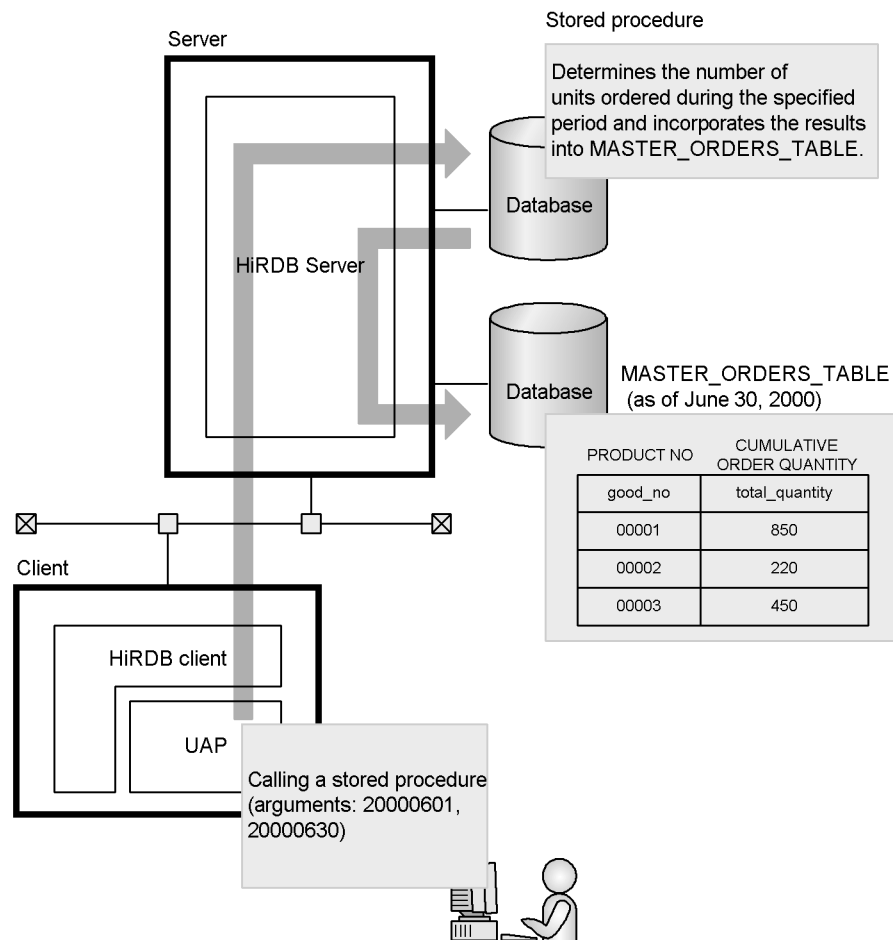
A UAP can perform the product management job shown in Figure 5-1 by simply calling the stored procedure with the argument described as follows specified.

Argument specified by UAP

Period for which number of units ordered is to be computed: (begin-date 20000601, end-date 20000630)

Figure 5-2 shows how a stored procedure is used.

Figure 5-2: Stored procedure usage



This illustrates how, when a stored procedure is used, it is possible to register a database access process at the database side, thus turning the database access process into a component.

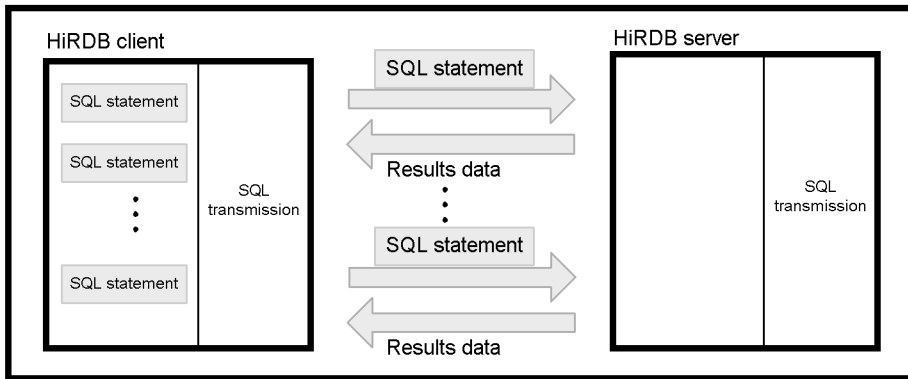
Even when the database access process is changed, all that has to be done is to change the stored procedure; there is no need to change the UAP, thus reducing the number of steps required for UAP development.

In order to execute multiple SQL statements, an application typically has to access the

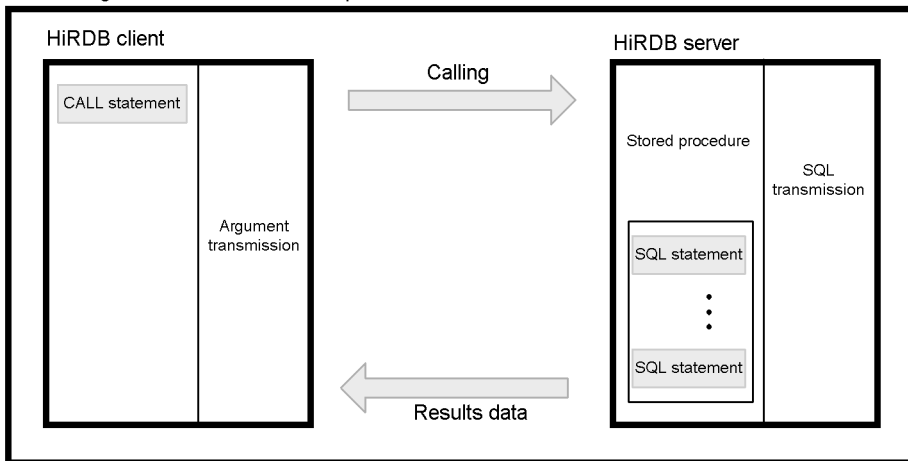
database as many times as there are SQL statements to be executed. By contrast, storing multiple SQL statements to be executed in the database as a stored procedure means that the multiple SQL statements can be executed with one call of the stored procedure and only one access to the database. This greatly reduces communication processing associated with the passing of data between the HiRDB server and HiRDB client applications, as well as the overhead associated with the parsing of SQL statements by the front-end server. Figure 5-3 shows the communications processing for an SQL stored procedure.

Figure 5-3: Communications processing for an SQL stored procedure

Normal transaction processing (in which SQL stored procedure is not used)



Processing that uses an SQL stored procedure



(2) Application of a stored function

With a stored function, the user can define data processing in a database as an arbitrary

function, using conditional branching (`IF` statement) and a routine control SQL such as SQL repetition (`WHILE` statement). Consequently, the user can turn data processing into a component. When a plug-in is used, the function provided by the plug-in is registered as a stored function in the database.

(3) **Creating RDAREAs for storing stored procedures and stored functions**

To use stored procedures or stored functions, the following types of RDAREAs must be created:

- Data dictionary LOB RDAREAs
- Data dictionary RDAREAs

For details about creating RDAREAs for storing stored procedures and stored functions, see the *HiRDB Version 8 System Operation Guide*.

(4) **Creating a UAP for calling stored procedures and stored functions**

Explained as follows are the methods of calling a stored procedure or a stored function. For details about creating a UAP for calling stored procedures and stored functions, see the *HiRDB Version 8 UAP Development Guide*.

Calling a stored procedure

You call a stored procedure by specifying, in the UAP, a `CALL` statement written in SQL.

Invoking a stored function

You call a stored function by specifying a *function call* as a value in an SQL statement. You can specify arguments when the function is called. The value will be returned in the `RETURN` statement of the SQL.

(5) **Overloading of stored functions**

Multiple stored functions with the same name can be defined, as long as they have different numbers of parameters and their data types are different. Stored functions with the same name are *mutually overloaded*. Because the overload function can be used to assign the same name to multiple functions with different parameter data types, it is possible to standardize the names of functions that have the same function. When a stored function is invoked, those facilities described as follows become the candidates for execution from among the facilities with the specified name and the same number of parameters as the number of specified arguments:

- Facilities in which the data type of each argument exactly matches the data type of the corresponding parameter
- If the data type of each argument does not exactly match the data type of the corresponding parameter, arguments and parameter data types are compared sequentially from left to right, and the facility with a parameter with the highest priority among the lower-priority parameters becomes the candidate.

For details about the rules for selecting the function to be invoked, see the *HiRDB Version 8 SQL Reference*.

(6) SQL procedures and definition of user-defined functions

Explained as follows are the SQL procedures and the definition of user-defined functions. For details about SQL procedures and the definition of user-defined functions, see the *HiRDB Version 8 UAP Development Guide*.

Defining a procedure for creating a stored procedure

To create a stored procedure, one of the following SQLs must be used first to define a *procedure*:

- CREATE PROCEDURE
- *Procedure body* specified by CREATE TYPE (for specifying a procedure for an abstract data type)

Defining a user-defined function for creating a stored function

To create a stored function, one of the following SQLs must be used first to define a *user-defined function*:

- CREATE FUNCTION
- *Function body* specified by CREATE TYPE (for specifying a function for an abstract data type)

Procedures, user-defined functions, and system-defined functions are referred to collectively as routines.

(7) Registering stored procedures and stored functions into the database

When an SQL that defines a procedure or a user-defined function is executed by database definition (`pdddef`), the procedure or user-defined function is compiled automatically, thus creating an SQL object. Moreover, the definition source of the SQL and the SQL object are stored in a data dictionary LOB RDAREA. This registers the stored procedure or stored function into the database. For details about registration of stored procedures and stored functions into the database, see the *HiRDB Version 8 System Operation Guide*.

(8) Re-creating stored procedures and stored functions

If the definition of a table or index is changed or a stored procedure is deleted by the DROP PROCEDURE statement, you need to re-create the voided stored procedure by executing the ALTER PROCEDURE statement. Similarly, if a stored function is deleted by the DROP FUNCTION statement, you need to re-create the voided stored function by executing the ALTER ROUTINE statement. For details about re-creating a stored procedure or a stored function, see the *HiRDB Version 8 System Operation Guide*.

5.4 Java stored procedures and Java stored functions

Stored procedures and stored functions in which processing procedures are coded in Java are called *Java stored procedures* and *Java stored functions*. Java stored procedures and Java stored functions are referred to collectively as *Java stored routines*. This section describes Java stored routines.

5.4.1 Characteristics of a Java stored routine

A Java stored routine has the following characteristics:

1. Absence of overhead between server and client

As in the case of SQL stored procedures and SQL stored functions, Java stored routines are processed at the server, which eliminates communications overhead between server and client.

2. Ability to code the procedure or function itself in Java

The Java coding language provides more sophisticated control than is possible with SQL.

3. Ability to operate between dissimilar DBMSs

Because Java is a platform-independent language, programs coded in Java can operate on dissimilar DBMSs that support Java stored routines.

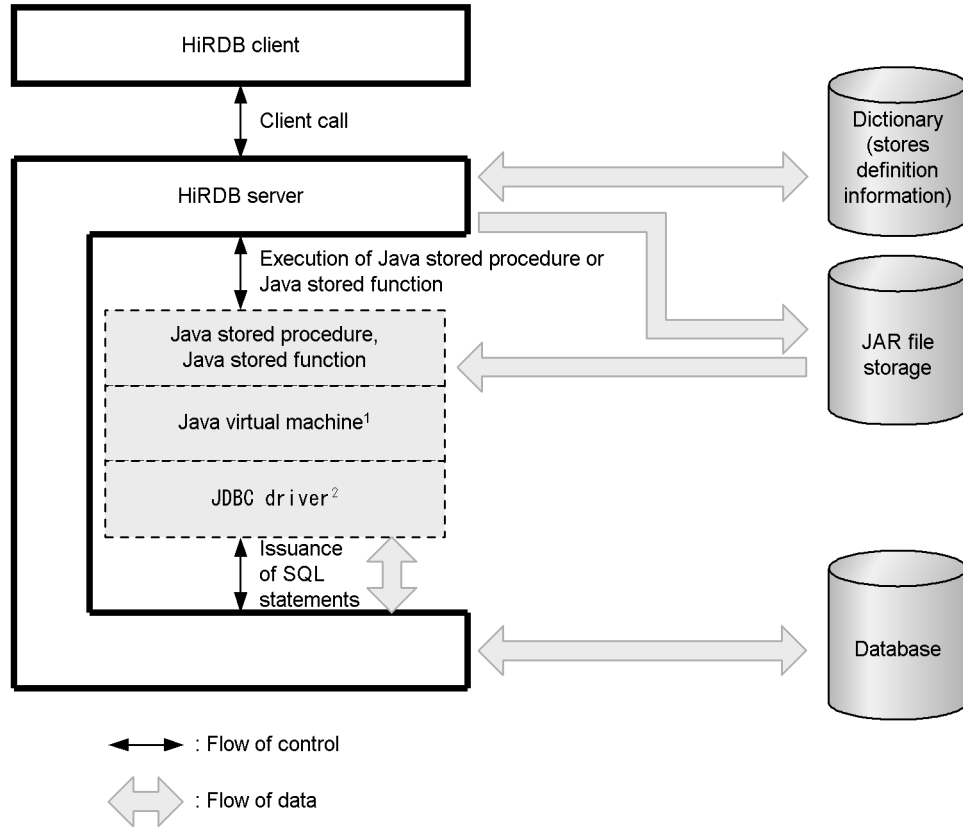
4. Simple debugging

The debugging of an SQL stored procedure or an SQL stored function requires actual execution at the server of the procedure or function. By contrast, Java stored routines can be debugged by providing a Java debugger at the client; moreover, this debugging process can address database access operations as well.

5.4.2 System configuration (position of a Java virtual machine)

Figure 5-4 shows the position of a Java virtual machine in a HiRDB system.

Figure 5-4: Position of Java virtual machine in a HiRDB system



¹ The Java virtual machine is included in the JRE (Java Runtime Environment). You can find information about and obtain the JRE from the Web sites of various platform vendors.

² The JDBC driver is a standard interface provided with HiRDB.

For details about how to obtain the JRE and system configuration examples, see the *HiRDB Version 8 System Operation Guide*.

5.4.3 Execution of Java stored routines

Before you can execute Java stored routines, you must install the JDBC driver. To do so, select the JDBC driver when you perform a HiRDB client installation.

For details about setting an environment for using Java stored procedures or Java stored functions, see the *HiRDB Version 8 System Operation Guide*.

5.4.4 Java stored routine creation and execution procedure

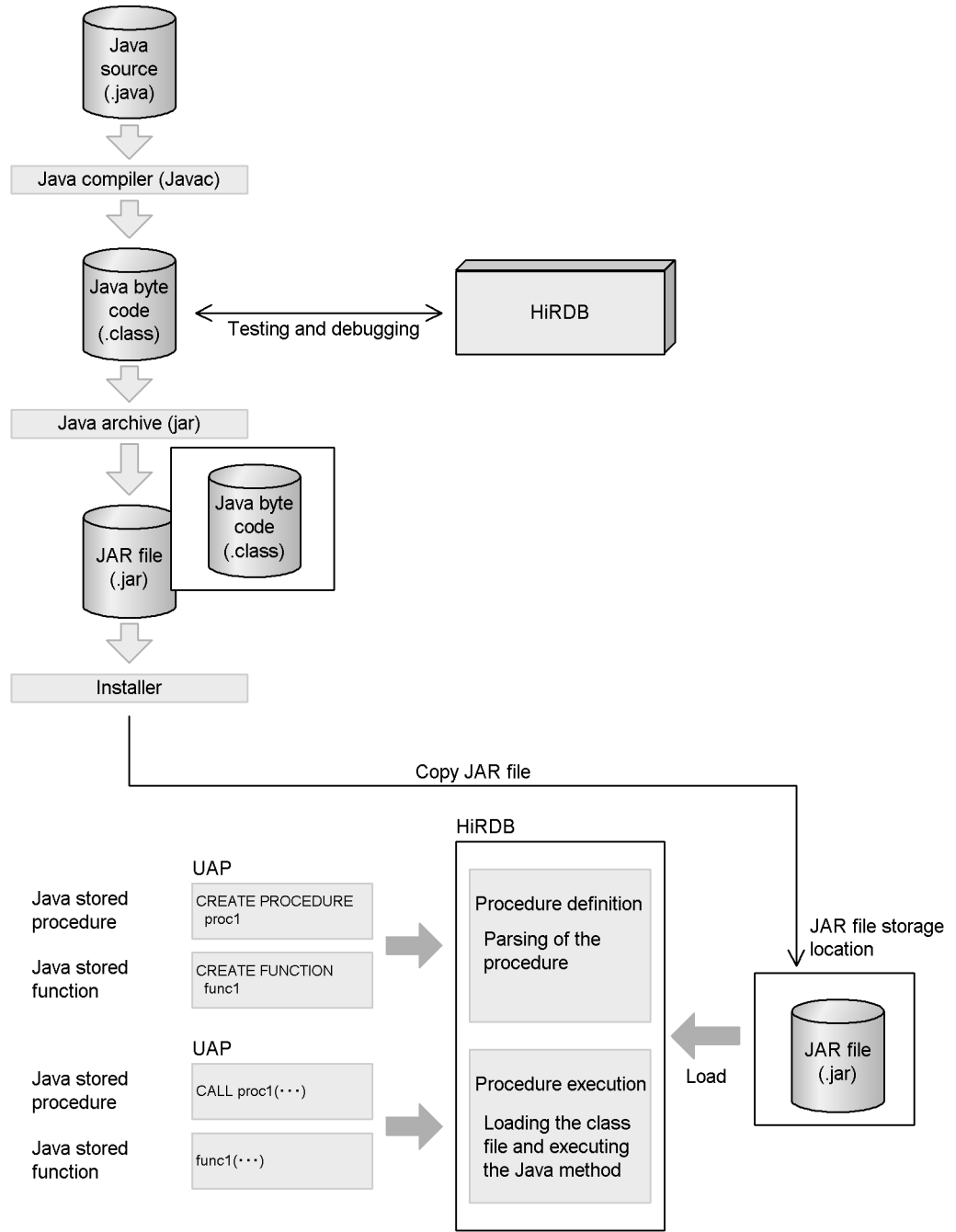
Following is the procedure for creating and executing a Java stored routine:

Procedure

1. Code the Java stored routine.
2. Register into HiRDB.
3. Define the Java stored routine.
4. Execute the Java stored routine.

Figure 5-5 shows the flow of events from creation of a Java stored routine to its execution.

Figure 5-5: Flow from creation to execution of a Java stored routine



(1) Coding a Java stored routine

In this step, you code either a procedure or a function in Java, then you compile the resulting Java program. Compilation produces a Class file. You can then test and debug the Class file, using the Java virtual machine at the client, and create a JAR file from the Class file.

(2) Registering into HiRDB

In this step, you register the JAR file into HiRDB.

Registration by a HiRDB administrator:

The HiRDB administrator uses the `pdjarsync` command.

Registration by a programmer:

The programmer uses the `INSTALL JAR` or `REPLACE JAR` statement in the embedded language. These SQL statements can be coded in either the `pddef` file or the UAP, and then executed.

(3) Defining the Java stored routine

You use the `CREATE PROCEDURE` or the `CREATE FUNCTION` statement to define a Java stored routine from the JAR file.

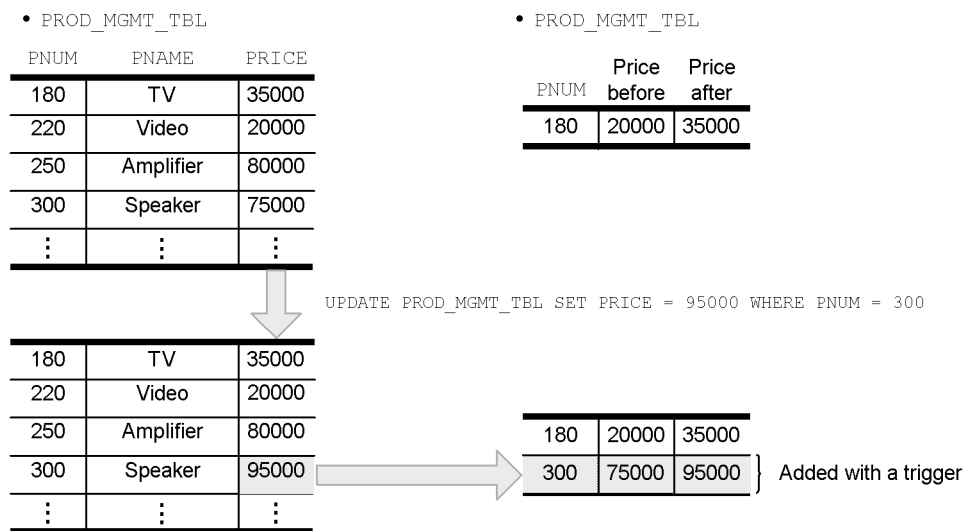
(4) Executing the Java stored routine

As in the case of executing a stored procedure or a stored function, you execute the SQL by specifying either a `CALL` statement or a function call. The `CALL` statement causes a Java method to be executed as a Java stored procedure. The function call causes a Java method to be executed as a Java stored function. The Java stored routine is executed on the Java virtual machine at the server.

5.5 Triggers

By defining a trigger, you can have SQL statements automatically execute when an operation (updating, insertion, deletion) is performed on a particular table. A trigger defines such specifications as the table to which the trigger applies, the SQL statement that activates the trigger (trigger event SQL), the SQL statements that are to be executed automatically (trigger SQL statements), and conditions for execution of the action (trigger action search conditions). When an SQL statement that satisfies the trigger action search conditions is executed on a table for which a trigger is defined, the trigger SQL statements are automatically executed. Figure 5-6 provides an overview of triggers.

Figure 5-6: Overview of triggers



Explanation

A trigger is defined to provide cumulative updating of data in the product management historical table on price changes that occur in the product management table. In this example, when the trigger event SQL statement (UPDATE in this case) is executed from the UAP, the product number, the price before the change, and the price after the change in the row containing the changed price are added to the product management historical table.

Note, however, that when you define a trigger for a table, you must re-generate any function, procedure, or trigger SQL object that uses that table, because any SQL object that uses that table becomes invalid. Similarly, if you define, change, or delete any resources that are used by the trigger (tables, indexes, and so on), the trigger SQL

object becomes invalid, so you must re-generate it as well. For details, see the *HiRDB Version 8 Installation and Design Guide*.

(1) Application criteria

Consider using a trigger to perform the following actions when an associated UAP operation occurs:

- To always update one table whenever a particular table is updated
- To always update particular columns associated with the updated row whenever a particular table is updated (relating columns)

(2) Before you define a trigger

When you define a trigger, an SQL object that codes the specified trigger action procedure is automatically generated and stored in the corresponding data dictionary LOB RDAREA. Thus, before you define a trigger, you must make sure that the data dictionary LOB RDAREA has sufficient capacity. For details about how to estimate the capacity of data dictionary LOB RDAREAs, see the *HiRDB Version 8 Installation and Design Guide*.

In addition, if you plan to execute trigger event SQL statements, you must also take into consideration the trigger SQL object size when you specify the buffer length for SQL objects. For details about how to estimate the buffer length for SQL objects, see the manual *HiRDB Version 8 System Definition*.

(3) Trigger definition statements

The following definition SQL statements are used to define a trigger, to re-generate an SQL object, and to delete a trigger.

`CREATE TRIGGER`

This statement defines a trigger. You can define triggers only for tables that you own. You cannot define triggers for tables owned by other users.

`ALTER TRIGGER`

This statement re-generates the SQL object of a trigger that has already been defined. You can also use the `ALTER ROUTINE` statement to re-generate SQL objects.

`DROP TRIGGER`

This statement deletes a trigger.

5.6 Integrity constraints

A constraint that guarantees that data in a database is valid is called an integrity constraint. Integrity constraints are set up in order to prevent accesses to the database from causing the data to lose its integrity. HiRDB supports two integrity constraints:

- NOT NULL constraint
- Uniqueness constraint

5.6.1 NOT NULL constraint

You can define the *NOT NULL constraint* for a column, and thereby prohibit the null value as data in the column. The NOT NULL constraint is specified on a column-by-column basis. When the NOT NULL constraint is defined for a column, a row containing the null value in that column cannot be added, nor can a row be updated so as to set the null value in that column. A column for which the NOT NULL constraint is defined must have a definite value in every row. An attempt to set the null value in such a column results in a constraint violation.

The NOT NULL constraint is set by specifying the `NOT NULL` option in the `CREATE TABLE` statement. For details about the NOT NULL constraint, see the *HiRDB Version 8 UAP Development Guide*.

Comment

The NOT NULL constraint is applied to columns for which the primary key is defined.

5.6.2 Uniqueness constraint

You can define the *uniqueness constraint* for a column, and thereby prohibit duplication of data in the column (each data entry in the column must be unique). When the uniqueness constraint is defined for a column, a row cannot be added if it contains a nonunique value in that column, nor can a row be updated so as to set a nonunique value in that column. Because columns for which the uniqueness constraint is defined are always required to have unique values, an attempt to assign a non-unique value to such a column results in a constraint violation. The uniqueness constraint can be specified for the following columns:

- Columns for which a cluster key is defined
- Columns for which an index key is defined

The uniqueness constraint is set for a column for which a cluster key is defined by specifying the `UNIQUE CLUSTER KEY` option in the `CREATE TABLE` statement. Similarly, the uniqueness constraint is set for a column for which an index key is defined by specifying the `UNIQUE` option in the `CREATE INDEX` statement. For details

about the uniqueness constraint, see the *HiRDB Version 8 UAP Development Guide*.

Comment

The uniqueness constraint is applied to columns for which the primary key is defined.

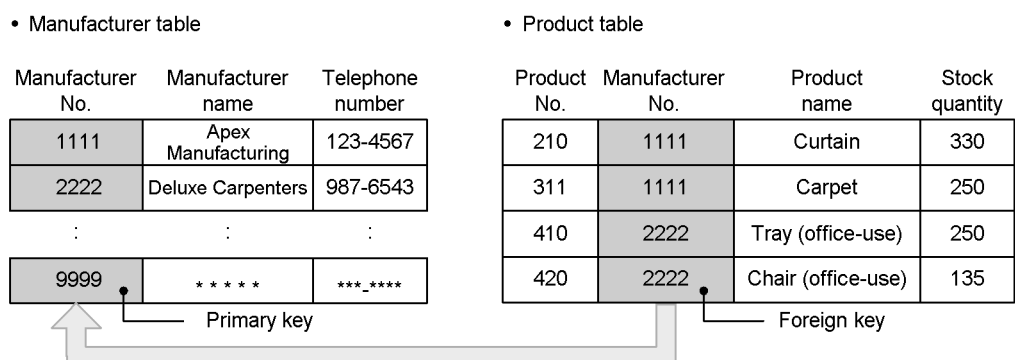
5.7 Referential constraints

Tables in a database are normally not independent entities, because typically there are links between tables. The data in a table that is not linked is probably of no value to other tables. To maintain referential integrity of data that is linked across tables, the concept of *referential constraints* is introduced. A referential constraint is defined for a specific column (called a *foreign key*) when a 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 with a foreign key is called a *referenced table*. In a referenced table, a primary key that is referenced by the foreign key must be pre-defined.

Note that execution of SQL code or a utility may cause a loss of referential integrity between a referenced table and a referencing table. When this occurs, the referencing table is placed in check pending status. For details about check pending status, see 5.9 *Check pending status*.

Figure 5-7 shows an example of a referenced table and a referencing table. In this example, the product table is the referencing table, and the manufacturer table is the referenced table. The foreign key in the referencing table references the primary key to determine the name of the manufacturer.

Figure 5-7: Example of referenced table and referencing table



For details about referential constraints, see the *HiRDB Version 8 Installation and Design Guide*.

Advantages of referential constraints

Defining a referential constraint enables HiRDB to check the integrity of data between tables and, because data operations can be automated, reduces the workload to create UAPs. The downside is that updating referenced tables and referencing tables takes longer because of the processing time required to check

the integrity of the data.

(1) Defining a referential constraint

To enable a referential constraint, you must first define in the referenced table the primary key that is referenced by the foreign key. To do so, use the `CREATE TABLE` definition SQL statement to specify `PRIMARY KEY` in the referenced table. To apply check pending status, specify `USE` in the `pd_check_pending` operand or omit specification of the `pd_check_pending` operand.

To define a referential constraint, you first specify `FOREIGN KEY` in the referencing table. In the `FOREIGN KEY` clause, you specify the following:

- Column to be referenced
- Referenced table
- Referential constraint action

A referential constraint action is an insertion, update, or deletion action in the referenced table; it is specified with `CASCADE` or `RESTRICT`.

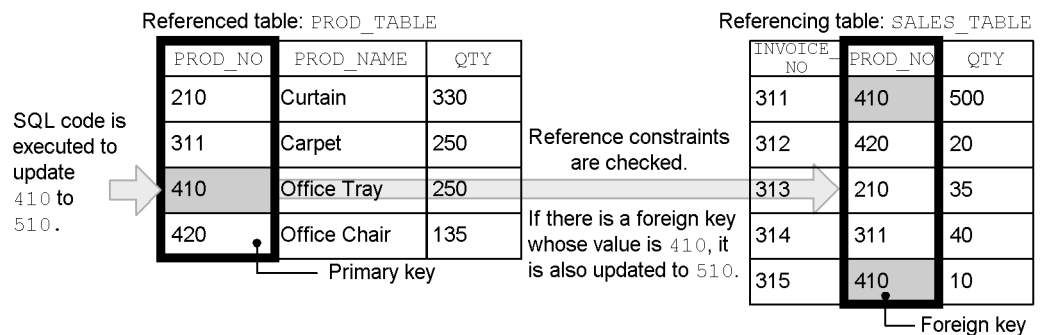
The following subsections explain the actions in the referenced table and the referencing table when `CASCADE` and `RESTRICT` are specified.

(a) When `CASCADE` is specified

When `CASCADE` is specified and a primary key of a referenced table is changed, the foreign key is also changed. When a foreign key of a referencing table is changed, a check is performed to determine if there is a row containing a primary key whose value is the same as the value of the foreign key after the change; the foreign key is not changed if such a change would result in a referential constraint violation.

Figures 5-8 and 5-9 show examples of the actions that occur when `CASCADE` is specified when SQL code is executed for a referenced table and for a referencing table.

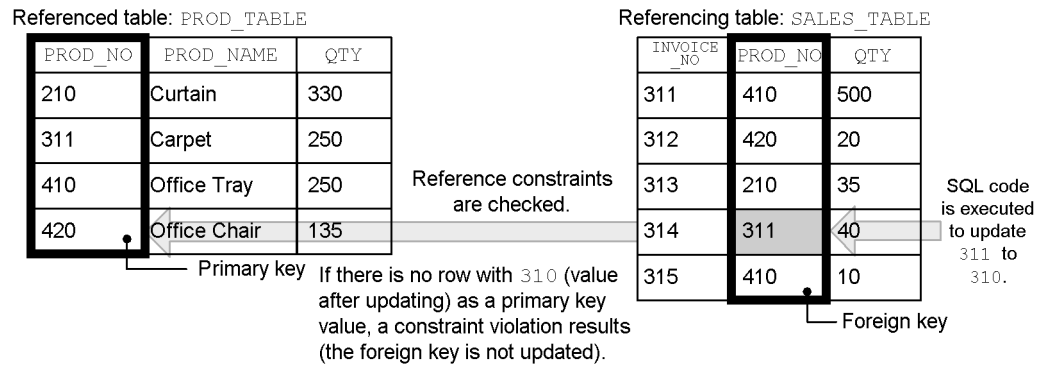
Figure 5-8: Example of the action when update SQL code is executed for a referenced table (with `CASCADE` specified)



Explanation

If a row contains a foreign key with the same value as the primary key, then in order to maintain constraints the foreign key is changed in the same way that the primary key is changed. In this case, updating is performed on the referenced table; insertion and deletion would be handled in the same manner.

Figure 5-9: Example of the action when update SQL code is executed for a referencing table (with CASCADE specified)



Explanation

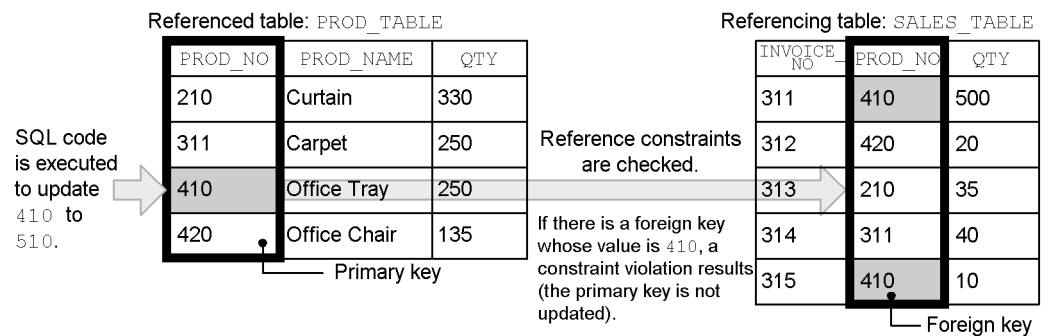
If a row contains a primary key with the same value as that of the foreign key after updating, the update is executed on the foreign key. The update is also executed if the foreign key contains a null value, even if there is no row containing a primary key with the same value. If there is no null value, a referential constraint violation results. If this occurs, there is no effect on the referenced table. Insertion and deletion would be handled in the same manner.

(b) When RESTRICT is specified

When RESTRICT is specified, then when a primary key of a referenced table is changed, a referential constraint violation occurs if there is a row that contains a foreign key with the same value. If the foreign key can be changed, a check is performed to determine if a row contains a primary key with the same value and, if a referential constraint violation would result, the foreign key is not changed.

Figure 5-10 shows the actions that occur when RESTRICT is specified when SQL code is executed for a referenced table. The action that occurs in a referencing table is the same as when CASCADE is specified (see *Figure 5-9*).

Figure 5-10: Example of the action when update SQL code is executed for a referenced table (with RESTRICT specified)



(2) Data manipulation and integrity

When you use a data manipulation SQL (other than `PURGE TABLE`) to manipulate data in a referenced table or referencing table, HiRDB performs checking when the SQL code is executed in order to guarantee data integrity. However, if you perform the data operations shown below, HiRDB may no longer be able to guarantee data integrity.

- Loading data with the database load utility (`pdload`)
- Reorganization or reloading with the database reorganization utility (`pdrorg`)
- Re-initialization of an RDAREA (`initialize rdarea`) with the database structure modification utility (`pdmod`)
- Execution of the `PURGE TABLE` statement
- Changing the partition storage conditions of a table with the `ALTER TABLE` statement

If you do perform any of these data operations, you must verify the integrity of the data. For details about how to verify data integrity, see the *HiRDB Version 8 Installation and Design Guide*. If `USE` is specified in the `pd_check_pending` operand, the referencing table is placed in check pending status if you perform any of these data operations.

5.8 Check constraints

A characteristic of databases is that value ranges, conditions, and other limitations are often imposed on the data contained in the tables. For example, when product information is stored in a database, the product cost cannot be a negative value. This means that there must not be negative product cost values in the database, which suggests that these values should be checked when they are inserted or updated. A check constraint is a function that maintains the integrity of table data by checking a constraint condition when data is inserted or updated and by suppressing operations on data that does not satisfy the condition. In this manual, a table for which a check constraint has been defined is called a check constraint table.

Note that execution of a utility may make it impossible to guarantee the integrity of data in a check constraint table. When this occurs, the check constraint table is placed in check pending status. For details about check pending status, see *5.9 Check pending status*.

For details about check constraints, see the *HiRDB Version 8 Installation and Design Guide*.

Advantages of check constraints

Defining a check constraint reduces the workload required to create a UAP, because the checking of data when it is inserted or updated is automated. The downside is that updating check constraint tables takes longer because of the processing time required to check the integrity of the data.

(1) Defining a check constraint

To define a check constraint, you first specify `CHECK` with `CREATE TABLE` of a definition SQL, and then use a search condition to specify the constraint condition for the table value. To use the check pending status, specify `USE` in the `pd_check_pending` operand or omit specification of the `pd_check_pending` operand.

(2) Data manipulation and integrity

When you use a data manipulation SQL to update, add, or delete data in a check constraint table, HiRDB performs checking when the SQL is executed in order to guarantee data integrity. However, if you perform certain data operations with the utilities listed below, HiRDB may no longer be able to guarantee data integrity.

- Loading data with the database load utility (`pdload`)
- Reloading data with the database reorganization utility (`pdreorg`)

If you do perform any of these data operations, you must verify the integrity of the data. For details about how to verify data integrity, see the *HiRDB Version 8*

Installation and Design Guide. When `USE` is specified in the `pd_check_pending` operand, the referencing table is placed in check pending status if you perform any of these data operations.

5.9 Check pending status

When the integrity of table data can no longer be guaranteed, such as when SQL code or a utility has been executed on a table for which a referential constraint or a check constraint is defined, HiRDB limits the data operations that can be performed on the referencing table or the check constraint table. The status in which data operations are limited in this way because integrity cannot be guaranteed is called *check pending status*. To place a referencing table or a check constraint table in check pending status in order to limit data operations, either you must specify `USE` in the `pd_check_pending` operand or you must omit specification of the `pd_check_pending` operand. You can remove check pending status from a table by executing the *integrity check utility* (`pdconstck`). You can also use the integrity check utility to forcibly place a table in check pending status.

If you specify `NOUSE` in the `pd_check_pending` operand, data operations are not limited even if referential integrity between tables can no longer be guaranteed. In such a case, when you execute SQL code or a utility that may cause integrity to be lost, it is advisable to use the integrity check utility to forcibly place the table in check pending status and then check its integrity.

For details about check pending status and how to check the integrity of data on which the integrity check utility has been applied, see the *HiRDB Version 8 Installation and Design Guide*.

(1) Events that set or remove check pending status

In addition to the integrity check utility, the following utilities, commands, and SQL code can assign or remove check pending status for a referencing table.

- Specification with the `constraint` statement of the database load utility (`pdload`)
- Specification with the `constraint` statement of the database reorganization utility (`pdrorg`) (reload or reorganize)
- Use of the database structure modification utility (`pdmod`) (re-initialization of an RDAREA)
- `PURGE TABLE` statement
- `ALTER TABLE (CHANGE RDAREA)`

For details about these utilities and commands, see the manual *HiRDB Version 8 Command Reference*. For details about SQL, see the manual *HiRDB Version 8 SQL Reference*.

(2) Operations that are limited for a check pending status table

Table 5-1 lists the operations that cannot be performed on a table that has been placed

in check pending status.

Table 5-1: Operations that cannot be performed on a check pending status table

Operation attempted on check pending status table	Result
Rebalancing utility (<code>pdrbal</code>)	Cannot be performed.
SELECT statement (searching a target table or searching a list created from a target table) INSERT statement (insertion of data into a target table) UPDATE statement (updating a target table) DELETE statement (deleting rows from a target table) ASSIGN LIST statement (creating a list from a target table)	Each of these operations can be performed if either of the following conditions is satisfied; these operations cannot be performed in any other case: <ul style="list-style-type: none"> • The operation is performed on a partitioned table whose storage condition is either key range partitioning or FIX hash partitioning • The RDAREA on which the operation is performed is not in check pending status
<code>pdrorg</code> (reorganizing)	You may not be able to perform reorganization on a partitioned table that has been reorganized with flexible hash partitioning. For details, see the section on the <i>Database Reorganization Utility (pdrorg)</i> in the manual <i>HiRDB Version 8 Command Reference</i> .

5.10 Improving database access performance

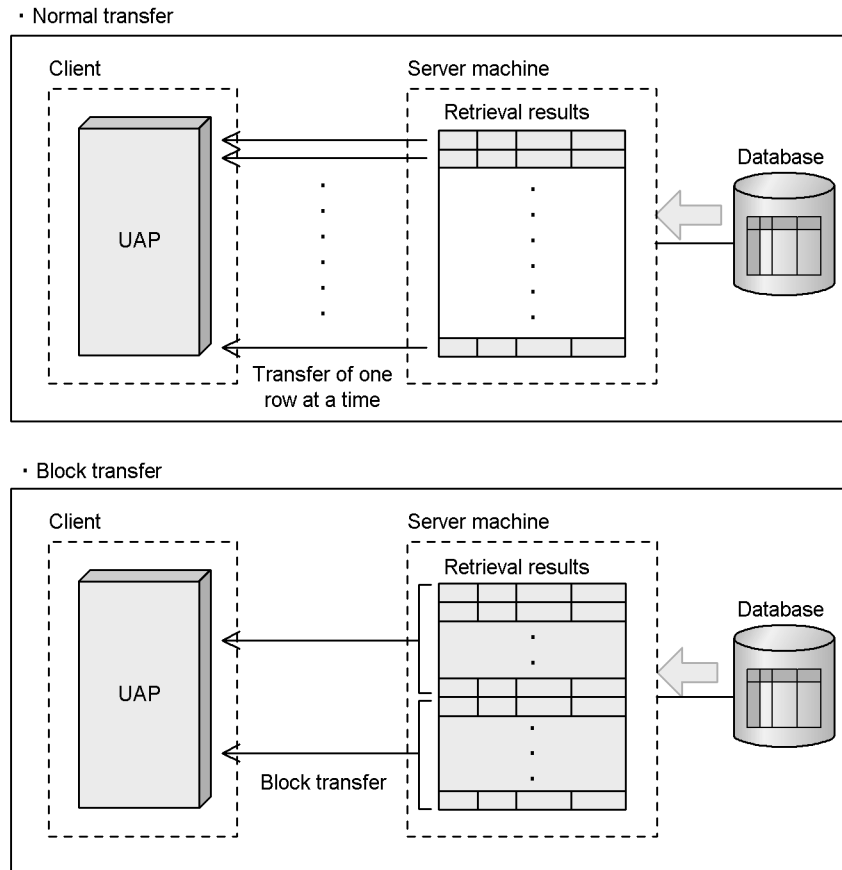
HiRDB provides the following functions that are designed to improve database access performance:

- Block transfer facility
- Rapid grouping facility
- Functions that use arrays
- Holdable cursor
- SQL optimization

5.10.1 Block transfer facility

The facility for transferring data from a HiRDB server to a HiRDB client in blocks of any number of rows is called the block transfer facility. The block transfer facility can improve retrieval performance when it is used for accessing a HiRDB server from a HiRDB client to retrieve a large volume of data. Although increasing the number of rows to be transferred can decrease the communications overhead and reduce retrieval time, it also increases the amount of memory required. Moreover, if the number of rows to be transferred is too large, resend will occur during communication, which requires additional time. The larger the number of rows to be transferred, the more frequently resend occurs, negating the benefit of the block transfer facility. For this reason, the number of rows specified for block transfer should not be too large. Figure 5-11 provides an overview of the block transfer facility.

Figure 5-11: Overview of block transfer facility



Using the block transfer facility

The block transfer facility can be used when the following conditions are satisfied:

1. A value of 2 or greater is specified for `PDBLKF` in the client environment definition or a value of 1 or greater is specified in `PDBLKBUFSIZE`.
2. The `FETCH` statement is specified, except under any of the following conditions:
 - An update uses a cursor
 - A search includes a `BLOB`-type selection expression
 - A search includes a `BINARY`-type selection expression whose definition length is more than 32,000 bytes, with the `PDBINARYBLKF` client

environment definition set to NO (or omitted)

- A search accepts a result that uses a BLOB locator type or BINARY locator type variable, and uses a holdable cursor

For details about client environment definition, see the *HiRDB Version 8 UAP Development Guide*.

5.10.2 Rapid grouping facility

When the GROUP BY clause of an SQL is used for grouping, sorting is performed before grouping. The facility for combining this grouping with hashing to achieve rapid grouping is called the rapid grouping facility. This facility can reduce the amount of time required for grouping when the number of groups is small and there is a large number of rows to be grouped.

Specifying the rapid grouping facility

To specify the rapid grouping facility, use an SQL optimization option, such as one of the following operands:

- `pd_optimize_level` system definition operand
- `PDSQLOPTLVL` client environment definition
- `CREATE PROCEDURE`, `CREATE TRIGGER`, `CREATE TYPE`, `ALTER PROCEDURE`, `ALTER ROUTINE`, or `ALTER TRIGGER` SQL optimization options

For details about the `pd_optimize_level` operand, see the *HiRDB Version 8 System Definition*. For specification of the `PDSQLOPTLVL` statement and details of the rapid grouping facility, see the *HiRDB Version 8 UAP Development Guide*.

5.10.3 Functions that use arrays

(1) *FETCH* facility using arrays

When an arrayed embedded variable is specified for the INTO clause of a `FETCH` statement, it becomes possible to obtain multiple rows of retrieval results at the same time. The `FETCH` facility using arrays can improve retrieval performance when it is used for accessing a HiRDB system from a HiRDB client to retrieve a large volume of data. Unlike the block transfer facility, the `FETCH` facility using arrays requires the fact that multiple rows of retrieval results will be obtained to be described in a program. The `FETCH` facility using arrays is valid only when the embedded variables and indicator variables specified in the INTO clause are all arrayed. For details about the `FETCH` facility using arrays, see the *HiRDB Version 8 UAP Development Guide*.

(2) *INSERT* facility using arrays

The `INSERT` facility using arrays makes it possible to execute a single SQL `INSERT` statement to insert multiple rows of data by specifying an array type variable in which

the multiple rows of data are set. By using the INSERT facility using arrays, you can greatly reduce the frequency of communications between the HiRDB client and the HiRDB server. If you are using HiRDB/Parallel Server, you can further reduce the frequency of communications among server applications on the HiRDB server as well. Accordingly, this facility works well for cases in which a HiRDB client accesses a HiRDB server to quickly insert a large amount of data.

How to use the INSERT facility using arrays

Static execution

With the INSERT statement, specify an embedded variable in the FOR clause, and change all embedded variables and indicator variables specified in the VALUES clause to be array type variables. Use the embedded variable specified in the FOR clause to control the number of rows that are inserted at one time.

Dynamic execution

Using the following procedure to execute:

1. Using the PREPARE statement, preprocess the INSERT statement (specify the ? parameter for all values in the VALUES clause that are to be inserted).
2. In the USING clause of the EXECUTE statement, use an array to specify the values to be assigned to the ? parameters entered in the preprocessed INSERT statement, and specify the embedded variable in the BY clause. Use the embedded variable specified in the BY clause to control the numbers of rows that are inserted at one time.

For details about the INSERT facility using arrays, see the *HiRDB Version 8 UAP Development Guide*.

(3) UPDATE facility using arrays

The UPDATE facility using arrays enables you to execute a single SQL UPDATE statement to update multiple table columns by specifying an array-type variable in which multiple data items are set. Because the frequency of communication between the HiRDB client and the HiRDB server is reduced, this facility works well in cases where a HiRDB client accesses a HiRDB server to perform high-speed updating of a large amount of data.

How to use the UPDATE facility using arrays

Static execution

Specify an embedded variable in the FOR clause of the UPDATE statement, and then change all of the embedded variables and indicator variables specified in the search conditions so that they are array-type variables. Use the embedded variable specified in the FOR statement to control the number

of data items that are updated at one time.

Dynamic execution

To perform dynamic execution:

1. Use a `PREPARE` statement to preprocess the `UPDATE` statement (specify the `?` parameter for the update values or in the search condition).
2. In the `USING` clause of the `EXECUTE` statement, use an array to specify the values to be assigned to the `?` parameter entered in the preprocessed `UPDATE` statement, and specify an embedded variable in the `BY` clause. Use the embedded variable specified in the `BY` clause to control the number of data items that are updated at one time.

For details about the `UPDATE` facility using arrays, see the *HiRDB Version 8 UAP Development Guide*.

(4) DELETE facility using arrays

The `DELETE` facility using arrays enables you to execute a single SQL `DELETE` statement to delete multiple rows by specifying an array-type variable in which multiple data items are set. Because the frequency of communication between HiRDB client and HiRDB server is reduced, this facility works well in cases where a HiRDB client accesses a HiRDB server to perform high-speed deletion of a large amount of data.

How to use the `DELETE` facility using arrays

Static execution

Specify an embedded variable in the `FOR` clause of the `DELETE` statement, and then change all of the embedded variables and indicator variables specified in the search conditions so that they are array-type variables. Use the embedded variable specified in the `FOR` statement to control the number of data items that are deleted at one time.

Dynamic execution

To perform dynamic execution:

1. Use the `PREPARE` statement to preprocess the `DELETE` statement (specify the `?` parameter in the search conditions).
2. In the `USING` clause of the `EXECUTE` statement, use an array to specify the values to be assigned to the `?` parameter entered in the preprocessed `DELETE` statement, and specify the embedded variable in the `BY` clause. Use the embedded variable specified in the `BY` clause to control the number of data items that are deleted at one time.

For details about the `DELETE` facility using arrays, see the *HiRDB Version 8 UAP Development Guide*.

5.10.4 Holdable cursor

A cursor that is not to be closed even when a `COMMIT` statement is executed is called a holdable cursor. When a holdable cursor is specified, it can be kept open until one of the following SQL statements is executed:

- `CLOSE` statement
- `DISCONNECT` statement (including the `DISCONNECT` statement that is executed automatically by the system after an error occurs)
- `ROLLBACK` statement (including the `ROLLBACK` statement that is executed automatically by the system after an error occurs)

A holdable cursor can reduce the incidence of locked resources, because it is possible that a `COMMIT` statement will be executed in the middle of a retrieval or in the midst of updating a large volume of data. Furthermore, because the `COMMIT` statement can be executed while the cursor is open, synchronization point dumps can be enabled to reduce restart time even when a large volume of data must be retrieved or updated. For details about holdable cursors, see the *HiRDB Version 8 UAP Development Guide*.

Specifying a holdable cursor

To use a holdable cursor, specify `UNTIL DISCONNECT` or `WITH HOLD` in `DECLARE CURSOR`.

5.10.5 SQL optimization

Taking into account the database status to determine the most efficient access paths is called *SQL optimization*. SQL optimization includes the *SQL optimization specifications*, the *SQL optimization options*, and the *SQL extension optimizing options*. Table 5-2 lists and describes the SQL optimization specification facilities, Table 5-3 lists and describes the SQL optimization option facilities, and Table 5-4 lists and describes the SQL extension optimizing option facilities. Note, however, that there are additional SQL optimization facilities that are not described here. Because these facilities invariably improve performance, they are not provided as options, but are always used.

Table 5-2: SQL optimization specification facilities

SQL optimization specification facility	Description
SQL optimization specification for index utilization	<p>Enables you to use table searches to specify which index to utilize. It also enables you to use table searches to specify not to utilize an index (by using a table scan).</p> <p>You can use the following SQL methods to specify an SQL optimization specification for index utilization:</p> <ul style="list-style-type: none"> • Table expressions • DELETE statement • UPDATE statement, format 1
SQL optimization specification for the join method	<p>Enables you to specify a join method (nest-loop-join, hash join, or merge join) for join tables.</p> <p>You can use the following SQL method to specify an SQL optimization specification for the join method:</p> <ul style="list-style-type: none"> • Table expressions
SQL optimization specification for the subquery execution method	<p>Enables you to specify whether or not to set hash execution as the method of executing subqueries in predicates.</p> <p>You can use the following SQL method to specify the SQL optimization specification for the subquery execution method:</p> <ul style="list-style-type: none"> • Subqueries

Table 5-3: Facilities of SQL optimization options

SQL optimization option facility	Explanation
Forced nest-loop-join	If an index is defined in the joining condition column, only nest loop join is used for joining. For details about the nest-loop-joining method, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Creating multiple SQL objects	Creates multiple SQL objects in advance and selects the optimal SQL object based on an embedded variable or ? parameter value during execution. For details about the creating multiple SQL objects, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Increasing the target floatable servers (back-end servers for fetching data)	Normally, a back-end server not used for fetching data is used as a floating server. When this optimization method is used, a back-end server used for fetching data is also used as a floating server. However, because the number of back-end servers that can be used as floating servers is computed by HiRDB, not all back-end servers can be used as floating servers. To use all back-end servers, specify both this facility and the increasing the number of floatable server candidates facility. For details about floating server allocation, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Prioritized nest-loop-join	If an index is defined for the joining condition column, nest-loop-join is used for the joining process with a higher priority. For details about the nest-loop-join method, see the <i>HiRDB Version 8 Command Reference</i> . For details about the nest-loop-joining method, see the <i>HiRDB Version 8 UAP Development Guide</i> .

SQL optimization option facility	Explanation
Increasing the number of floatable server candidates	Normally, the needed number of floating servers is computed by HiRDB from the available floating servers and allocated. When this optimization method is applied, all available floating servers are used. However, back-end servers used for fetching data cannot be used as floating servers. To also use the back-end servers used for fetching data as floating servers, specify this facility together with the increasing the target floatable servers (back-end servers for fetching data) facility. For details about floating server allocation, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Priority of OR multiple index use	Specify this facility to use a retrieval method that uses OR multiple indexes with a higher priority. OR multiple index use is a method that evaluates a search condition by using an index to retrieve each condition from multiple conditions joined with ORs in a search condition and obtains a sum set of the retrieval results. For details about the priority of OR multiple index use, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Group processing, ORDER BY processing, and DISTINCT set function processing at the local back-end server	Specify this facility to perform group processing, ORDER BY processing, and DISTINCT set function processing at the back-end server that defines the table (local back-end server), without having to use a floating server. For details about the group processing method, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Suppressing use of AND multiple indexes	Ensures that an access path using AND multiple indexes is never used. <ul style="list-style-type: none"> When the search condition contains multiple conditions linked with AND and different indexes are defined for the individual columns (e.g., SELECT ROW FROM T1 WHERE C1=100 AND C2=200), use of AND multiple indexes means that individual indexes are used to create work tables for the rows that satisfy the condition and the product set of these rows is obtained. For details about the suppressing use of AND multiple indexes, see the <i>HiRDB Version 8 UAP Development Guide</i>.
Rapid grouping facility	Uses hashing to rapidly process the grouping specified by the GROUP BY clause of an SQL. For details about the grouping method, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Limiting the target floatable servers (back-end servers for fetching data)	Normally, a back-end server not used for fetching data is used as a floating server. When this optimization method is used, only those back-end servers used for fetching data are used as floating servers. For details about floating server allocation, see the <i>HiRDB Version 8 UAP Development Guide</i> .

SQL optimization option facility	Explanation
Separating data collecting servers	If increasing the target floatable servers (back-end servers for fetching data) and limiting the target floatable servers (back-end servers for fetching data) are both specified, they will function as the facility for separating data collecting servers. If the facility for separating data collecting servers is applied, a back-end server other than the data transfer source is allocated for data collection to an SQL that needs to collect data from multiple back-end servers into a single back-end server. Back-end servers other than those used for collecting data (including back-end servers for fetching data) are allocated as front-end servers for other purposes. For details about the floating server allocation method, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Suppressing index use (forced table scan)	Normally, HiRDB determines whether indexes should be used. When this optimization method is applied, methods that do not use indexes are given higher priority. However, index use cannot be suppressed in the following cases: joining results in a nest-loop-join, a structured repetition predicate is specified as the search condition, or a condition includes the index type plug-in dedicated facility. For details about the suppressing index use, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Forcing use of multiple indexes	Specify this optimization method to retrieve tables by forcibly selecting use of AND multiple indexes. When this optimization method is not specified and multiple conditions linked with AND have been specified, two indexes at most are used even if use of AND multiple indexes is selected. The number of indexes to be used will vary slightly depending on the table definition, index definition, and search condition. When this optimization is specified, all conditions that can narrow the search range using indexes will be used. When AND multiple indexes are used, search candidates can be narrowed down to a certain degree using these indexes. Moreover, AND multiple indexes are effective when there is little overlap in the product set. For details about the forcing use of multiple indexes, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Suppressing creation of update-SQL work tables	If this optimization is specified when index key value no locking is being applied, HiRDB does not create work tables for internal processing even if an index is used for a retrieval with the <code>FOR UPDATE</code> clause specified, an <code>UPDATE</code> statement, or a <code>DELETE</code> statement. Therefore, SQL statements can be processed at high speed. Work tables will be created if the index key value no-locking option is not being applied. Whether or not indexes are being used can be determined from the access path display utility. For details about the suppressing creation of update-SQL work tables, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Derivation of rapid search conditions	If this optimization facility is specified, rapid search conditions are derived. Rapid search conditions refer to conditions ranging from search conditions in the <code>WHERE</code> clause and <code>ON</code> search conditions in the <code>FROM</code> clause, to new conditions derived from CNF conversions or condition transitions. Deriving rapid search conditions improves search performance because it allows you to narrow down the rows that are searched at an early stage. However, generating and executing rapid search conditions takes time, or unexpected access paths may be used. In general, we recommend that you avoid specifying this optimization option. Instead, specify rapid search conditions directly in an SQL statement. For details about deriving rapid search conditions, see the <i>HiRDB Version 8 UAP Development Guide</i> .

SQL optimization option facility	Explanation
Application of scalar operation-included key conditions	If this optimization facility is specified, within the limitations that specify scalar operations, if all columns included in the scalar operation are index configuration columns, searching is narrowed by evaluating these columns by one index key value at a time. These conditions are evaluated as key conditions. For details about applying key conditions included in scalar operations, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Facility for batch acquisition from functions provided by plug-ins	If you specify a function provided by a plug-in as a search condition and HiRDB performs the search using a plug-in index, HiRDB normally acquires the results (row location information and the passed value if needed) returned from the function provided by the plug-in one row at a time. By applying this optimization, HiRDB can acquire in a batch multiple rows of results returned from the function provided by the plug-in, which reduces the number of times that HiRDB has to call the function. For details about the facility for batch acquisition from functions provided by plug-ins, see the <i>HiRDB Version 8 UAP Development Guide</i> .

Table 5-4: Facilities of SQL extension optimizing options

SQL extension optimizing option facility	Explanation
Application of optimizing mode 2 based on cost	Optimizes using optimizing mode 2 based on cost. For details about optimizing mode 2 based on cost, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Hash-execution of a hash join or a subquery	Optimizes by applying hash join for a joined retrieval. If the retrieval is accompanied by a subquery, the subquery is processed by hashing. Whether or not hash-execution of a hash join or a subquery is to be applied should be decided by taking into consideration the join method and external reference. For details about join methods and external reference, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Suppression of foreign server execution of SQL statements that include a join	Suppresses creation of an SQL statement containing a join when an SQL statement that accesses a foreign table would be created from a query containing an access to a foreign table. Instead of creating an SQL statement that includes a join, this facility creates an SQL statement that retrieves the data from the foreign table that would have been inserted by the join. Note that the join processing is performed by HiRDB. For details about suppression of foreign server execution of SQL statements that include a join, see <i>HiRDB External Data Access Version 8</i> .
Forced foreign server execution of SQL statements that include a direct product	Attempts to create an SQL statement that includes a direct product when an SQL statement that accesses a foreign table is created from a query containing an access to a foreign table. For details about forced foreign server execution of SQL statements that include a direct product, see <i>HiRDB External Data Access Version 8</i> .

SQL extension optimizing option facility	Explanation
Suppression of unconditionally generated derived rapid search conditions that can be executed on foreign servers	Enables you to suppress unconditionally derived rapid search conditions that can be executed on a foreign server. Generation and execution of derived rapid search conditions may take time, or an unexpected access path may be used. Specify this optimization facility for such cases. Note that, if derivation of rapid search conditions is specified as an SQL optimization option, this optimization facility is ignored, even if specified. For details about deriving rapid search conditions, see the <i>HiRDB Version 8 UAP Development Guide</i> .

Setting SQL optimization options and SQL extension optimizing options

The following methods can be used to specify SQL optimization options and SQL extension optimizing options:

1. Specifying the `pd_optimize_level` or `pd_additional_optimize_level` operand of the system common definition or front-end server definition
2. Specifying `PDSQLOPTLVL` and `PDADDITIONALOPTLVL` in the client environment definition
3. Specifying an SQL optimization option or SQL extension optimizing option in an SQL statement of a stored routine or trigger.

If more than one of these methods is specified concurrently, the order of precedence is 3, 2, 1. For details about SQL optimization options and SQL extension optimizing options, see the *HiRDB Version 8 UAP Development Guide*.

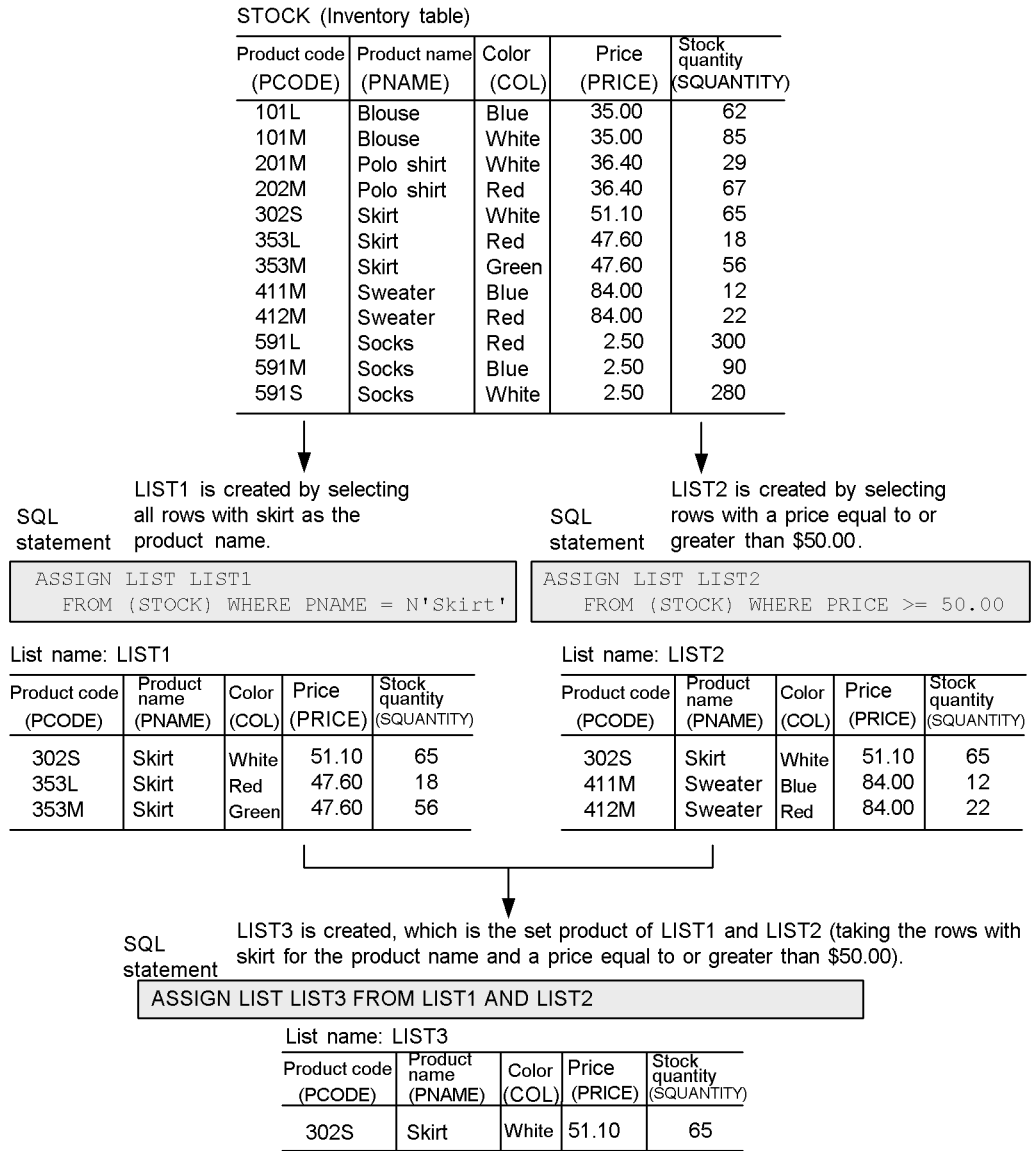
If you also specify an SQL optimization specification at the same time, the SQL optimization specification has precedence over any SQL optimization option or SQL extension optimizing option that has been specified. For details about the SQL optimization specifications, see the manual *HiRDB Version 8 SQL Reference*.

5.11 Narrowed search

Narrowed search is the process of retrieving records by gradually narrowing the search processing. In a narrowed search, the `ASSIGN LIST` statement generates a *list*, which refers to either saved data or a set of saved data with a temporary name (list name). The list is created during the process so that information can be retrieved by gradually narrowing the search for data by specifying conditions until an appropriate number of data items can be identified.

If an existing list created using known identical or substantially similar conditions is available, processing performance can be improved by using it. When multiple conditions are specified, a search can be performed by combining multiple lists. For details about narrowed searches, see the *HiRDB Version 8 UAP Development Guide*. Figure 5-12 shows an example of a search using a list.

Figure 5-12: Example of a search using a list



To retrieve the data whose product name is skirt and whose unit price is equal to or greater than \$50.00 from the inventory table (STOCK), you should search list LIST3, which is faster than searching the inventory table with conditions specified.

SQL statement

```
SELECT * FROM LIST LIST3
```



Product code (PCODE)	Product name (PNAME)	Color (COL)	Price (PRICE)	Stock quantity (SQUANTITY)
302S	Skirt	White	51.10	65

Preparations for performing a narrowed search

Following are the prerequisites for conducting narrowed searches:

Specifications in a system definition

It is necessary that the following two operands related to narrowed searches be specified in the system common definition:

- `pd_max_list_users`: Specifies the maximum number of users who are to be permitted to create lists.
- `pd_max_list_count`: Specifies the maximum number of lists that one user will be permitted to create.

For details about the narrowed search operands in the system common definition, see the *HiRDB Version 8 System Definition*.

Creating a list RDAREA

A list RDAREA can be created with either the database initialization utility (`pdinit`) or the database structure modification utility (`pdmod`). For details about list RDAREAs, see the *HiRDB Version 8 Installation and Design Guide*.

5.12 Accessing databases using DB access products

You can also use DB access products to access your databases. The following DB access products are discussed in this section:

- ODBC Driver
- HiRDB OLE DB Provider
- HiRDB.NET Data Provider
- JDBC Driver
- SQLJ

These DB access products are included in HiRDB/Run Time and HiRDB/Developer's Kit.

For details about DB access products, see the *HiRDB Version 8 UAP Development Guide*.

5.12.1 ODBC Driver

To access HiRDB from an ODBC-compatible application program, you use ODBC Driver. ODBC-compatible application programs include Microsoft Access, Microsoft Excel, and others.

ODBC Driver is also used when a UAP that uses ODBC functions accesses HiRDB. Note that the only ODBC functions that can be used are ones provided by HiRDB.

5.12.2 HiRDB OLE DB Provider

To access HiRDB from an OLE DB-compatible application program, you use HiRDB OLE DB Provider.

As with ODBC, OLE DB is an API designed for accessing a wide range of data sources. Interfaces suitable for accessing non-SQL data sources are defined as well.

5.12.3 HiRDB.NET Data Provider

To access HiRDB using ADO.NET, you use HiRDB.NET Data Provider. HiRDB.NET Data Provider complies with the ADO.NET specifications.

HiRDB.NET Data Provider bundles a set of common basic interfaces that are provided in the System.Data space of .Net Framework. In addition, it provides as extensions the INSERT facility using arrays and access to repetition columns.

5.12.4 JDBC Driver

To access HiRDB from a program written in Java, you use JDBC Driver. You also need JDBC Driver when you use JBuilder to develop Java stored procedures and Java stored

functions.

5.12.5 SQLJ

SQLJ is a language specification for using Java to write and execute static SQL statements as embedded SQL statements. SQLJ is made up of SQLJ Translator and SQLJ runtime libraries.

SQLJ Translator interprets an SQLJ source program, and generates a profile in which the Java source file and SQL information are stored. The user then runs a Java compiler to compile the Java source file and create a class file (executable file).

The SQLJ runtime libraries are used when the generated profile and class file are executed.

Chapter

6. HiRDB Architecture

This chapter explains the HiRDB architecture.

This chapter contains the following sections:

- 6.1 HiRDB environment setup
- 6.2 HiRDB file system areas
- 6.3 System files
- 6.4 Work table files
- 6.5 HiRDB system definitions
- 6.6 HiRDB startup and termination
- 6.7 Delayed rerun
- 6.8 Database access processing method
- 6.9 Transaction control
- 6.10 Locking
- 6.11 Operation without collecting a database update log

6.1 HiRDB environment setup

Hitachi provides tools designed to assist you in setting up a HiRDB environment. It is highly recommend that you use one of these tools when you set up your HiRDB environment:

- Simple setup tool
- Automatic generation from a batch file (`SPsetup.bat`)

Hint:

In general, you should use the simple setup tool to set up your HiRDB environment.

Table 6-1 shows the advantages and disadvantages of the various environment setup methods. For details about each of these setup methods, see the *HiRDB Version 8 Installation and Design Guide*.

Table 6-1: Advantages and disadvantages of the environment setup methods

Environment setup method	Overview	Advantage	Disadvantage
Simple setup tool	You enter HiRDB environment setup information as instructed on windows that are displayed. The HiRDB environment is then set up based on the information you provide.	This is the easiest method to use to set up a HiRDB environment. The simple setup tool assures you of being able to start HiRDB operations. It also allows you to change values already specified in the system definition.	There is a limited range of HiRDB system configurations that can be set up with the simple setup tool.
Commands ¹	You use HiRDB commands to set up a HiRDB environment.	This method gives you complete flexibility to set up a HiRDB system configuration in any way you wish.	A certain amount of knowledge of HiRDB is required in order to use this method. Basically, an understanding of the facilities and settings described in this manual is required. This environment setup method is more difficult than the other methods.
Batch file (<code>SPsetup.bat</code>) ²	You execute a batch file to set up a HiRDB environment.	This is an easier method for setting up a HiRDB environment than the commands method.	There is a limited range of HiRDB system configurations that can be set up with a batch file.

1

Before you attempt to set up an actual system for production use, you should try performing a simple installation. Once you become familiar with the procedures for configuring a HiRDB system by using a sample file to set up a test system, you should be ready to set up a production-use system that is more appropriate to your needs.

For details about how to perform a simple installation, see the *HiRDB Version 8 Installation and Design Guide*.

2

The batch file method can be used only for a HiRDB/Single Server system. For details about how to set up a HiRDB/Parallel Server environment, see `%PDDIR%\HiRDEF\readme.txt`.

Note:

- You cannot use the simple setup tool to set up a plug-in environment.
- A batch file automatically configures HiRDB environment setting information. The HiRDB system administrator can then make changes to the values to configure a more appropriate environment.

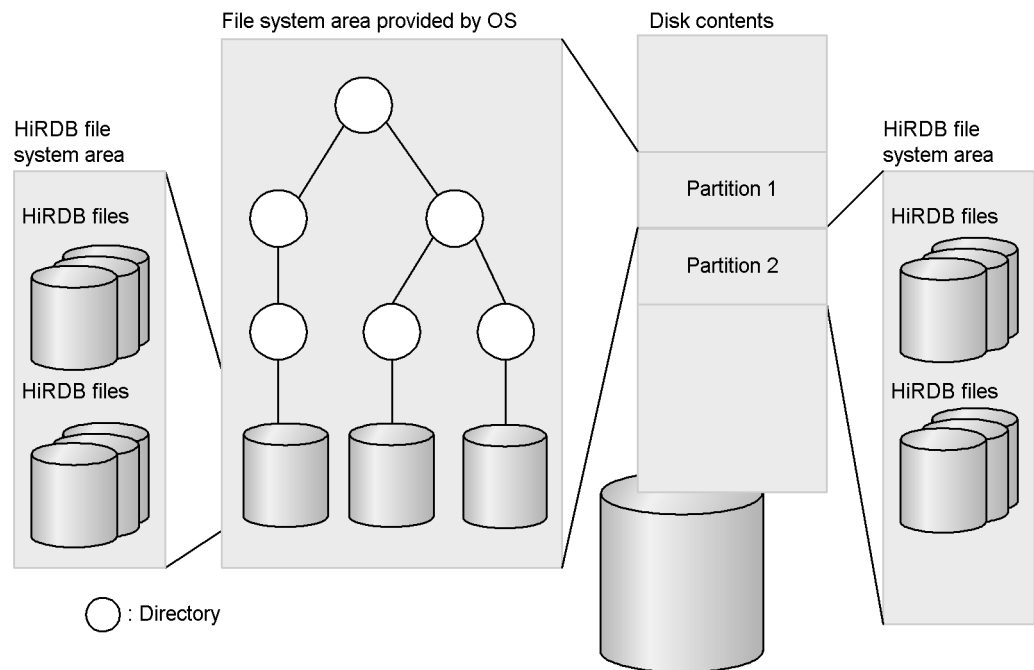
6.2 HiRDB file system areas

A special file in HiRDB that stores various types of HiRDB information, such as the information needed to restore the system status in the event of a table or index error, is called a *HiRDB file*. An area in which HiRDB files are created is called a *HiRDB file system area*. A HiRDB file system area must be provided before the special HiRDB files that constitute the system files and RDAREAs are created.

(1) Relationship between a HiRDB file system area and a file system area provided by the operating system

A disk used by the operating system for performing I/O operations is divided into contiguous areas called *partitions*. A partition can be used as a file system area provided by the operating system or as a HiRDB file system area. Figure 6-1 shows the relationship between HiRDB file system areas and file system areas provided by the OS.

Figure 6-1: Relationship between HiRDB file system areas and file system areas provided by the OS



(2) Files used for HiRDB file system areas

The HiRDB file system areas are created in Windows partitions.

(a) Normal Windows files

Create a HiRDB file system area by creating files in a normal partition provided by Windows. To do this, use the `pdfmkfs` command.

(b) Direct disk access (raw I/O)

You can use not only normal partitions but also Windows' direct disk access (raw I/O) to create HiRDB file system areas. The latter uses the raw I/O facility. The raw I/O facility lets you access partitions and logical drives in the same manner as files. However, some of the HiRDB file system areas cannot be created with raw I/O. To use the raw I/O facility, you need to provide unformatted partitions. In Windows, you can create partitions from the window displayed by choosing **Computer Management** and then **Disk Management**.

Note

- Do not use a partition that consists of multiple partitions (such as a volume set or a stripe set).
- You need to assign a drive letter to each partition.
- The raw I/O facility supports only fixed disks with a sector length of 512 bytes as drives.

(3) HiRDB file system area creation units

We recommend that you create a separate HiRDB file system area for each of the items listed and described in Table 6-2. For details about how to design the various HiRDB file system areas, see the *HiRDB Version 8 Installation and Design Guide*.

Table 6-2: Type of HiRDB file system area

Type of HiRDB file system area	Option ¹	Description
RDAREA	DB	HiRDB file system area in which RDAREAs (other than list RDAREAs) are stored. This area is always needed.
Shared RDAREA	SDB	HiRDB file system area in which shared RDAREAs are created. This area is needed if you use shared RDAREAs.
System file	SYS	HiRDB file system area in which system log files, synchronization point dump files, and status files are stored. This area is always needed.
Audit trail file		HiRDB file system area in which audit trail files are created. This area is needed to use the security audit facility.
Work table file ²	WORK	HiRDB file system area in which work table files are stored. This area is always needed.

Type of HiRDB file system area	Option ¹	Description
Utility	UTL	HiRDB file system area in which files used by utilities (backup files, unload data files, unload log files, index information files, or differential backup management files) are created.
	NUTL	HiRDB file system area in which files used by utilities are created. It differs from UTL in that the Windows cache is not used regardless of the value specified for the <code>pd_ntfs_cache_disable</code> operand.
List RDAREA ³	WORK	HiRDB file system area in which list RDAREAs are stored. This area is needed to perform narrowed searches.

¹ The value of the `-k` option specification that is specified when you create a HiRDB file system area with the `pdfmkfs` command.

² Raw I/O cannot be used on a HiRDB file system area for work table files. Create this area in a normal partition.

³ You cannot create a list RDAREA in a HiRDB file system area that uses raw I/O.

(4) Creating a HiRDB file system area

The `pdfmkfs` command is used to create a HiRDB file system area.

Reference note:

If you use either of the following environment setup support tools the first time you install HiRDB, a HiRDB file system area will be created based on the information you enter:

- Simple setup tool
- Batch file (`SPsetup.bat`)

For details about designing and creating HiRDB file system areas, see the *HiRDB Version 8 Installation and Design Guide*.

(5) Maximum size of HiRDB file system areas

Table 6-3 shows the maximum size of a HiRDB file system area.

Table 6-3: Maximum size of a HiRDB file system area

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

6.3 System files

A file that stores information that is needed in order to recover the system status after an error is called a system file. System files are referred to in general as follows:

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

6.3.1 System log files

A file in which system log information is stored is called a *system log file*. *System log* refers to information on the history of database updating, which is commonly called a log (or *journal* in mainframe terminology). HiRDB collects system log information in system log files for use:

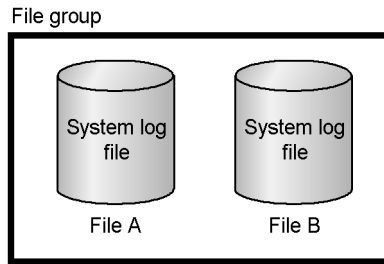
- By HiRDB for recovering the database after HiRDB or a UAP has terminated abnormally.
- By HiRDB administrators as information that is input when using the `pdrstr` command to recover a database.
- By the HiRDB administrator as the input information for acquiring statistical information.

The HiRDB administrator should create system log files as a safeguard against failures, as well as for acquiring statistical information.

(1) Organization of system log files

HiRDB operates with logical units of system log files called *file groups*. A file group consists of either one or two system log files. A configuration of two system log files is called a *duplexed system log file*. The respective system log files are distinguished by referring to them as *file A* and *file B*. When duplexed system log files are used, HiRDB acquires the same contents into both log files. Thus, if one of the files should fail, the other file can be used, thus maintaining system reliability. Figure 6-2 shows the organization of system log files.

Figure 6-2: Organization of system log files



(2) System log file creation

The `pdloginit` command is used to create system log files. Additionally, you specify the following HiRDB system definition operands to create an environment in which the system log files can be used:

- `pdlogadfg` operand (specifies the file group name of the system log files)
- `pdlogadpf` operand (specifies the system log file names that comprise the file group)

For details about designing and creating system log files, see the *HiRDB Version 8 Installation and Design Guide*. For details about using the system log files, see the *HiRDB Version 8 System Operation Guide*.

Reference note:

If you use either of the following environment setup support tools when you install HiRDB for the first time, system log files will be created based on the information you enter (the `pdlogadfg` and `pdlogadpf` operands will also be configured):

- Simple setup tool
- Batch file (`SPsetup.bat`)

6.3.2 Synchronization point dump files

If system logs are the only system files used to recover HiRDB when it has terminated abnormally, recovery processing must employ every system log from the point in time that HiRDB was last started. For this reason, recovering the system from system logs alone may take a significant amount of time. To mitigate this issue, points (called *synchronization points*) are established at fixed time intervals while HiRDB is running, and management information (*synchronization point dumps*) needed for recovery operations is saved at these points. This eliminates the need for system logs prior to the last synchronization point, which shortens the system recovery time.

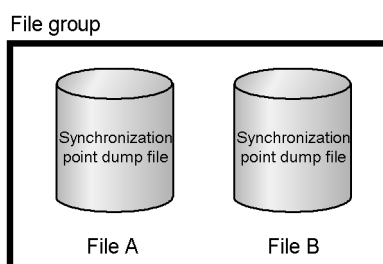
At each synchronization point, HiRDB incorporates into the database the details of all

the database updates that have been made since the last synchronization point or since HiRDB was started. The HiRDB administrator should create synchronization point dump files as a safeguard against failures.

(1) Organization of synchronization point dump files

HiRDB operates with logical units of synchronization point dump files called *file groups*. A file group consists of either one or two synchronization point dump files. A configuration of two synchronization point dump files is called a *duplexed synchronization point dump file*. The respective synchronization point dump files are distinguished by referring to them as *file A* and *file B*. When synchronization point dump files are duplexed, HiRDB collects and saves the same synchronization point dump information into both files. This enhances system reliability because, even if an error occurs in one of the files, the other file is still available. Figure 6-3 shows the organization of synchronization point dump files.

Figure 6-3: Organization of synchronization point dump files



(2) Number of guaranteed valid generations

If the most recent synchronization point dump file that has been generated cannot be read because an error has occurred in the file, HiRDB attempts to read the file that was created one generation earlier. If HiRDB cannot read the file that was created one generation earlier, it attempts to read the file that was created one generation previous to that one. In this way, HiRDB continues attempting to read previous generations of files if the read attempt fails.

The *number of guaranteed valid generations* refers to the number of previous generations of synchronization point dump files for which overwriting is prohibited. For example, if the number of guaranteed valid generations is 2, two generations of synchronization point dump files (the most recent generation and one generation previous to that one) cannot be overwritten. Thus, increasing the number of guaranteed valid generations enhances system reliability because, even if an error occurs in a synchronization point dump file, synchronization point dump files of the number set in the number of guaranteed valid generations will always be available.

Note that the number of synchronization point dump files you need is the number of guaranteed valid generations + 1.

(3) Synchronization point dump file creation

The `pdloginit` command is used to create synchronization point dump files. Additionally, you specify the following HiRDB system definition operands to create an environment in which the synchronization point dump files can be used:

- `pdlogadfg` operand (specifies the file group name of the synchronization point dump files)
- `pdlogadpf` operand (specifies the synchronization point dump file names that comprises the file group)

For details about designing and creating synchronization point dump files, see the *HiRDB Version 8 Installation and Design Guide*. For details about using synchronization point dump files, see the *HiRDB Version 8 System Operation Guide*.

Reference note:

If you use either of the following environment setup support tools when you install HiRDB for the first time, synchronization point dump files will be created based on the information you enter (the `pdlogadfg` and `pdlogadpf` operands will also be configured):

- Simple setup tool
- Batch file (`SPsetup.bat`)

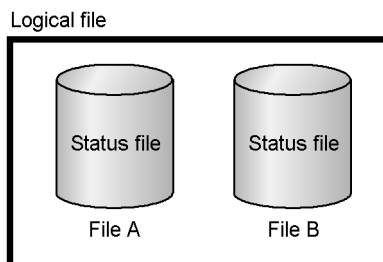
6.3.3 Status files

A file that stores system status information required by HiRDB for restarting the system is called a *status file*. The two types of status files are *unit status files* that store restart information for a unit and *server status files* that store restart information for a server. The HiRDB administrator should create status files for use in restarting HiRDB.

(1) Organization of status files

HiRDB operates with logical units of status files called *logical files*. A single logical file consists of two status files. Status files are duplexed in this manner, and the respective status files are distinguished by referring to them as *file A* and *file B*. HiRDB acquires the same system status information into both files, so that if one of the files fails, the other file can be used, thus maintaining system reliability. Figure 6-4 shows the organization of the status files.

Figure 6-4: Organization of status files



(2) Unit status file creation

The `pdstsinit` command is used to create unit status files. Additionally, you specify the `pd_syssts_file_name` operand of the unit control information definition to create an environment in which unit status files can be used.

The logical file name of the status files and the names of the status files that belong to the logical file are specified in the `pd_syssts_file_name` operand.

For details about designing and creating unit status files, see the *HiRDB Version 8 Installation and Design Guide*. For details about using unit status files, see the *HiRDB Version 8 System Operation Guide*.

Reference note:

If you use either of the following environment setup support tools when you install HiRDB for the first time, unit status files will be created based on the information you enter (the `pd_syssts_file_name` operand will also be configured):

- Simple setup tool
- Batch file (`SPsetup.bat`)

(3) Server status file creation

The `pdstsinit` command is used to create server status files. Additionally, you specify the `pd_sts_file_name` operand of the server definition to create an environment in which unit status files can be used. The logical file name of the status files and the names of the status files that belongs to the logical file are specified in the `pd_sts_file_name` operand.

For details about designing and creating server status files, see the *HiRDB Version 8 Installation and Design Guide*. For details about using server status files, see the *HiRDB Version 8 System Operation Guide*.

Reference note:

If you use either of the following environment setup support tools when you install HiRDB for the first time, server status files will be created based on the information you enter (the `pd_sts_file_name` operand will also be configured):

- Simple setup tool
- Batch file (`SPsetup.bat`)

6.3.4 System file components

Table 6-4 shows the components that comprise the system files.

Table 6-4: System file components

Type of system file		Creation unit	Number	Duplexed?
HiRDB/ Single Server	System log file	Needed for all units.	2 to 200 groups	O
	Synchronization point dump file		2 to 60 groups	O
	Server status file		1 to 7 files × number duplexed	Y
	Unit status file		1 to 7 files × number duplexed	Y
HiRDB/ Parallel Server	System log file	Needed on all servers other than the system manager.	2 to 200 groups per server	O
	Synchronization point dump file	Needed on all servers other than the system manager.	2 to 60 groups per server	O
	Server status file	Needed on all servers other than the system manager.	1 to 7 files × number duplexed per server	Y
	Unit status file	Needed for all units.	1 to 7 files × number duplexed per server	Y

Y: Always duplexed

O: Duplexing optional

(a) Configuration of system files in a HiRDB/Single Server

Figure 6-5 shows the configuration of the system files in a HiRDB/Single Server.

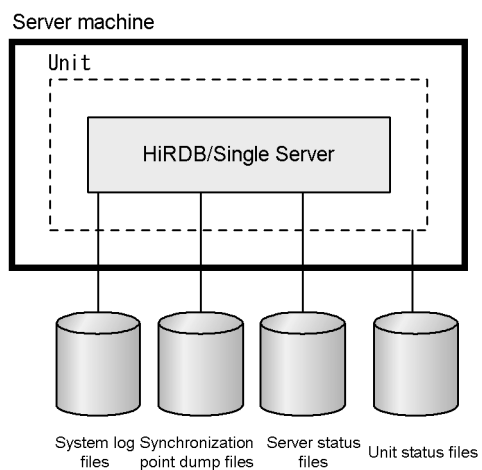
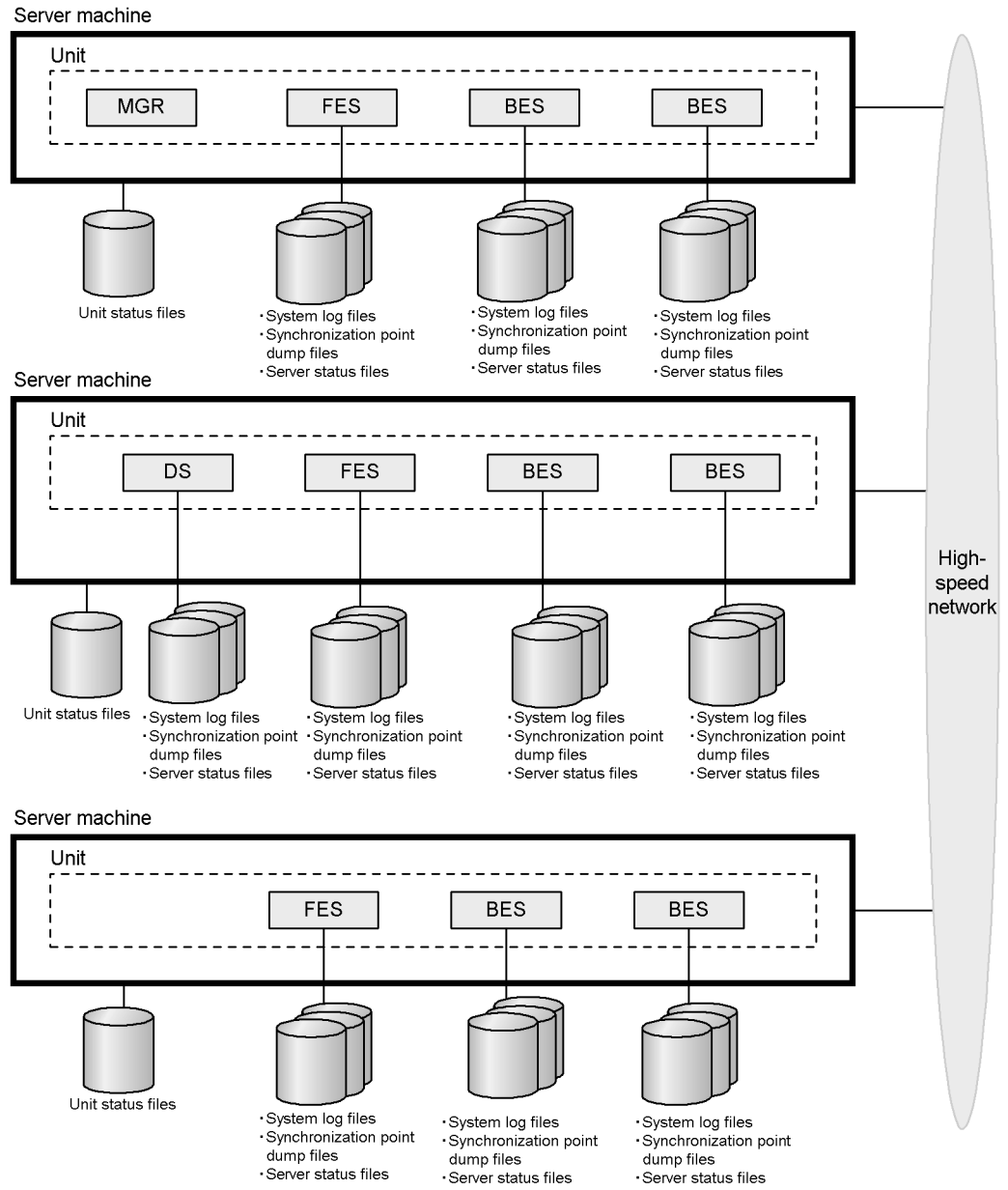
Figure 6-5: Configuration of system files in a HiRDB/Single Server**(b) Configuration of system files in a HiRDB/Parallel Server**

Figure 6-6 shows the configuration of the system files in a HiRDB/Parallel Server.

Figure 6-6: Configuration of system files in a HiRDB/Parallel Server



6.4 Work table files

A file that stores temporary information needed for executing an SQL is called a *work table file*. Work table files are created automatically by HiRDB. The HiRDB administrator must create HiRDB file system areas for the work table files.

(1) SQL statements and operations that require a work table file

(a) SQL statements that require a work table file

A work table file is used when multiple tables are joined by a `SELECT` statement for retrieval or during execution of certain SQL processing, such as `CREATE INDEX`.

The following types of SQL processing require work table files:

1. Retrieval specifying the `UNION [ALL]` or `EXCEPT [ALL]` clause
2. Specification in an `UPDATE` or `DELETE` statement of a retrieval condition based on a column for which an index is defined
3. `DROP SCHEMA`
4. `DROP TABLE`
5. `DROP INDEX`
6. Revocation with `REVOKE` statement of access privileges
7. `CREATE INDEX`
8. Creation with `ASSIGN LIST` statement of a list from a base table
9. Specification in a `SELECT` statement of any of the following:
 - Retrieval by joining multiple tables
 - Specification of `ORDER BY` clause for a column for which an index is not defined
 - Specification of `ORDER BY` clause for a row-partitioned table
 - Specification of a value expression containing a set function in a selection expression^{*}
 - Specification of a value expression containing the `COUNT (*) OVER ()` window function in a selection expression
 - Specification of `GROUP BY` clause
 - Specification of `DISTINCT` clause
 - Specification of a retrieval condition based on multiple columns for which an index is defined

- Specification of a retrieval condition for a column for which a repetition column index is defined
- Specification of the facility for batch acquisition from functions provided by plug-ins for the SQL optimization option, and specification of and searching for functions provided by plug-ins that use a plug-index for a retrieval condition.
- Specification of a retrieval condition based on a column for which an index is defined, and either a `FOR UPDATE` clause is specified or there is an update using this cursor
- Specification of a `FOR READ ONLY` clause
- Specification of a subquery of a quantified predicate
- Specification of a subquery of the `IN` predicate
- Creation of an internally derived table in a retrieval from a view table or a retrieval in which a `WITH` clause is specified

* Applies to a HiRDB/Parallel Server only; not applicable to a HiRDB/Single Server.

(b) Operations that require a work table file

The following operations require a work table file:

- Batch creation of an index
- Re-creation of an index
- Reorganization of an index
- Execution of the rebalancing utility

(2) Organization of work table files

Work table files are required at the following servers:

- Single servers
- Dictionary servers
- Back-end servers

(3) Creating a HiRDB file system area for work table files

The `pdfmkfs` command is used to create a HiRDB file system area for work table files. Additionally, you specify the `pdwork` operand of the server definition to create an environment in which a HiRDB file system area for work table files can be created. The name of the HiRDB file system area for work tables is specified in the `pdwork` operand.

For details (size, etc.) about designing and creating work table files, see the *HiRDB Version 8 Installation and Design Guide*.

Reference note:

If you use either of the following environment setup support tools when you install HiRDB for the first time, a HiRDB file system area for work table files will be created based on the information you enter (the `pdwork` operand will also be configured):

- Simple setup tool
- Batch file (`SPsetup.bat`)

6.5 HiRDB system definitions

The operating environment for HiRDB is set by specifications in the HiRDB system definitions. The files in which the HiRDB system definitions are stored are called *HiRDB system definition files*.

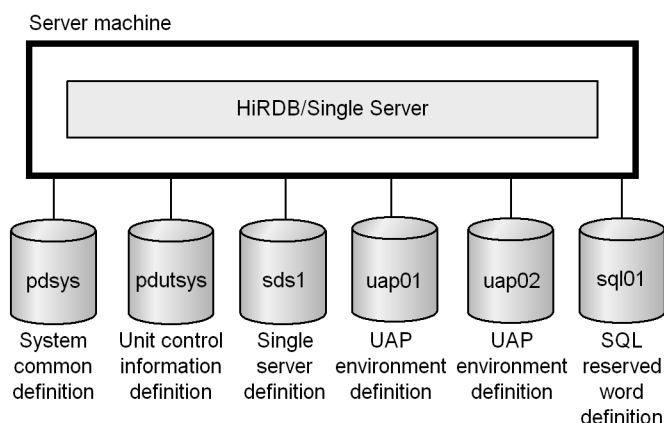
6.5.1 HiRDB system definitions for a HiRDB/Single Server

Table 6-5 lists the types of HiRDB system definitions and the files in which the HiRDB system definitions are stored. Figure 6-7 shows an example of a configuration of HiRDB system definition files.

Table 6-5: HiRDB system definitions (HiRDB/Single Server)

HiRDB system definition type	Storage file name	Description
System common definition	%PDDIR%\conf\pdsys	Defines the configuration of HiRDB and common information. This file is required for each HiRDB/Single Server.
Unit control information definition	%PDDIR%\conf\pduatsys	Defines unit control information. This file is required for each unit.
Server common definition	%PDDIR%\conf\pdsvrc	Defines default values for single server definition operands. This file is optional.
Single server definition	%PDDIR%\conf\ <i>server-name</i>	Defines the execution environment for a single server. This file is required for each single server.
UAP environment definition	%PDDIR%\conf\pduapenv\ <i>any-name</i>	Defines the UAP execution environment. This file is optional. You can create a maximum of 4096 UAP environment definitions.
SQL reserved word definition	%PDDIR%\conf\pdrsvwd\ <i>any-name</i>	Defines SQL reserved words. This file is required to use the SQL reserved word deletion facility. For details about the SQL reserved word deletion facility, see the <i>HiRDB Version 8 UAP Development Guide</i> .

Figure 6-7: Example configuration of HiRDB system definition files (HiRDB/Single Server)



6.5.2 HiRDB system definitions for a HiRDB/Parallel Server

Table 6-6 lists the organization of HiRDB system definitions. Figure 6-8 shows a configuration example of HiRDB system definition files. Figure 6-9 shows a configuration example of HiRDB system definition files when the HiRDB External Data Access facility is used.

Table 6-6: HiRDB system definitions (HiRDB/Parallel Server)

HiRDB system definition type	Storage file name	Description
System common definition	%PDDIR%\conf\pdsys	Defines the configuration of HiRDB and common information. This file is required for each unit. All units must have the same system common definition.
Unit control information definition	%PDDIR%\conf\pdutsys	Defines unit control information. This file is required for each unit.
Server common definition	%PDDIR%\conf\pdsvrc	Defines default values for individual server definition operands. This file is optional.
Front-end server definition	%PDDIR%\conf\server-name	Defines the execution environment for a front-end server. This file is required for each front-end server.
Dictionary server definition	%PDDIR%\conf\server-name	Defines the execution environment for a dictionary server. This file is required for each dictionary server.

HiRDB system definition type	Storage file name	Description
Back-end server definition	%PDDIR%\conf\ <i>server-name</i>	Defines the execution environment for a back-end server. This file is required for each backend server.
UAP environment definition	%PDDIR%\conf\pduapenv\ <i>any-name</i>	Defines the execution environment for UAPs. This file is optional. You must create UAP environment definitions in a unit that contains a front-end server. If you have multiple front-end servers, define these definitions on the front-end server to which you wish to apply the UAP environment definitions. You can create a maximum of 4096 UAP environment definitions.
SQL reserved word definition	%PDDIR%\conf\pdrsvwd\ <i>any-name</i>	Defines SQL reserved words. This file is required to use the SQL reserved word deletion facility. For details about the SQL reserved word deletion facility, see the <i>HiRDB Version 8 UAP Development Guide</i> .
Foreign server information definition	%PDDIR%\conf\ <i>foreign-server-name</i>	Defines the connection environment for foreign servers. You must define these definitions in a unit that contains a back-end server for connecting to foreign servers. This file is needed when the HiRDB External Data Access facility is being used.
Hub optimization information definition	%PDDIR%\conf\ <i>any-name</i>	Defines optimization information for the HiRDB External Data Access facility. This file is needed when the HiRDB External Data Access facility is being used. Identical hub optimization information definitions must be defined for each unit.

Figure 6-8: Example configuration of HiRDB system definition files (HiRDB/Parallel Server)

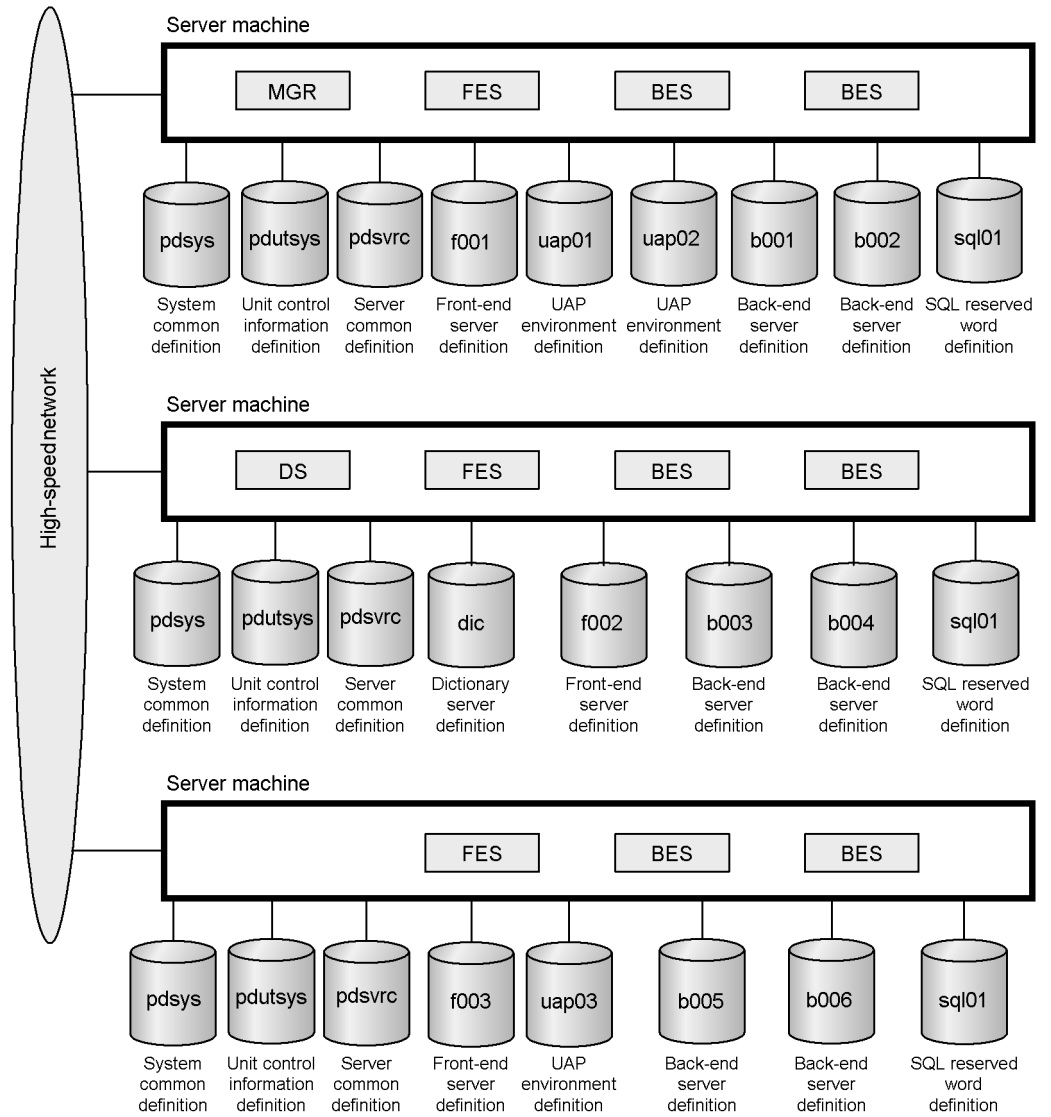
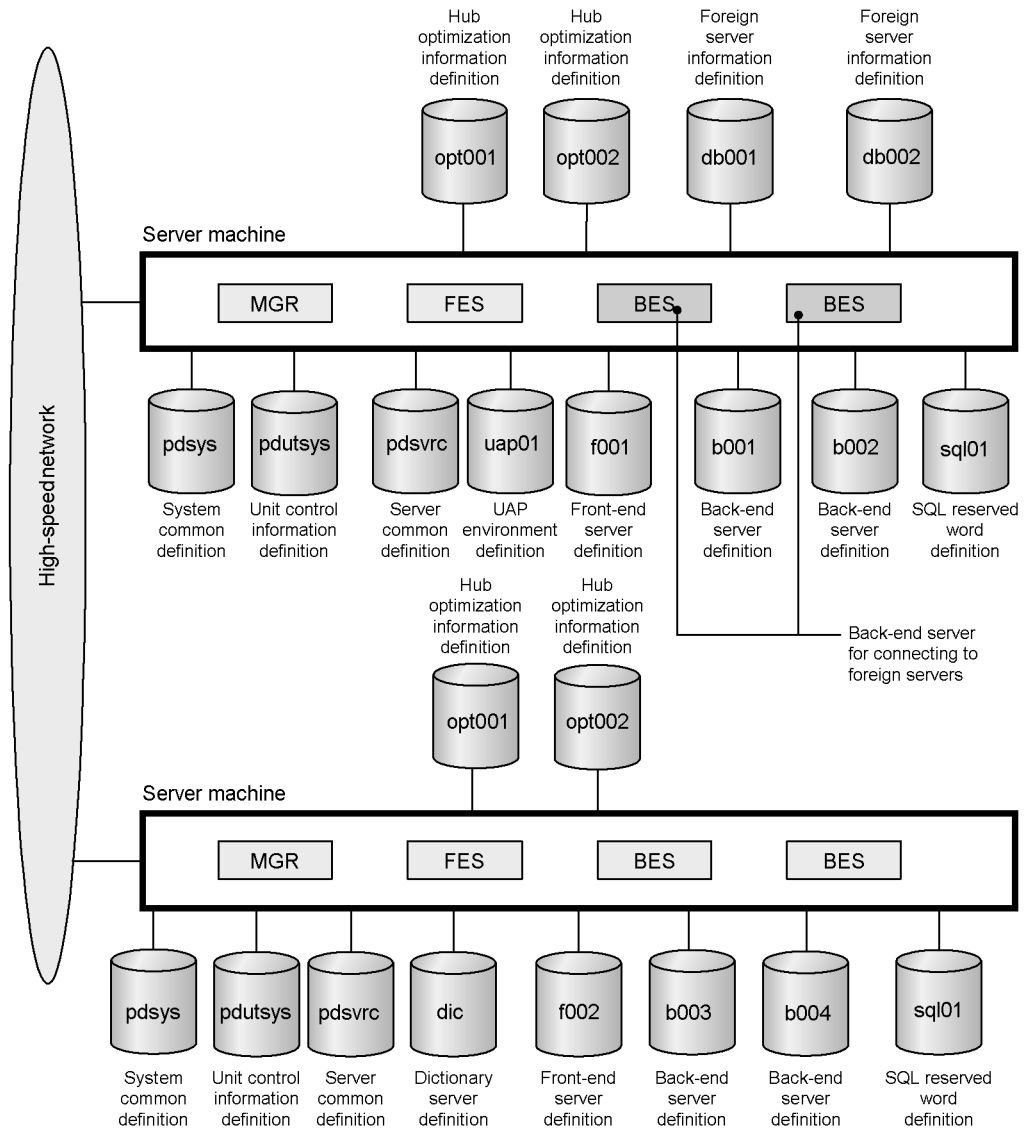


Figure 6-9: Example configuration of HiRDB system definition files (when the HiRDB External Data Access facility is being used)



Explanation

- A foreign server information definition file is created in a unit that contains a back-end server for connecting to foreign servers.
- Because this is a multiple front-end server environment, a hub optimization

information definition file is created for each unit.

6.5.3 HiRDB system definition file creation

During system building, the HiRDB administrator uses one of the following methods to create HiRDB system definition files:

Batch file (SPsetup.bat)

Executing the batch file creates HiRDB system definition files automatically under `%PDDIR%\conf`.

Windows editor

HiRDB system definition files can be created under `%PDDIR%\conf` with a Windows editor such as Notepad and specification of required operands in the HiRDB system definitions.

For details about creating HiRDB system definition files, see the *HiRDB Version 8 Installation and Design Guide*. For details about the operands of the HiRDB system definitions, see the *HiRDB Version 8 System Definition*.

After creation of a HiRDB system definition

You can use the `pdconfchk` command to check the operands of a HiRDB system definition. This command checks the integrity of the specifications of the definition operands that are required to start HiRDB. We recommend that you execute the `pdconfchk` command after a HiRDB system definition is created (especially when a Windows editor such as Notepad was used to create the HiRDB system definition).

6.5.4 System reconfiguration command (pdchgconf command)

With the exception of UAP environment definitions, normally you must stop HiRDB before you can change any HiRDB system definitions. By using the system reconfiguration command, you can change HiRDB system definitions while HiRDB is running. This allows you to perform the following operations while HiRDB is running:

- Adding, removing, and moving units
- Adding, removing, and moving servers
- Adding system files
- Adding, removing, and modifying system buffers

The system reconfiguration command is extremely useful for systems that are running continuously 24 hours a day. For details about how to use the system reconfiguration command, see the *HiRDB Version 8 System Operation Guide*.

Note that you must have HiRDB Advanced High Availability to use the system reconfiguration command.

6.6 HiRDB startup and termination

This section explains HiRDB startup and termination. To start HiRDB, you execute the `pdstart` command; to terminate HiRDB, you execute the `pdstop` command. In addition, there is a method (called *automatic startup*) that enables HiRDB to start automatically when the operating system starts, as well as a method for HiRDB/Parallel Servers (called *reduced activation*) that enables operable units to start, even if there are units that cannot start.

6.6.1 Startup and termination modes

Starting HiRDB is governed by the *startup mode*; closing HiRDB is governed by the *termination mode*. This section explains the startup and termination modes. For details about the HiRDB startup and termination procedures, see the *HiRDB Version 8 System Operation Guide*.

(1) Startup modes

Table 6-7 shows the HiRDB startup modes.

Table 6-7: HiRDB startup modes

Startup mode	Explanation	Input command		
		Startup on a system basis	Startup on a unit basis ¹	Startup on a server basis ¹
Normal startup	This startup mode is used when the previous termination mode was normal termination. Information from the previous operation is not inherited.	pdstart	pdstart -u pdstart -x	pdstart -s
Restart	This startup mode is used when the previous termination mode was one of those listed as follows; information from the previous operation is inherited: <ul style="list-style-type: none"> • Planned termination • Abnormal termination • Forced termination 			N

Startup mode	Explanation	Input command		
		Startup on a system basis	Startup on a unit basis ¹	Startup on a server basis ¹
Forced startup ²	This startup mode is used to start HiRDB forcibly when it cannot be restarted. Information from the previous operation is not inherited during this process, so the database cannot be recovered; the HiRDB administrator must recover the database.	pdstart dbdestroy	pdstart -u dbdestroy pdstart -x dbdestroy	N

N: Not usable (this startup method cannot be executed).

¹ In the case of a HiRDB/Parallel Server, startup and termination can be executed on a unit basis or on a server basis.

² When HiRDB is started forcibly, all RDAREAs (including system RDAREAs) that were updated since the last time HiRDB was started will have been destroyed. Thus, when forced startup is used, the destroyed RDAREAs must be recovered with the database recovery utility (`pdrstr`). If the RDAREAs are not recovered, correct operation of HiRDB cannot be guaranteed. These RDAREAs can be recovered using the system log only. See the `KFPS01262-I` message that was output the previous time the `pdstart` command was used, and use as the input information to the database recover utility (`pdrstr`) the log file group name that was used at that time, which is shown in the message, as well as the system log that has been generated since then.

(2) Termination modes

Table 6-8 shows the HiRDB termination modes.

Table 6-8: HiRDB termination modes

Termination mode	Explanation	Input command		
		Termination on a system basis	Termination on a unit basis	Termination on a server basis
Normal termination	Prohibits CONNECT requests, and terminates HiRDB after all user processes are finished. If you cannot stop HiRDB by executing the <code>pdstop</code> command because a utility is active, the <code>KFPS05074-E</code> message is output. If a unit cannot be stopped, the <code>KFPS05070-E</code> message is output. In these cases, the <code>pdstop</code> command ends with return code 8.	<code>pdstop</code>	<code>pdstop -u</code> <code>pdstop -x</code>	<code>pdstop -s</code>
Planned termination	Prohibits acceptance of transactions; terminates HiRDB after all users, including utilities, have disconnected.	<code>pdstop -P</code>	N	N
Forced termination	Terminates HiRDB immediately without waiting for completion of transactions being processed (these transactions become rollback targets ¹ during a restart).	<code>pdstop -f</code>	<code>pdstop -f -u</code> <code>pdstop -f -x</code>	N
Abnormal termination	Termination mode in which HiRDB is terminated by an error. HiRDB terminates immediately without waiting for completion of transactions being processed (these transactions become rollback targets ¹ during a restart).	NA	NA	NA

NA: Not applicable.

N: Not usable (this termination method cannot be executed).

¹ Transactions that are being processed become rollback targets during a restart, except in the following cases:

- Database load utility (`pdload`) or database reorganization utility (`pdreorg`) is being executed in the no-log mode
- UAP is being executed in the no-log mode

After HiRDB has been restarted, the HiRDB administrator must either recover the RDAREAs from backup copies or re-execute the utility. For details about the no-log mode, see the *HiRDB Version 8 System Operation Guide*. For details about the procedure for recovering RDAREAs when the no-log mode is being used, see the *HiRDB Version 8 System Operation Guide*.

6.6.2 HiRDB automatic startup

HiRDB *automatic startup* means that HiRDB starts automatically when OS starts. *Manual startup* means that the `pdstart` command is entered after OS has started to start HiRDB. Which startup method is to be used is specified in the `pd_mode_conf` operand of the system common definition.

When automatic startup is specified, HiRDB (the unit) restarts automatically, even after HiRDB (the unit) has terminated abnormally. However, if three restart attempts in a row result in abnormal termination, there are no more attempts to restart the system automatically.

Reference note:

- If HiRDB terminates abnormally during its startup or termination processing, the next startup will have to be by manual startup.
- A startup mode (forced startup, unit-basis startup, server-basis startup) that involves specification of `pdstart` command arguments (options) cannot be specified when startup is by automatic startup.

6.6.3 Reduced activation (applicable to HiRDB/Parallel Server only)

Normally, a HiRDB/Parallel Server cannot be started if any of its units will not start. If an event such as an error prevents a unit from starting, the error must be resolved before HiRDB can be started. However, the reduced activation facility can be implemented so that HiRDB can be started using only the normal units; this is called *reduced activation*. To enable reduced activation, specify the following operands:

- `pd_start_level`
- `pd_start_skip_unit`

For details about using reduced activation, see the *HiRDB Version 8 System Operation Guide*.

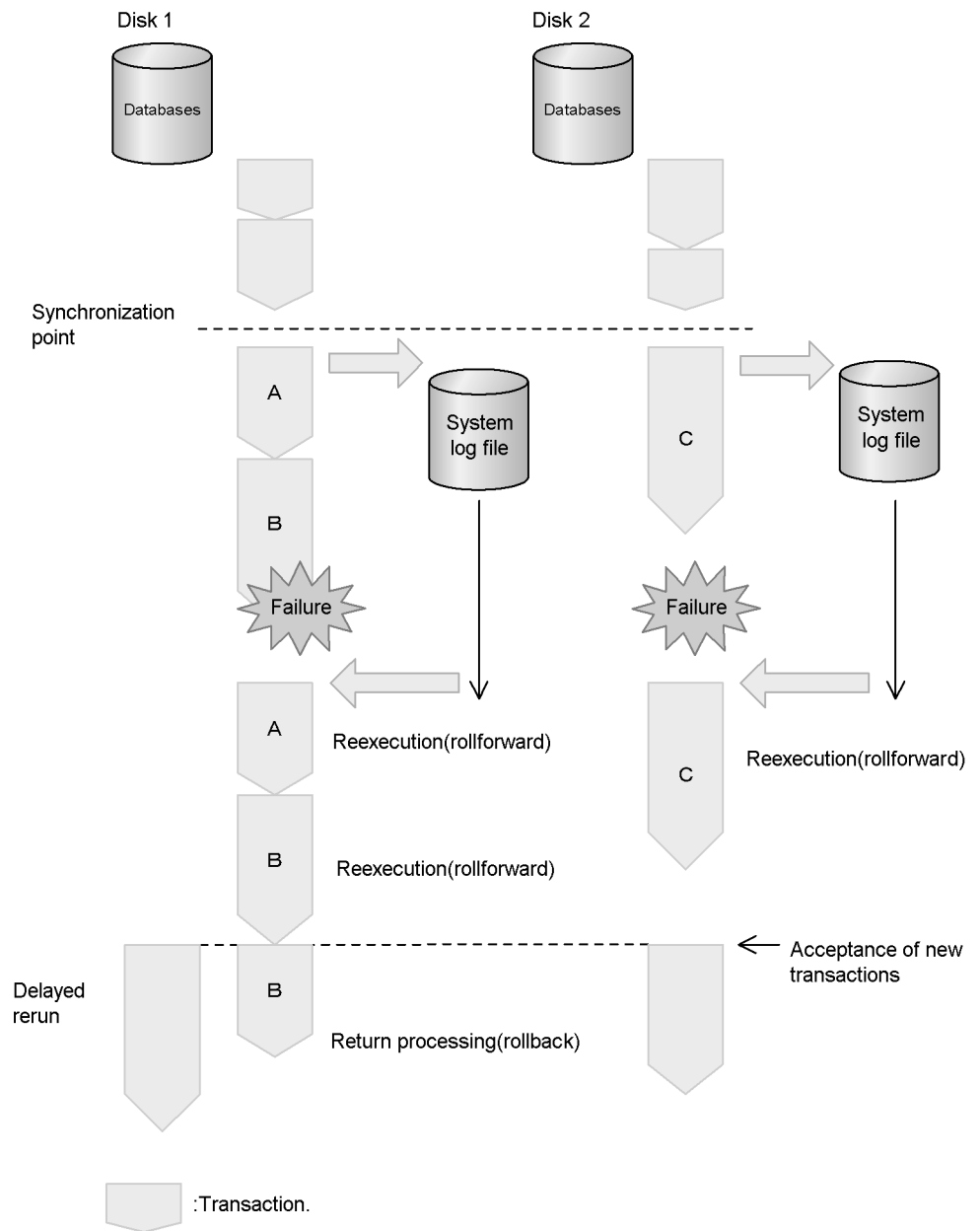
6.7 Delayed rerun

In HiRDB, a recovery process called *delayed rerun* reduces the downtime for transaction processing when a system failure occurs. When a failure occurs in the database, HiRDB uses the system log file to re-execute all update processing that occurred since the most recent synchronization point up to the time the failure occurred. During this processing, the transaction that was updating the database when the failure occurred is re-executed first (*rollforward*) to the point at which the failure occurred, and then the database is returned (*rollback*) to its status before the update processing was executed.

HiRDB executes rollback for the data on the disk that was being updated when the failure occurred and accepts new transactions for data that is not subject to the rollback. In this way, the system downtime for transaction processing is minimized.

Figure 6-10 shows the concept of delayed rerun.

Figure 6-10: Concept of delayed rerun



Explanation

A failure occurs during transaction B that accesses the database on disk 1; this

failure occurs after completion of transaction C for disk 2. Consequently, transactions A, B, and C that executed since the previous synchronization point are rolled forward. Subsequently, transaction B, which was being executed when the failure occurred, is rolled back. Because the transactions for the database on disk 2 are not the targets of rollback, new transactions are being accepted.

6.8 Database access processing method

HiRDB uses global buffers to manage database input/output processes. This section explains the HiRDB database access processing method that uses global buffers. HiRDB provides the following facilities to improve the performance of the database access processing method:

- Global buffers
- Prefetch facility
- Asynchronous READ facility
- Deferred write processing
- Facility for parallel writes in deferred write processing
- Incorporation during commit
- LRU management of global buffers
- Page access using the snapshot method
- Local buffers
- Global buffer pre-writing
- BLOB data file I/O facility
- BLOB and BINARY data addition update and partial extraction facility

6.8.1 Global buffers

A global buffer is a buffer that is used for input and output of data stored in the RDAREAs of a disk. Global buffers are allocated in the common memory. Buffers used for storing data for updating before the data in the database has been updated are called *update buffers*. Buffers that are used for referencing data or for storing data that has been updated in the database are called *reference buffers*.

Global buffers are always allocated for RDAREAs and indexes that store data. Table 6-9 lists the types of global buffers:

Table 6-9: Types of global buffers

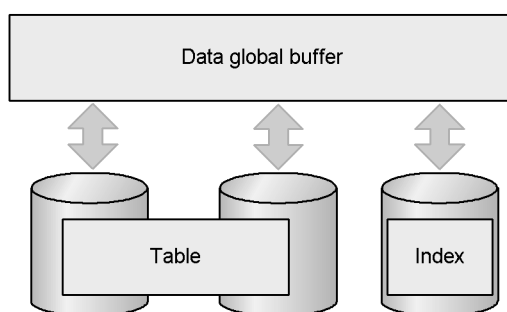
Global buffer type	Explanation
Data global buffers	Global buffers used for input/output operations on table data. Data global buffers are allocated in units of RDAREAs.
Index global buffers	Global buffers used for input/output operations on index data. Index global buffers are allocated in units of indexes.

Global buffer type	Explanation
LOB global buffers	Global buffers used for input/output operations on LOB-attribute data. LOB global buffers are allocated in units of LOB RDAREAs.

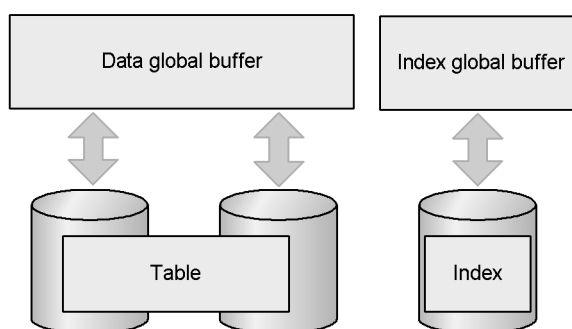
You can improve performance even more by using these global buffers in combination. For example, when you separately define data global buffers and index global buffers, data and index searches each operate independently, even if they are running at the same time. Consequently, the number of index inputs/outputs can be reduced even for a full-text search of a large amount of data, resulting in reduced processing time. Figure 6-11 shows the concept of global buffers.

Figure 6-11: Concept of global buffers

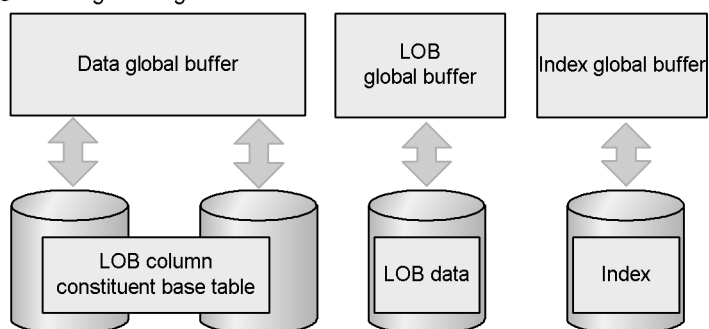
- Allocating the same global buffer for table data and index data



- Allocating separate global buffers for table data and index data



- Allocating a LOB global buffer



(1) Units for allocating global buffers

- Each RDAREA must have a data global buffer assigned to it. Multiple RDAREAs can be assigned to the same global buffer.
- Index global buffers can be allocated to indexes as necessary.

- LOB global buffers can be allocated to LOB RDAREAs as necessary.
- Global buffers can be allocated to system RDAREAs as necessary.
- For details about designing global buffers (and allocating global buffers to RDAREAs), see the *HiRDB Version 8 Installation and Design Guide*.

(2) Global buffer allocation procedures

You can allocate global buffers by specifying the `pdbuffer` operand. The following examples show the use of the `pdbuffer` operand to allocate global buffers:

Examples

```
pdbuffer -a gbuf01 -r RDAREA01,RDAREA02 -n 1000      1
pdbuffer -a gbuf01 -r LOBAREA01 -n 1000             2
pdbuffer -a gbuf01 -i USER01.INDX01 -n 1000        3
pdbuffer -a gbuf01 -b LOBAREA02 -n 1000            4
```

Explanation

1. Allocates a data global buffer to two RDAREAs (RDAREA01 and RDAREA02).
2. Allocates a data global buffer to a LOB RDAREA (LOBAREA01).
3. Allocates an index global buffer to an index (USER01.INDX01).
4. Allocates a LOB global buffer to a LOB RDAREA (LOBAREA01).

For details about the `pdbuffer` operand, see the *HiRDB Version 8 System Definition*.

(3) Dynamic updating of global buffers

Because you must modify the `pdbuffer` operand to add, change, or delete global buffers, you normally need to stop HiRDB to perform this operation. However, if you install HiRDB Advanced High Availability, you can use the `pdbufmod` command to add, change, and delete global buffers while HiRDB is running. This is called *dynamic updating of global buffers*. As use examples, we recommend that you perform dynamic updating of global buffers in the following cases:

- Allocating global buffers for an RDAREA that was added
- Changing a global buffer allocated for an RDAREA
- Changing a global buffer definition as a result of global buffer tuning

For details about dynamic updating of global buffers, see the *HiRDB Version 8 System Operation Guide*.

6.8.2 Prefetch facility

Table data on disk normally is read from a global buffer (or a local buffer) one page at a time. However, this data can also be read in batches of multiple pages, rather than

one page at a time. This capability is provided by the *prefetch facility*. By using the prefetch facility, you can reduce the number of I/O operations between the table data on disk and the global buffer (or local buffer).

(1) *Application criteria for the prefetch facility*

The prefetch facility is available when the following conditions are satisfied:

1. The accessed data is in a HiRDB file system area for which the raw I/O facility is used.
2. A large data set is being manipulated.
3. One of the following SQL statements is being executed:
 - A `SELECT`, `UPDATE`, or `DELETE` statement that does not use an index
 - A `SELECT`, `UPDATE`, or `DELETE` statement that performs an ascending search* using an index or cluster key (except for `=` and `IN` conditions)

* For multi-column indexes, searching is performed in the order specified by the index definitions.

(2) *Specifying the prefetch facility*

For a global buffer

The prefetch facility is enabled by specifying a value of at least 1 for the `-m` option of the `pdbuffer` operand. The specification in the `-p` option of the `pdbuffer` operand is of the number of pages to be fetched at a time. For details about the `pdbuffer` operand, see the *HiRDB Version 8 System Definition*.

For a local buffer

The prefetch facility is enabled by specifying the number of pages to be fetched at a time in the `-p` option of the `pdlbuffer` operand. For details about the `pdbuffer` operand, see the *HiRDB Version 8 System Definition*.

6.8.3 Asynchronous READ facility

When you are using the prefetch facility to input multiple pages at a time into a global buffer, pages are synchronously pre-fetched from the DB process and input to the prefetch buffer. When the prefetch facility is being used, the *asynchronous READ facility* sets up two prefetch buffers and, while the DB process is using one of the buffers, a READ process pre-fetches pages from the other buffer asynchronously with the DB process. By executing the DB processing and the prefetch input at the same time, you can reduce the processing time. Moreover, with HiRDB/Parallel Server, by alternating the threads for the I/O wait time, you can reduce the I/O wait time.

Note, however, that you cannot use the asynchronous READ facility with a local buffer. Nor can you use it on an RDAREA that has a `SCHEDULE` attribute. Use the

prefetch facility in these cases.

Enabling the asynchronous READ facility

You specify the number of asynchronous READ processes using the `pd_max_ard_process` operand. If you specify 0 (or do not specify anything) in this operand, the asynchronous READ facility remains inactive. If it is active, you must also specify the prefetch facility (by specifying a value of 1 or greater in the `-m` option of the `pdbuffer` operand).

6.8.4 Deferred write processing

Deferred write processing is a process that writes pages that have been updated in the global buffer onto the disk whenever a particular number of pages has been updated (rather than when a `COMMIT` statement is issued). The point in time at which the number of updated pages has reached the particular value (which is determined by HiRDB) is called the *deferred write trigger*.

The number of updated pages to be written onto the disk is determined by HiRDB on the basis on the updated output page rate at the deferred write trigger that is specified in the `-w` option of the `pdbuffer` operand. When deferred write processing is in effect, data is not written onto the disk even when a `COMMIT` statement is issued, thus reducing the input/output processing load.

For details about using deferred write processing, see the *HiRDB Version 8 Installation and Design Guide*.

Setting up deferred write processing

Deferred write processing is used by specifying both the `pd_dbsync_point` and the `pdbuffer` operands of the system common definition. For details about the `pd_dbsync_point` and `pdbuffer` operands, see the *HiRDB Version 8 System Definition*.

6.8.5 Facility for parallel writes in deferred write processing

The facility for parallel writes in deferred write processing is a function that enables multiple processes to execute deferred write processing. Because multiple processes perform write processing, the facility reduces the amount of the time required to write to disk. For details about the facility, see the *HiRDB Version 8 Installation and Design Guide*.

Specifying the facility for parallel writes in deferred write processing

To use the facility for parallel writes in deferred write processing, specify in the `pd_dfw_awt_process` operand the number of processes that can perform write processing, and specify in the `pd_dbbuff_rate_updpage` operand the request rate of the deferred write trigger. If the `pd_dfw_awt_process` operand is not specified, the facility for parallel writes in deferred write processing does not become effective.

Considerations about using the facility

Use of the facility for parallel writes in deferred write processing increases CPU usage, because of the increase in the number of processes.

6.8.6 Incorporation during commit

Normally, pages that have been updated in the global buffer are written onto the disk whenever a `COMMIT` statement is issued. This is called *incorporation during commit*. When incorporation during commit is in effect, there is no need to recover the database from a synchronization point during full recovery processing of the system, which reduces the amount of time required for full recovery processing. For details about using incorporation during commit, see the *HiRDB Version 8 Installation and Design Guide*.

Setting up incorporation during commit

Incorporation during commit is used by specifying `commit` in the `pd_dbsync_point` operand. For details about the `pd_dbsync_point` operand, see the *HiRDB Version 8 System Definition*.

6.8.7 LRU management method for global buffers

The LRU management method for global buffers that is appropriate to the types of job (online or batch) can be selected. The two LRU management methods are described as follows.

(1) Management of reference buffers and update buffers in independent LRUs

This method manages reference buffers and update buffers in independent LRUs. If a shortage of global buffers occurs, the reference buffer in the global buffer that was accessed least recently is purged from the memory. If the number of references and updates per transaction is relatively small, as in the case of online jobs, managing reference buffers and update buffers in independent LRUs improves processing performance.

To apply LRU management to global buffers, `SEPARATE` (default value) must be specified in the `pd_dbbuff_lru_option` operand. For details about the `pd_dbbuff_lru_option` operand, see the *HiRDB Version 8 System Definition*.

(2) Management of global buffers in a single LRU

This method manages all global buffers in a single LRU. If a shortage of global buffers occurs, the buffer in the global buffer pool that was accessed least recently is purged from the memory. If a large number of retrievals and updates coexist, as in a combination of online jobs and batch jobs, managing the reference buffers in a single LRU improves processing performance.

One of the following actions must be taken to manage global buffers in a single LRU:

- Specify `MIX` in the `pd_dbbuff_lru_option` operand.

- Specify the updated output page rate at the deferred write trigger in the `-w` option of the `pdbuffer` operand.

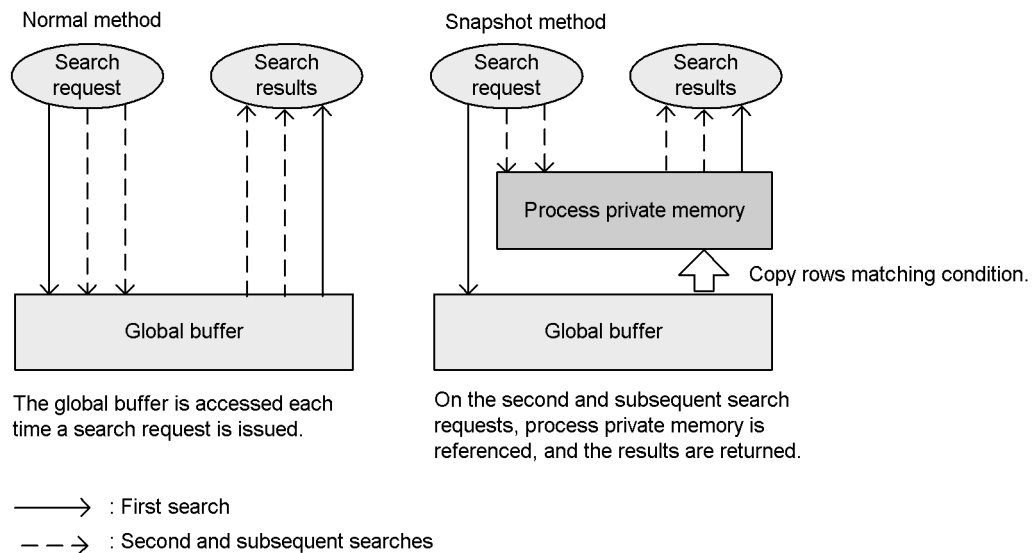
For details about the `pd_dbbuff_lru_option` and `pdbuffer` operands, see the *HiRDB Version 8 System Definition*.

6.8.8 Accessing pages using the snapshot method

When a search that cannot use a performance-enhancing facility (such as the rapid grouping facility) is being performed, global buffers are accessed roughly the same number of times as the number of rows there are that match the condition. With the *snapshot method*, the first time data is accessed, all rows in the buffer that match the search condition are copied to process private memory. The second and subsequent times that the same page is accessed, process private memory is referenced, and the search results are returned. This enables search times to be reduced after the first search. It also reduces the number of times that a global buffer is accessed, and avoids concentrating accesses to the same buffer.

Figure 6-12 provides an overview of the snapshot method.

Figure 6-12: Overview of the snapshot method



Specification method

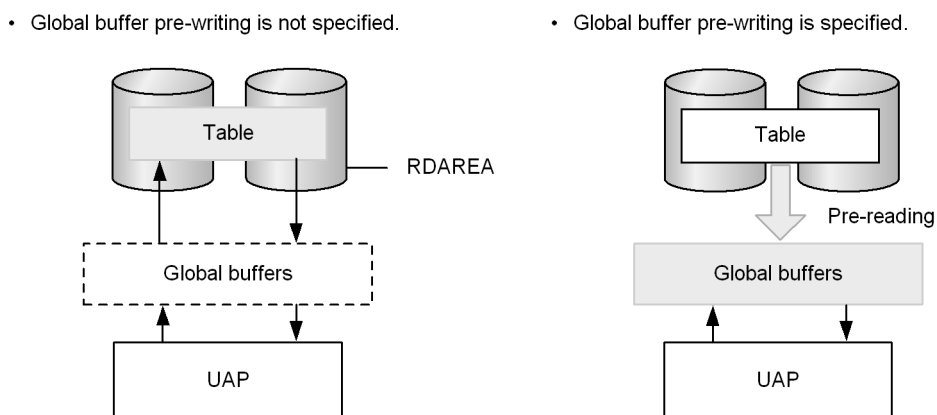
Specify `SNAPSHOT` (default value) in the `pd_pageaccess_mode` operand.

6.8.9 Global buffer pre-writing

Global buffer pre-writing is a function that loads in advance data from a specified table or index into global buffers. Figure 6-13 provides an overview of global buffer

pre-writing.

Figure 6-13: Overview of global buffer pre-writing



Explanation

- Global buffer pre-writing is not specified

When a UAP accesses HiRDB immediately after HiRDB starts, data must be read from the table (a physical I/O occurs) because the global buffers do not hold any data. When data in this table is accessed later, pages from this table that have been written into global buffers are not read from the table. However, accesses to data on other pages must still be read from the table.

- Global buffer pre-writing is specified

Table data is written into global buffers in advance, so the table can be accessed without data having to be read from the table (no physical I/O occurs). Subsequent accesses to this table do not require that data be read from the table.

(1) Advantages of global buffer pre-writing

Because reading of table and index data from specified tables is performed in advance, the global buffer hit ratio is improved. By specifying global buffer pre-writing before online operations or other activities are started of tables or indexes that you believe will be accessed frequently, you can expect to achieve a higher buffer hit ratio.

(2) Execution method

To enable global buffer pre-writing, specify the tables or indexes you wish to have pre-written into the global buffers, and execute the global buffer residence utility (`pdpgbfn`).

(3) Considerations when using

You should consider the following points about using global buffer pre-writing:

- You must have more global buffers than the number of pages stored in the tables or indexes that are to be pre-read.
- If you do not have enough global buffers, older page information will be forced out of the global buffers according to the LRU management method (the oldest page in the accessed global buffers is forced out as specified by the value in the `pd_dbbuff_lru_option` system definition operand). This means that executing `pdpgbfon` is pointless if you do not have enough global buffers.
- When you specify pre-writing with the global buffer residence utility (`pdpgbfon`), the pre-fetch facility becomes available. This makes it possible to reduce execution time by specifying the prefetch facility when you define global buffers.

6.8.10 Local buffers

A local buffer is a buffer that is used for input and output of data stored in RDAREAs on disk, and is allocated in process private memory. Table 6-10 lists and describes the types of local buffers available.

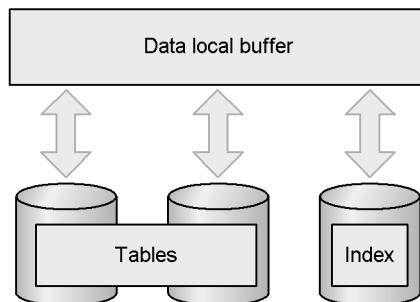
Table 6-10: Types of local buffers

Local buffer type	Description
Data local buffer	Local buffers used for table data I/O. Data local buffers are allocated on a per-RDAREA basis.
Index local buffer	Local buffers used for index data I/O. Index local buffers are allocated on a per-index basis.

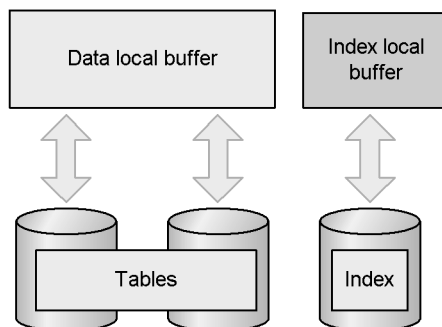
You can use these local buffers to improve performance. For example, by separately defining data local buffers and index local buffers, data and index searches each operate independently, even if they are running at the same time. This enables the number of index I/O operations to be reduced even for a full text search of a large amount of data, resulting in reduced processing time. Figure 6-14 provides an overview of local buffers.

Figure 6-14: Overview of local buffers

- When table data and index data are allocated to the same local buffer



- When table data and index data are allocated to different local buffers



(1) Local buffer application criteria

Defining local buffers is beneficial if all of the following conditions are satisfied:

- A large amount of data is being searched or updated.
- The RDAREA to be accessed cannot be accessed from another UAP.

However, do not define local buffers for a UAP that establishes an emergency connection to HiRDB, due to the significant adverse effects it has on the system (memory resource strain, process monopolization, and so on.).

(2) Local buffer allocation procedure

Use the `pdlbuffer` operand to allocate local buffers. The following example shows how to use the `pdlbuffer` operand to allocate local buffers:

Example:

```
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 an index (USER01 . INDX01).

For details about the `pdlbuffer` operand, see the *HiRDB Version 8 System Definition*.

6.8.11 BLOB data file output facility

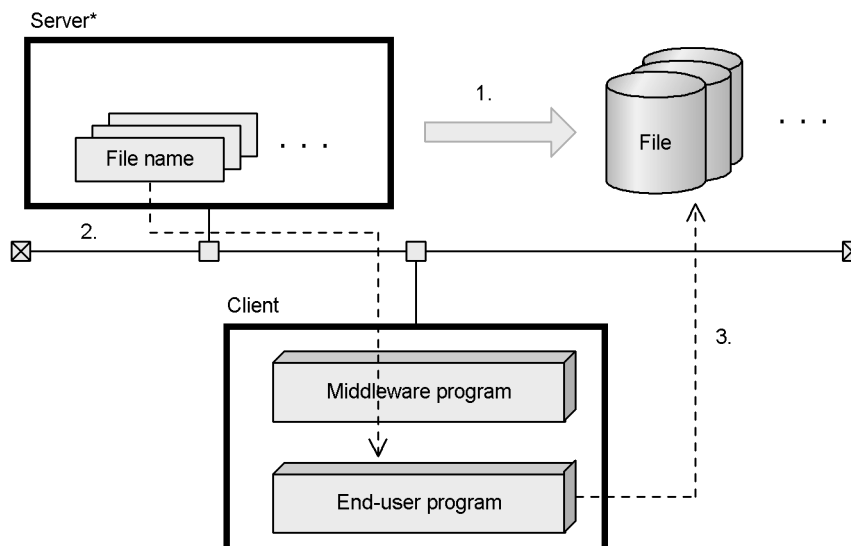
The following issues arise if BLOB data stored in a HiRDB server is to be retrieved by a client:

- A memory area for storage of BLOB data must be provided at the client.
- The server requires memory for a send buffer for returning BLOB data and a receive buffer to enable the client to receive BLOB data.
- Allocating large-object memory space for BLOB data reduces the available memory resources.
- If the end-user's software includes a middleware program that operates as a HiRDB client, the processes of sending and receiving BLOB data can further increase memory requirements.

To mitigate such increases in the memory requirements associated with BLOB data retrievals, you can provide settings so that, instead of retrieved BLOB data being returned to the client, it is output to a unit in which either a single server or a front-end server is running. This means that only the server IP address and the file name need to be returned to the client. This is called the *BLOB data file output facility*.

Figure 6-15 provides an overview of the BLOB data file output facility.

Figure 6-15: Overview of the BLOB data file output facility



* Machine at which either a single server or a front-end server is running

Explanation

1. When the BLOB data is retrieved from the client, it is output to the file row by row, and column by column.
2. The name of the file to which the BLOB data is output in (1) is returned to the client.
3. Based on the returned file name, access is made to the BLOB data file located at the server.

(1) Application criteria

Use this technique to reduce memory requirements associated with BLOB data retrievals. This technique reduces the memory required for the client program and for inter-client communication buffers, at the expense of increased disk input/output time and capacity at the server. Therefore, if this technique is to be used, the tradeoff between memory requirements and disk input/output time must be evaluated.

(2) Specification

Use of the BLOB data file output facility is specified in a `WRITE` specification of the SQL. A `WRITE` specification can be specified in either a cursor specification or a query specification. For details about `WRITE` specifications, see the *HiRDB Version 8 SQL Reference*.

The client can obtain as the SQL retrieval results only the IP address of the server, the

BLOB data storage location that is set in the SQL, and the file name of the BLOB data. With this information, the client is able to identify the particular BLOB data stored at the server.

(3) Notes on using the BLOB data file output facility

- When BLOB data files are no longer needed, they should be deleted by the user. The following point should be noted with regard to deleting a file (a file can be deleted unconditionally once the cursor has been closed or the transaction resolved):
 - If the file is deleted immediately after a `FETCH` and the BLOB data is the same as the result of the `FETCH` before the same cursor is searched, it may not be possible to re-create the file under the same file name. Because of this, you should make note of the file's original file name so that you will still be able to delete it after its name has been changed.
- The occurrence of an error or of rollback will not result in deletion of a BLOB data file that has been created. Files that are left undeleted occupy OS resources, such as disk space.
- When any of the following facilities is used, you should check that the disk has adequate free space:

`FETCH` facility with arrays

One execution of `FETCH` generates as many files as there are array elements.

Block transfer facility

The first execution of `FETCH` creates as many files as the number of rows involved in a block transfer. Subsequently, after the same number of `FETCH` operations as the number of block transfer rows, as many files as the number of block transfer rows are created each time `FETCH` is executed, and this process is repeated.

- If the file name is the same as other transactions or cursor searches, there is a potential for files to be overwritten or damaged. In such a case, for each transaction or cursor, you should change the directory name or file name in the file prefix to avoid duplicate names.

(4) Examples of using the BLOB data file output facility

Shown as follows are examples of retrievals using the BLOB data file output facility.

(a) Retrieval from BLOB columns

Retrieve columns C1 and C2 from table T1. The system outputs the BLOB data from C1 to a file, and obtains the file name. Figure 6-16 shows this example retrieval using the BLOB data file output facility (retrieval of BLOB columns).

Figure 6-16: Example of a retrieval using the BLOB data file output facility (retrieval of BLOB columns)

Table T1

C1	C2
BLOB value 1	10
BLOB value 2	20
BLOB value 3	30
BLOB value 4	40

SQL statement

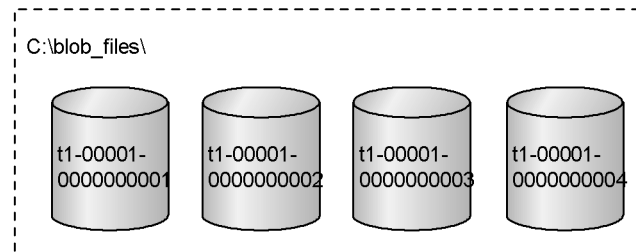
```
SELECT WRITE(C1, 'C:\blob_files\t1', 0), C2 FROM T1
```

Values returned as retrieval results

C1	C2
172.16.nn.nn:C:\blob_files\t1-00001-0000000001	10
172.16.nn.nn:C:\blob_files\t1-00001-0000000002	20
172.16.nn.nn:C:\blob_files\t1-00001-0000000003	30
172.16.nn.nn:C:\blob_files\t1-00001-0000000004	40

BLOB data output to the server

IP address: 172.16.nn.nn



(b) Retrieval of an abstract data type with the BLOB attribute

From table T2, retrieve column ADT1 that makes the `CONTAINS` function `TRUE`. In this case, the BLOB value resulting from passing the column value to an argument of the `EXTRACTS` function is output to a file, and the file name is obtained. In this example, all entries are retrieved. Figure 6-17 shows this example retrieval using the BLOB data file output facility (retrieval of an abstract data type with the BLOB attribute).

Figure 6-17: Example of a retrieval using the BLOB data file output facility (retrieval of an abstract data type with the BLOB attribute)

Table T2

ADT1
Abstract data type value 1
Abstract data type value 2
Abstract data type value 3
Abstract data type value 4

SQL statement

```
SELECT WRITE(EXTRACTS(ADT1,...),'C:\blob_files\t2',0) FROM T2
WHERE CONTAINS(ADT1,...) IS TRUE
```

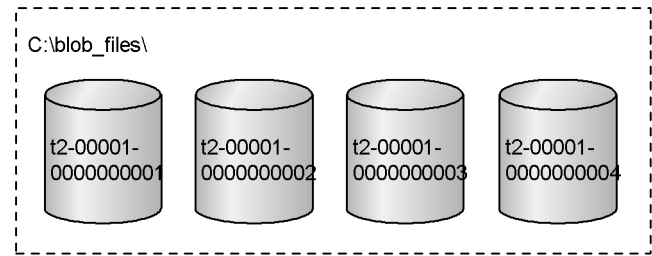


Values returned as retrieval results

ADT1
172.16.nn.nn:C:\blob_files\t2-00001-0000000001
172.16.nn.nn:C:\blob_files\t2-00001-0000000002
172.16.nn.nn:C:\blob_files\t2-00001-0000000003
172.16.nn.nn:C:\blob_files\t2-00001-0000000004

BLOB data output to the server

IP address: 172.16.nn.nn



6.8.12 BLOB and BINARY data addition update and partial extraction facility

(1) Overview of BLOB and BINARY data addition update and partial extraction facility

When BLOB or BINARY data is updated after new data has been added to a registered BLOB or BINARY data item, or when an entire BLOB or BINARY data item is retrieved as the result of BLOB or BINARY data being searched, both the server and the client must allocate a large amount of memory for the BLOB or BINARY data, putting a strain on memory resources.

Use of the BLOB and BINARY data addition update and partial extraction facility allows you to avoid straining memory resources, because it allocates only the memory required for the actual amount of BLOB or BINARY data being added or extracted.

However, for BINARY data, the definition length must be 32,001 bytes or greater.

BLOB or BINARY data addition update:

By specifying a concatenation operation in the SET clause of the UPDATE statement, you can add new data to BLOB or BINARY data that has been registered.

BLOB and BINARY data partial extraction

By specifying the SUBSTR scalar function, you can extract only a specified portion of the BLOB or BINARY data.

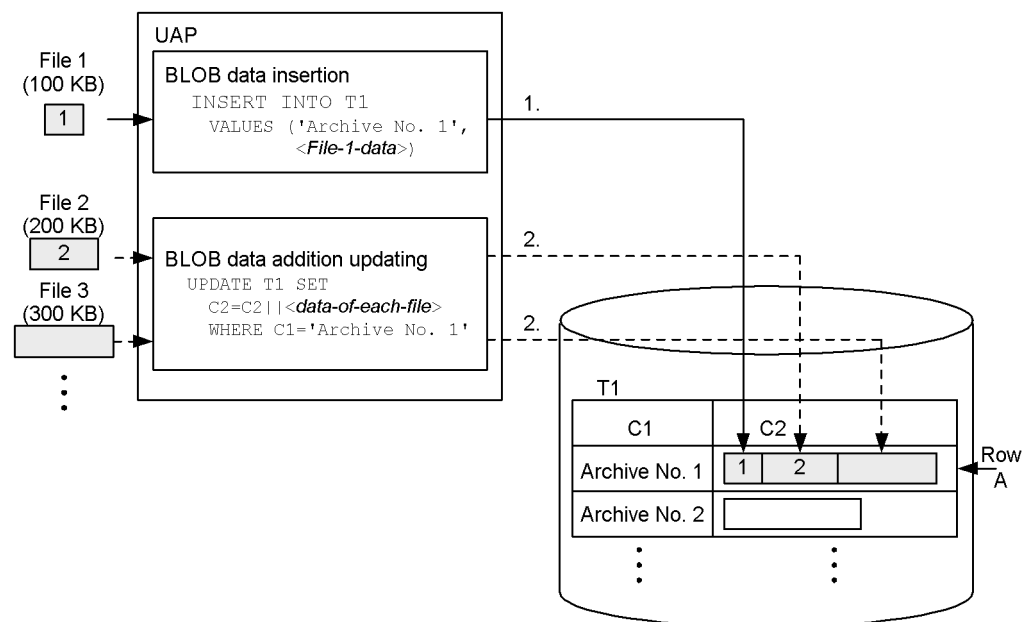
For details about the BLOB and BINARY data addition update and partial extraction facility, see the *HiRDB Version 8 UAP Development Guide*.

(2) Usage examples of the BLOB and BINARY data addition update and partial extraction facility

(a) BLOB data addition update

A single BLOB data item is stored across a number of files. Figure 6-18 shows an example of BLOB data addition updating.

Figure 6-18: Example of BLOB data addition updating



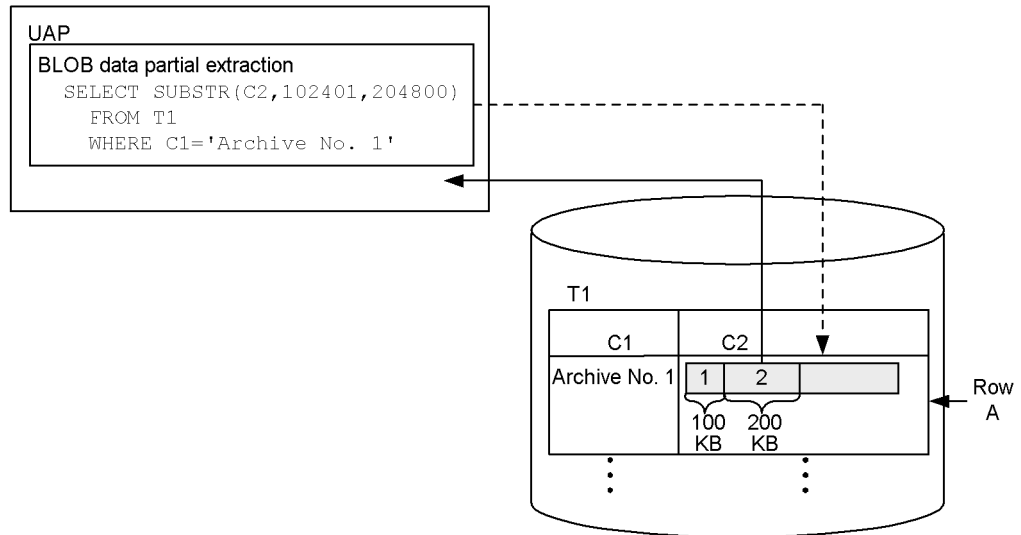
Explanation

1. BLOB data from file 1 is inserted into column C2 of row A in table T1.
2. Addition updating is performed by concatenating BLOB data of file 2 to column 2 of row A. Subsequent data additions are also performed in a similar manner.

(b) BLOB data partial extraction

An area in file 2 is extracted from the BLOB data in column C2 of row A that was stored using BLOB data addition updating. Figure 6-19 shows an example of BLOB data partial extraction.

Figure 6-19: Example of BLOB data partial extraction

**Explanation**

Using the `SUBSTR` scalar function, from column 2 of row A, the length of the data from the data column of file 2 ($200 \times 1024 = 204800$ bytes) is extracted beginning at the start position of the data column of file 2 (at the $100 \times 1024 + 1 = 102401$ byte position).

6.8.13 Locator facility**(1) Overview of the locator facility**

If a retrieved BLOB or BINARY data item is received by a client UAP as a BLOB-type or BINARY-type embedded variable, a memory area sufficient to store the received data must be allocated on the client side. This may strain the memory resources on the client side when large amounts of data are being retrieved. A large amount of data must

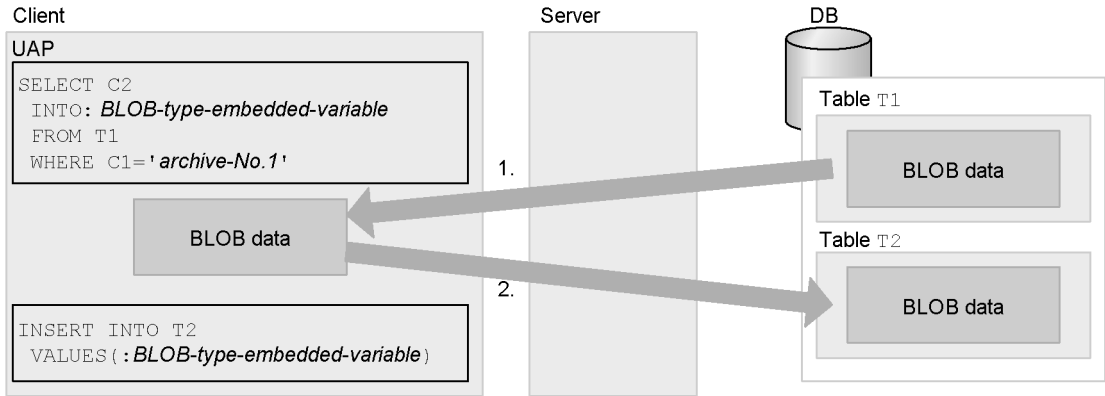
also be transferred from the server to the client. However, if only a portion of the data is needed, or if the received data will be specified without change in another SQL statement and then returned to the server, transferring the data to the client is a waste of resources.

The locator facility is designed to overcome this problem. A locator is a four-byte value that identifies data on the server. By specifying a locator embedded variable in a single-row `SELECT` statement or in the `INTO` clause of the `FETCH` statement, you can retrieve in the search results only the locator value that identifies the data rather than the entire data item. You can also specify the locator embedded variable that identifies such data in another SQL statement, which enables you to process the data identified by the locator.

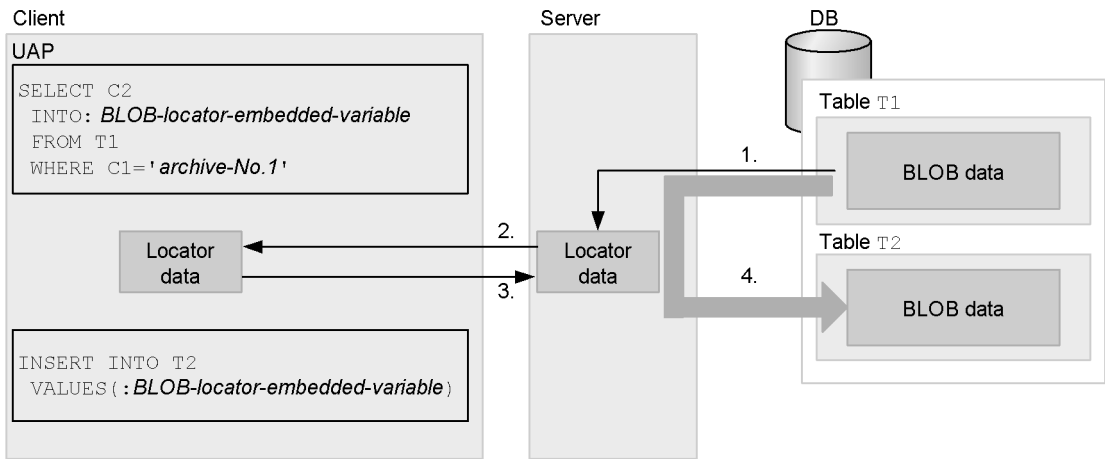
Figure 6-20 provides an overview of the locator facility.

Figure 6-20: Overview of the locator facility

- Locator facility is not used



- Locator facility is used



Explanation

Locator facility is not used:

1. The BLOB data retrieved from the database is transferred from the server to the client.
2. The BLOB data is transferred from the client to the server, and then stored in the database.

Locator facility is used:

1. The server creates a locator data item that identifies the data retrieved from the database.
2. The locator data item is transferred from the server to the client.
3. The locator data item is transferred from the client to the server.
4. The BLOB data on the server that was identified by the locator data item is stored in the database.

(2) Application criteria

The locator facility is useful whenever you retrieve BLOB or BINARY data.

By using the locator facility, the client need only allocate enough memory to store the size of the actual data being used. The amount of data transferred between the server and the client is also reduced because a locator is used.

(3) Advantages

Using the locator facility reduces the amount of memory required on the client side. It also reduces the amount of data transferred between the server and the client.

(4) How to use

To receive the value of a locator, replace in the SQL statement the embedded variable that receives the BLOB-type or BINARY-type data with the applicable locator-type embedded variable. To process the data allocated in the locator, specify the applicable locator-type embedded variable in the SQL statement, rather than specifying a BLOB-type or BINARY-type embedded variable.

For details about the locator facility, see the *HiRDB Version 8 UAP Development Guide*.

6.9 Transaction control

This section explains the *transaction control* that is performed when a HiRDB database is accessed from a UAP. A *transaction* is a logical unit of work. In the case of HiRDB, transaction control has the following meaning:

Transaction control unique to HiRDB

This encompasses transactions that are within the range of HiRDB specifications, and refers to processing in which either a `COMMIT` or a `ROLLBACK` statement is issued by an SQL statement to determine whether an updating of table data is to be put into effect or is to be invalidated.

Transaction control based on XA interface

This applies to a transaction that is executed by UAP processing when linkage to OLTP is established using the XA interface; it refers to the process that determines whether the UAP processing under OLTP using an XA interface function (the `tx_commit` or `tx_rollback` function) is to be put into effect or is to be invalidated.

6.9.1 Connection to and disconnection from HiRDB

Connection to the HiRDB system

Becoming able to access a HiRDB database by means of a UAP is called *connection to HiRDB*. Connection must be made to the HiRDB system before a transaction can be started. Connection to HiRDB is defined with the `CONNECT` control SQL statement.

Disconnection from the HiRDB system

Terminating access to the HiRDB database by the UAP and disconnecting from the HiRDB is called *disconnection from HiRDB*. When disconnection from HiRDB occurs, transactions are terminated and a synchronization point is set. Disconnection from HiRDB is defined with the `DISCONNECT` control SQL statement.

6.9.2 Multi-connection facility

A UAP of a single HiRDB client can connect simultaneously to multiple HiRDB servers. This is called the *multi-connection facility*. The multi-connection facility enables a single application process at a HiRDB client to make multiple connections to a single HiRDB server. A single application process can also connect to multiple HiRDB servers. Each of the multiple connections is treated as an independent transaction (i.e., it is handled by the HiRDB server as though each connection were made from a separate process). Because a single UAP can make multiple connections, the number of UAPs to be executed can be reduced, thus reducing the overall memory

requirements of the UAPs.

To use the multi-connection facility, a dedicated library that is provided must be linked. For details about the multi-connection facility, see the *HiRDB Version 8 UAP Development Guide*.

Setting up the multi-connection facility

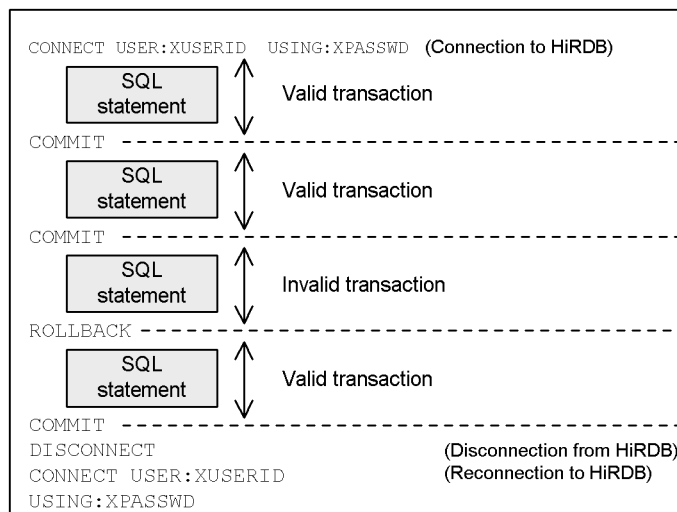
The multi-connection facility is used by defining the following SQLs in the UAP definition:

- Allocation of a connection handle (ALLOCATE CONNECTION HANDLE statement)
- Declaration of connection handle setup (DECLARE CONNECTION HANDLE SET statement)
- Cancellation of connection handle setup (DECLARE CONNECTION HANDLE UNSET statement)
- Release of connection handle (FREE CONNECTION HANDLE statement)

6.9.3 Transaction startup and termination

A transaction is started when the first SQL is executed, and a transaction terminates when a synchronization point is set (a commit or rollback). Any number of transactions can be started and terminated while connection to HiRDB is maintained. Figure 6-21 shows examples of transaction startup and termination.

Figure 6-21: Examples of transaction startup and termination



6.9.4 Commit and rollback

The process of placing into effect in the database the updates made by a transaction is called *commit*. The process of invalidating the updates made by a transaction is called *rollback*. When commit and rollback occur is determined as follows:

- At the times defined by an SQL
- At the times that are set automatically by HiRDB

Each of these types of commit and rollback timing is explained as follows.

(1) Commit timing

This section explains commit timing.

Timing defined by an SQL

The `COMMIT` control SQL statement can be specified to commit a transaction whenever a `COMMIT` statement is executed.

Commit set automatically by HiRDB

- Commit is performed automatically by HiRDB when a definition SQL or `PURGE TABLE` statement is executed.
- Commit is performed automatically by HiRDB when a UAP terminates.

(2) Rollback timing

This section explains rollback timing.

Timing defined by an SQL

The `ROLLBACK` control SQL statement can be specified to roll back the process to the previous commit point whenever a `ROLLBACK` statement is executed.

Rollback set automatically by HiRDB

- If a process cannot continue during SQL execution, HiRDB will roll back the process implicitly to the previous commit point.
- If a UAP terminates abnormally, HiRDB will roll back the process to the previous commit point.

(3) Commitment control on a HiRDB/Parallel Server

A HiRDB/Parallel Server provides the following two methods of commitment control:

- One-phase commit
- Two-phase commit

(a) One-phase commit

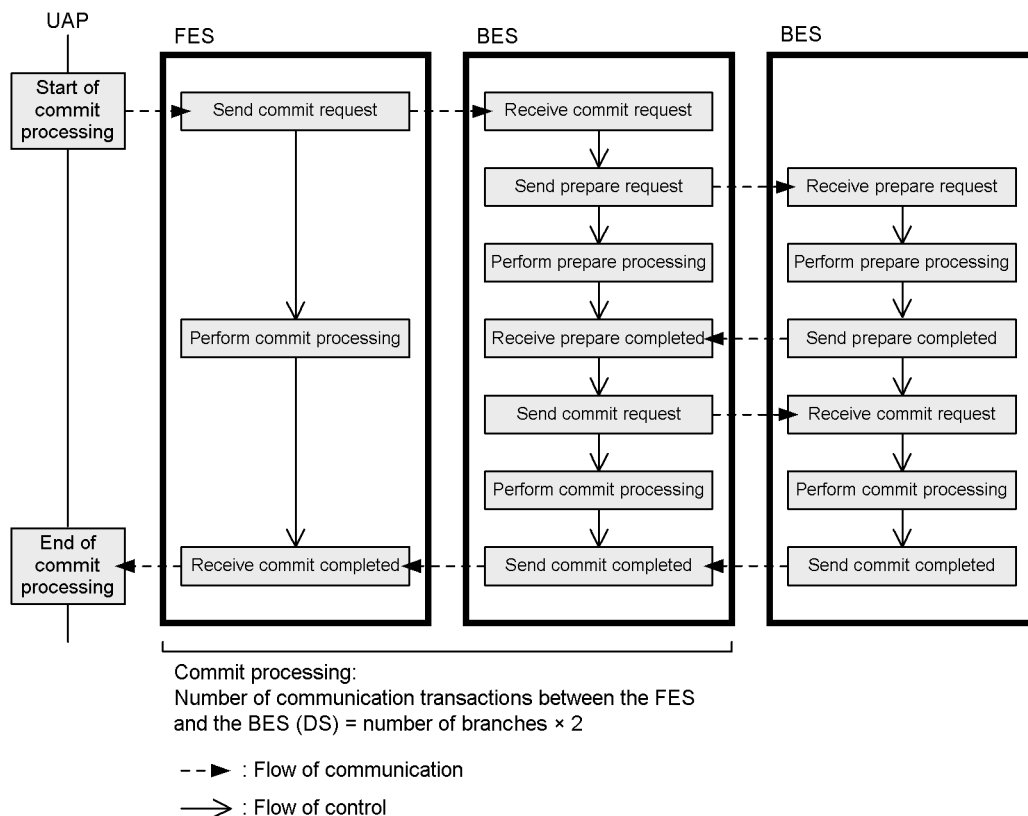
With one-phase commit, only commit processing is performed, rather than both

prepare processing and commit processing (which is the case with two-phase commitment control). This means that the number of communication transactions for synchronization point processing between the front-end server and the back-end server (dictionary server) is the number of branches $\times 2$ (whereas, with two-phase commit, it is the number of branches $\times 4$), which improves transaction processing performance. To use one-phase commit, specify `ONEPHASE` (default) in the `pd_trn_commit_optimize` operand.

One-phase commit is used only when one branch within a single transaction is being updated. Otherwise, two-phase commit must be used.

Figure 6-22 shows the processing for one-phase commit.

Figure 6-22: Processing for one-phase commit

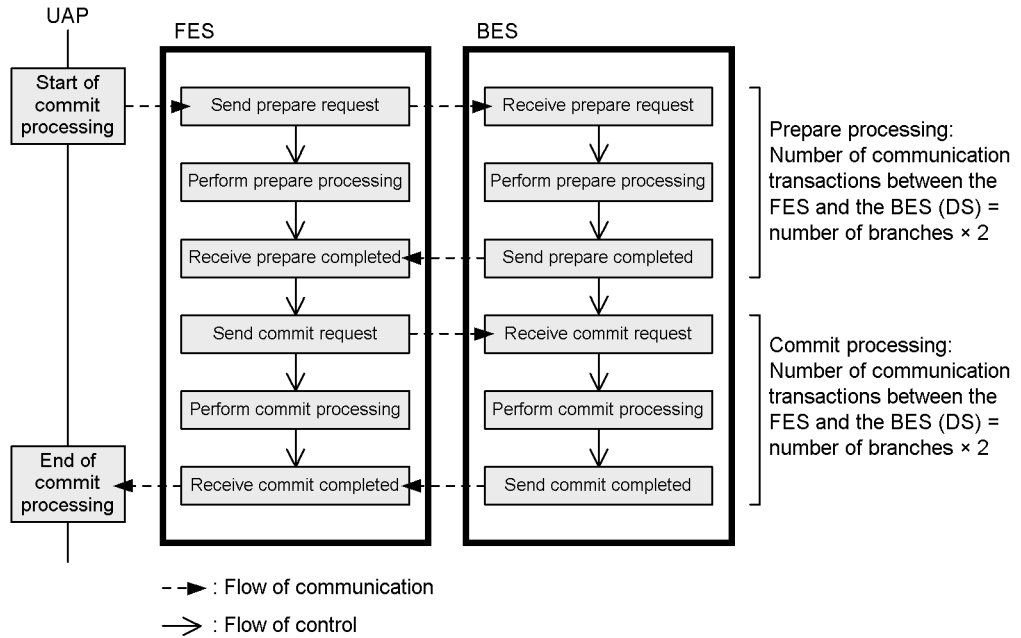


The process of executing one-phase commit for commitment control is called *one-phase optimization*.

(b) Two-phase commit

Two-phase commit performs synchronization point processing of the transaction by separating commitment control into two phases, prepare processing and commit processing. The number of communication transactions for synchronization point processing between the front-end server and the back-end server (dictionary server) is the number of branches \times 4. Figure 6-23 shows the processing for two-phase commit.

Figure 6-23: Processing for two-phase commit



The type of commitment control used on a HiRDB/Parallel Server is determined by the issuer of the commit and the execution environment of the transaction. Table 6-11 shows how commitment control is determined on a HiRDB/Parallel Server.

Table 6-11: Commitment control on a HiRDB/Parallel Server

Condition			HiRDB commitment control
Issuer of the commit	Commitment control indicated by the commit issuer	Execution environment of the transaction	
UAP	—	For a reference transaction	One-phase commit
		If one server is updated by the transaction and <code>ONEPHASE</code> is specified (or nothing is specified) in the <code>pd_trn_commit_optimize</code> operand	
		Other than the above	Two-phase commit
OLTP system	One-phase commit	For a reference transaction	One-phase commit
		If one server is updated by the transaction and <code>ONEPHASE</code> is specified (or nothing is specified) in the <code>pd_trn_commit_optimize</code> operand	
		Other than the above	Two-phase commit
	Two-phase commit	For a reference transactions	One-phase commit
		Other than the above	Two-phase commit

—: Not applicable

6.9.5 UAP transaction management under OLTP environment

When transaction processing that conforms to the XA interface is performed using a UAP under the OLTP environment, commit or rollback of the transaction is executed from the UAP using an *API that is compatible with X/Open*.

Transaction transfer

Execution of transaction commitment using a different process from the one the UAP used to access HiRDB is called *transaction transfer*. UAP in this context refers to the UAP that connects to HiRDB by means of the HiRDB XA library.

6.9.6 Automatic reconnect facility

When a connection to a HiRDB server is lost due to a server process going down, system switchover, network failure, or other cause, the *automatic reconnect facility* automatically re-establishes the connection. By using the automatic reconnect facility, you can continue execution of a UAP without having to be aware that the connection to the HiRDB server was lost. To use the automatic reconnect facility, specify `YES` in

the `PDAUTORECONNECT` client environment definition.

Application criteria

Whenever you execute the system reconfiguration command or whenever you are updating to the HiRDB update version, all HiRDB clients are placed in wait status until that processing is completed. While in wait status, the elapsed time is monitored against the time specified in `PDCWAITTIME`. If the time specified in `PDCWAITTIME` is exceeded, wait status is released, and a `PDCWAITTIME` timeout error is returned to the UAPs. Depending on the timing of this error, communications processing errors may occur because the UAPs may not be able to detect that the system reconfiguration command is executing or that updating to the HiRDB update version is being performed. If you know in advance that the system reconfiguration command will be executed or that updating to the HiRDB update version will be performed, consider using the automatic reconnect facility. By using this facility, processing continues without an error being returned to the UAPs, even when the system reconfiguration command is executing or updating to the HiRDB update version is being performed.

6.10 Locking

When multiple users attempt to manipulate data in the same table at the same time, the HiRDB system automatically implements locking in order to maintain data integrity. Especially inasmuch as the individual servers in a HiRDB/Parallel Server do not in principle share resources, an independent means of locking is used for each server. Depending on the nature of the operation to be performed, you can modify a lock that is implemented automatically by HiRDB.

6.10.1 Units of locking

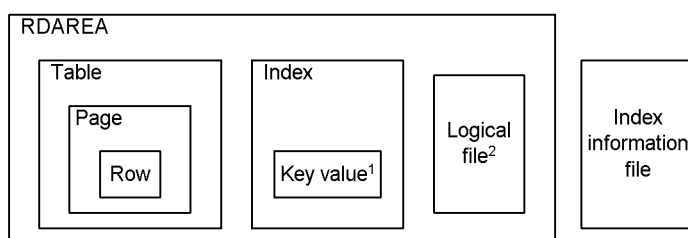
This section explains the units of locking and their inclusion relationships.

(1) *Locked resources and their inclusion relationships*

HiRDB prevents illegal referencing and updating by applying locks based on a unit called the *locked resource*. Locking is applied sequentially from top to bottom of the locked resources. If when locking is applied there is a transaction that cannot execute because it is in contention with other transactions for a resource, that transaction goes onto wait status.

Locked resources have inclusion relationships. Therefore, once a lock is applied to a higher-order resource, there is no need to apply locks to the resources that are below it hierarchically. Figure 6-24 shows locked resources and their inclusion relationships. For example, a table is a higher-order resource and a page is a lower-order resource.

Figure 6-24: Locked resources and their inclusion relationships



¹ If index key value no-lock is applied, the key value is not locked. For details about index key value no-lock, see the *HiRDB Version 8 UAP Development Guide*.

² Logical file used by a plug-in.

(2) *Setting the lowest-order locked resource unit*

For each table, the user can set the lowest-order locked resource unit for the locking to be applied automatically by HiRDB. The lowest-order locked resource unit, the setting procedures, and their advantages and disadvantages are explained as follows.

(a) Locking by row

To set the row as the lowest-order unit of locking, `LOCK ROW` is specified in the `CREATE TABLE` definition SQL. Locking that uses the row as the lowest-order unit of locking is called *row locking*. Compared to locking by page, locking by row results in better concurrent execution because it is a lower unit of locked resources. On the other hand, locking by row increases the processing time required for locking and increases memory usage.

(b) Locking by page

To set the page as the lowest-order unit of locking, `LOCK PAGE` is specified in the `CREATE TABLE` definition SQL. Locking that uses the page as the lowest-order unit of locking is called *page locking*. Compared to locking by row, locking by page reduces the processing time required for locking and reduces memory usage. On the other hand, locking by page results in poor concurrent execution.

6.10.2 Lock modes

HiRDB provides the following five *lock modes* for each type of locked resource (such as table or page):

1. Shared mode (PR: Protected Retrieve)

One transaction monopolizes the locked resource and allows only referencing by other transactions.
2. Exclusive mode (EX: EXclusive)

One transaction monopolizes the locked resource and does not allow referencing, adding, updating, or deletion by other transactions.
3. Shared retrieval mode (SR: Shared Retrieve)

If the lock is applied to a certain resource in the shared mode, the shared retrieval mode applies to the resources above that locked resource; referencing, adding, updating, and deletion of the locked resource can be performed by other transactions.
4. Shared update mode (SU: Shared Update)

If the lock is applied to a certain resource in the exclusive mode, the shared update mode applies to the resources above that locked resource; referencing, adding, updating, and deletion can be performed by other transactions.
5. Protected update mode (PU: Protected Update)

Referencing, adding, updating, and deletion can be performed; referencing only can be performed by other transactions. Unlike modes 1-4, this mode occurs as a result of lock mode transition.

6.10.3 Automatic locking by HiRDB

This section explains the locking of tables and rows that is performed automatically by HiRDB during data updating, addition, and deletion and during data retrieval.

(1) Locking by HiRDB during data updating, addition, and deletion

Locking of a table

Referencing, adding, updating, and deletion of the table are allowed by other transactions as well. In other words, the *shared update (SU) mode* goes into effect.

Locking of a row

The transaction monopolizes the resource and does not allow referencing, adding, updating, or deletion of the row by other transactions. In other words, the *exclusive (EX) mode* goes into effect.

(2) Locking by HiRDB during data retrieval

Locking of a table

Referencing, adding, updating, and deletion of the table are allowed by other transactions as well. In other words, the *shared retrieval (SR) mode* goes into effect.

Locking of a row

The transaction monopolizes the resource and allows only referencing of the row by other transactions. In other words, the *shared (PR) mode* goes into effect.

6.10.4 Changing the lock based on a user setting

The user may change the lock that is implemented automatically by HiRDB. Locking of tables and rows during data updating, addition, and deletion and during data retrieval is explained as follows.

(1) Locking during data updating, addition, and deletion based on a user setting

When `IN EXCLUSIVE MODE` is specified in the `LOCK TABLE` statement, the user may change the lock to be used during data updating, addition, and deletion. Referencing, adding, updating, or deletion of data in the table by other transactions are not allowed. In other words, the *exclusive (EX) mode* goes into effect. On the other hand, because row-based locking is not applied, the overhead for applying the lock is reduced, thus preventing a shortage of buffers for locking. However, because table-based locking is applied, other transactions may have to wait for a long time.

(2) Locking during data retrieval based on a user setting

When either of the following SQLs is specified, the user may change the lock to be used during data retrieval:

WITHOUT LOCK of SELECT statement

The data that is retrieved is not locked until the transaction terminates. Because locking is released beginning with rows that have been retrieved, better concurrent execution can be achieved.

IN SHARE MODE of LOCK TABLE statement

The transaction monopolizes the resource and allows only referencing of the data in the table by other transactions. In other words, the *shared (PR) mode* goes into effect. On the other hand, because row-based locking is not applied, the overhead for applying the lock is reduced, thus preventing a shortage of buffers for locking. However, because table-based locking is applied, other transactions may have to wait for a long time.

6.10.5 Lock period

When a transaction locks a resource, it monopolizes that resource until commit or rollback occurs. For example, when an update is made to a locked resource (row or page), the automatic locking by HiRDB implements the exclusive mode, which does not allow referencing, adding, updating, or deletion by other transactions. Consequently, other transactions must wait until commit or rollback occurs before it becomes possible to access the row that is being updated. However, if `UNTIL DISCONNECT` is specified in the `LOCK` statement, the lock will remain in effect until a `DISCONNECT` statement is executed or until commit occurs after the table is deleted.

6.10.6 Deadlock

Deadlock is the status when two transactions are both waiting for the other to release resources when they have allocated the same resources but in a different order. Consequently, processing by both transactions is stalemated and cannot proceed. Deadlock often occurs between a referencing transaction and an updating or deletion transaction. Changing the access order of the UAPs can reduce the incidence of deadlocks. When deadlock occurs during operation of HiRDB, HiRDB outputs deadlock information that includes information on deadlocks that have occurred among multiple transactions in the server. For details about preventing deadlock, see the *HiRDB Version 8 UAP Development Guide*.

6.11 Operation without collecting a database update log

HiRDB acquires a history of database updates (the database update log in the system log) by UAPs (or utilities^{*}). This information is collected into system log files. Acquisition of database update log information can also be disabled; the advantage of doing so is that processing time, and consequently, execution time, for the UAP (or utility) is reduced.

* The following utility programs can be used:

- Database load utility (pdload)
- Database reorganization utility (pdrorg)
- Rebalancing utility (pdrbal)

(1) Acquiring a database update log

There are three modes for collecting the database update log that is used during execution of a UAP (or a utility), as shown in Table 6-12.

Table 6-12: Database update log acquisition modes

Database update log acquisition mode	Explanation
Log acquisition mode	Acquires the database log needed for rollback or roll-forward. Normally, the log acquisition mode is used.
Pre-update log acquisition mode	Acquires only the database update log necessary for a rollback.
No-log mode	Does not acquire a database update log.

(2) Specifying the database update log acquisition mode

Table 6-13 shows the methods for specifying the database update log acquisition mode.

Table 6-13: Methods of specifying the database update log acquisition mode

Database update log acquisition mode specification method	Explanation
UAP	The database update log acquisition method is specified in the PDDBLOG operand of the client environment definition. You can specify in the PDDBLOG operand either the log acquisition mode or the no-log mode; the pre-update log acquisition mode cannot be specified.

Database update log acquisition mode specification method	Explanation
Database load utility (pdload)	The database update log acquisition method is specified in the -1 option of the database load utility (pdload).
Database reorganization utility (pdroorg)	The database update log acquisition method is specified in the -1 option of the database reorganization utility (pdroorg).
Rebalancing utility (pdrbal)	The database update log acquisition method is specified in the -1 option of the rebalancing utility (pdrbal).
Using a user LOB RDAREA	In the case of data stored in a user LOB RDAREA, the database update log acquisition method is specified in the RECOVERY operand of CREATE TABLE. If abstract data type data is stored in the LOB RDAREA, the pre-update log acquisition mode cannot be specified (it is ignored if specified).

(3) Notes on the RECOVERY operand

The database update log acquisition method specified in the RECOVERY operand is subject to change when either the PDDBLOG operand or the -1 option is specified. Table 6-14 shows the relationship between the specification value in the RECOVERY operand and the PDDBLOG operand or the -1 option, on the one hand, and the value that is assumed during execution of the UAP (or the utility), on the other hand.

Table 6-14: Relationship between the specification value in the RECOVERY operand and PDDBLOG operand or the -1 option, and the value that is assumed during execution of the UAP (or the utility)

Specification of PDDBLOG operand or -1 option	Specification of RECOVERY operand		
	ALL	PARTIAL	NO
ALL a (log acquisition mode)	ALL	PARTIAL	NO
PARTIAL p (pre-update log acquisition mode)	PARTIAL	PARTIAL	NO
NO n (no-log mode)	NO	NO	NO

ALL or a: log acquisition mode

PARTIAL or p: pre-update log acquisition mode

NO or n: no-log mode

(4) Differences in operating method depending on database update log acquisition mode

The different database update log acquisition modes can result in differences in the operating methods in the following two areas:

- Processing performed by HiRDB and action to be taken by the user in the event of abnormal termination of the UAP
- Timing for database recovery

(a) Processing performed by HiRDB and the action to be taken by the user in the event of abnormal termination of the UAP

Table 6-15 shows the processing performed by HiRDB and the action to be taken by the user in the event of abnormal termination of the UAP.

Table 6-15: Processing performed by HiRDB and the action to be taken by the user in the event of abnormal termination of the UAP

Database update log acquisition mode	HiRDB processing	User action
Log acquisition mode	Rolls back an updated RDAREA to its status before the UAP executed or to the last synchronization point set before the abnormal termination.	If the RDAREA is rolled back to its status before the UAP was executed, re-execute the UAP. If the RDAREA is rolled back to the last synchronization point that was set before the abnormal termination, execute the processing following the synchronization point.
Pre-update log acquisition mode		
No-log mode	Does not perform roll back. Error shutdown (logless shutdown) is performed on the updated RDAREA. The contents of the RDAREA are destroyed.	Using as input information a backup that was made before execution of the UAP, execute the database recovery utility (<code>pdrstr</code>) to recover the RDAREA, and then re-execute the UAP.

(b) Timing at which database can be recovered

Table 6-16 shows the times at which the database can be recovered by the database recovery utility (`pdrstr`).

Table 6-16: Times for recovering the database

Database update log acquisition mode	Time for database recovery
Log acquisition mode	Either the backup acquisition point or any synchronization point subsequent to the backup acquisition point

Database update log acquisition mode	Time for database recovery
Pre-update log acquisition mode	Backup acquisition point
No-log mode	

(5) Notes on backup (important)

1. During execution of a UAP (or utility) in the no-log mode or the pre-update log acquisition mode, do not make a backup using the `updateable` mode (`-M s` specified).
2. After execution of a UAP (or utility) in the no-log mode or the pre-update log acquisition mode, make a backup in one of the following modes:
 - Referencing/updating-impossible mode (`-M x` specified)
 - Referencing-permitted mode (`-M r` specified)

Chapter

7. Database Management

This chapter explains how the HiRDB administrator manages a database.

This chapter contains the following sections:

- 7.1 Database recovery
- 7.2 Preparations for database errors
- 7.3 Reorganizing tables and indexes
- 7.4 Reusing used free pages and used free segments
- 7.5 Adding, expanding, and moving RDAREAs
- 7.6 Space conversion facility
- 7.7 Facility for conversion to a decimal signed normalized number

7.1 Database recovery

The database recovery utility (`pdrstr`) is provided in the event a database needs to be recovered because of a disk error or for some other reason. This section briefly explains how to recover databases. For details about how to recover databases, see the *HiRDB Version 8 System Operation Guide*.

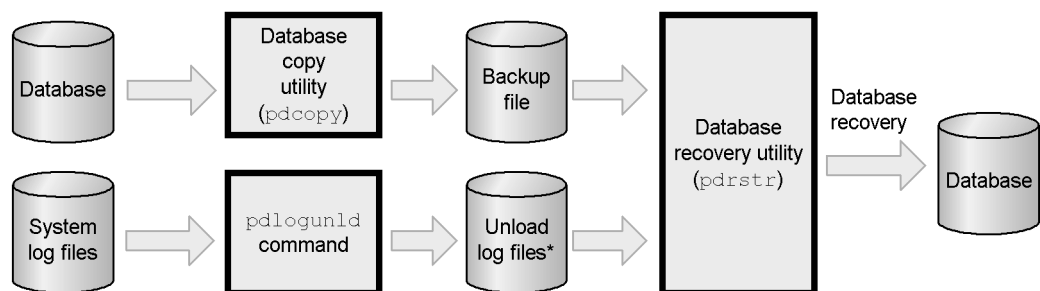
7.1.1 Overview of database recovery

To recover a database, you use the following files as input information for the database recovery utility (`pdrstr`).

- Backup files
- Unload log files

Figure 7-1 provides an overview of the database recovery process.

Figure 7-1: Overview of database recovery process



* An *unload log file* is created by unloading system log files (system log information) that contain information comprising a history of the updates to the database.

Explanation

- Backups are made by the database copy utility (`pdcopy`).
- The `pdlogunld` command creates an unload log file.
- Using the backup and unload log files as the input, the database recovery utility (`pdrstr`) recovers the database.

For details about making backup files and of creating unload log files, see Section 7.2 *Preparations for database errors*.

7.1.2 Times at which database can be recovered

A database can be recovered to its status at the following times:

- At the time a backup was made
- To its most recent status (the point of the most recent synchronization point)

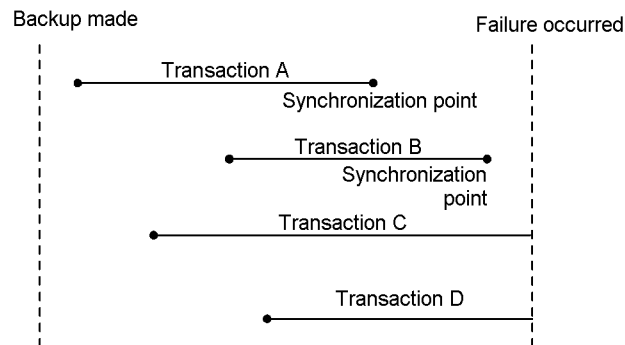
(1) Recovering to the point a backup was made

To recover a database to its status when a backup was made, you do not need an unload file. Only the backup file is required.

(2) Recovering to the most recent synchronization point before the failure occurred

The point in time at which a transaction is completed is called a *synchronization point*. Synchronization point processing that validates a transaction-induced update is called a *commit*. Synchronization point processing that invalidates a transaction is called a *rollback*. Recovering a database to a transaction synchronization point is called *recovering to the most recent synchronization point before the failure occurred*. Because any transaction being processed when the failure occurred (a transaction that had not reached a synchronization point) is invalid, update processing performed by such transactions is not recovered. Figure 7-2 shows the transactions that are recovered.

Figure 7-2: Transactions that are recovered (with recovery to the most recent synchronization point before the failure occurred)



Explanation

Because processing of transactions A and B finished and they reached their synchronization points, the database is recovered to these synchronization points.

Because processing of transactions C and D was ongoing, the processing of these transactions is invalid, and is not recovered.

To recover to the most recent synchronization point before the failure occurred, in addition to backup files, you need the unload log files from which log files are output after a backup was made.

7.2 Preparations for database errors

If an error occurs in the database as a result of a disk error or for some other reason, the HiRDB administrator must perform the database recovery operation. To be prepared for database errors, the HiRDB administrator is responsible for performing the following tasks:

- Making backups
- Creating unload log files (unloading system log information)

In addition, the following functions are available as options for making backups:

- Differential backup facility
- Backup hold
- Frozen update command (`pddbfrz` command)
- NetBackup linkage facility

7.2.1 Making backups

To be prepared for database errors, it is important to make backup copies of a database on a regular basis; the *database copy utility* (`pdcopy`) is provided for this purpose.

(1) Backup units

Backups can be made in the units shown in Table 7-1. The unit is specified in an option of the database copy utility (`pdcopy`).

Table 7-1: Backup units

Backup unit	Explanation	<code>pdcopy</code> option specification
System	Makes a backup of all RDAREAs, including RDAREAs used by the system (such as the master directory RDAREA).	<code>-a</code>
Unit*	Makes a backup of a specified unit's RDAREAs.	<code>-u unit-name</code>
Server*	Makes a backup of a specified server's RDAREAs.	<code>-s server-name</code>
RDAREA	Makes a backup of a specified RDAREA.	<code>-r RDAREA-name</code>

* Applicable to HiRDB/Parallel Servers.

(2) Backup acquisition modes

A backup acquisition mode is specified in the `-M` option of the database copy utility (`pdcopy`). Table 7-2 lists the backup acquisition modes.

Table 7-2: Backup acquisition modes

Backup acquisition mode (-m option specification)	Explanation	Difference in RDAREA recovery depending on backup acquisition point
Referencing/ updating-impossible mode (x)	While the backup is being made, RDAREAs being backed up cannot be referenced or updated. Before a backup, you must use the <code>pdhold -c</code> command to shut down and close the affected RDAREAs.	A backup made with this method can be used to recover the database to its status when the backup was made. In addition, by using system log information, you can recover the database to any synchronization point since the backup was made.
Referencing-permitted mode (r)	While the backup is being made, the RDAREAs being backed up can be referenced but not updated.	
Updatable mode (s) ¹	While the backup is being made, the RDAREAs being backed up can be both referenced and updated.	The database cannot be recovered to its status when the backup was made, but it can be recovered to any synchronization point since the backup was made. Therefore, to recover the database, you need the backup and the system log information ² from a synchronization point subsequent to when the backup was made.

¹ A backup should not be made in the updatable mode during execution of a UAP (including a utility) that is running in the no-log mode or the pre-update log acquisition mode.

² The LAN IDs and generation numbers of the system log files required to recover the RDAREA are output to the process results output file of the database recovery utility.

(3) Server machine for storage of backup file

A backup file can be created anywhere on a server machine on which HiRDB is running. A backup file need not be created on the server machine on which the RDAREAs are located. Devices such as CMT and DAT may not be available on the same server machine as the RDAREAs being backed up. The server machine on which a backup file is to be stored is specified in an option of the database copy utility (`pdcopy`).

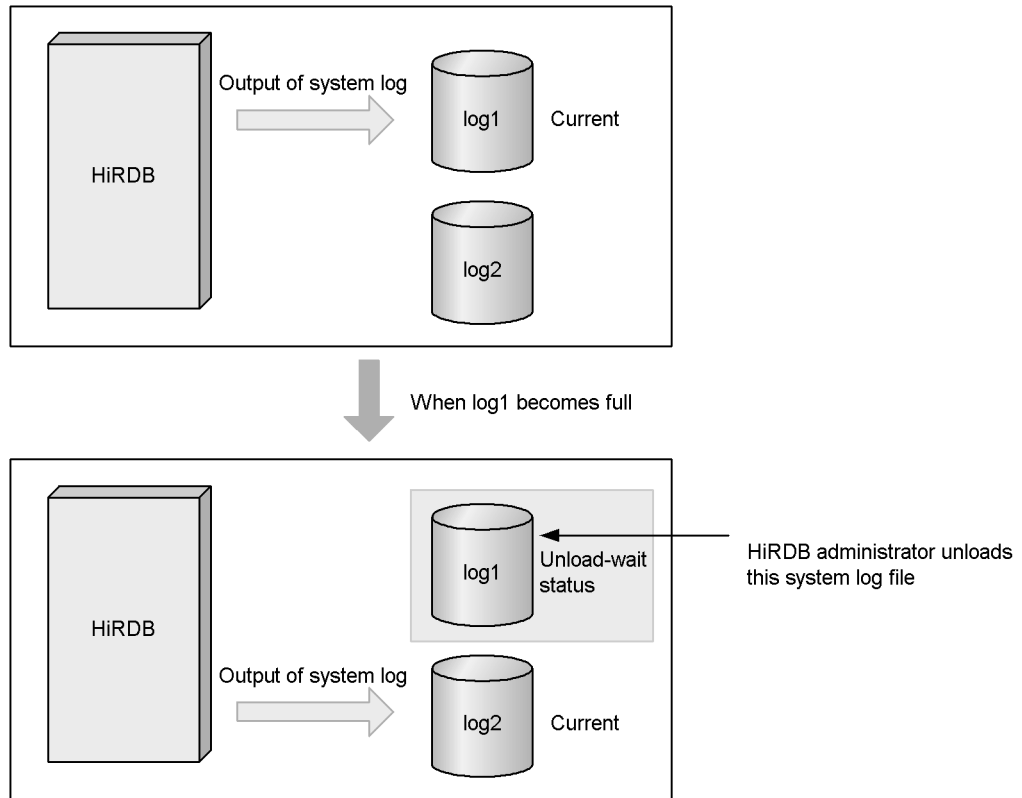
7.2.2 Creating an unload log file (unloading system log)

The `pdlogunld` command can be used to unload system log information and create an unload log file. Before you create an unload log file, you should check the status of the system log files.

(1) Status of system log files

The file to which system log information is output is called the *current file*. If system log information fills the current file, the output for the system log information is changed to another system log file in a process called *system log file swapping*; what was the current file becomes an *unload-wait file*. The HiRDB administrator uses the `pdlogunld` command to unload unload-wait files. Figure 7-3 shows changes the system log file status changes.

Figure 7-3: System log file status changes



System log information cannot be output to an unload-wait file. *If all system log files become unload-wait files, system log information cannot be output any longer, and HiRDB terminates abnormally.* For this reason, it is essential that system log information be unloaded.

Notes on starting HiRDB normally

When HiRDB is started normally following a normal termination, system log files are swapped, creating an unload-wait file. This file should be unloaded

without fail. *If termination and startup are repeated without unloading system log information, HiRDB will eventually be unable to start.*

Specifying `pd_log_rerun_swap=Y` in the server definition causes system log files to be swapped when HiRDB is restarted.

(2) Determining the status of the system log files

You can check the status of the system log files with the `pdlogls` command, as shown in the following example:

Example

Use the `pdlogls` command to check the status of the system log files.

```
pdlogls -d sys
HOSTNAME : k95x620(155702)
Group   Type Server  Gen No.  Status  Run ID  Block No.
log1    sys  sds01    1  osu---u  36e75ad6  1  2
log2    sys  sds01    1  oc-d--u  36e873b3  1  c
log3    sys  sds01    0  os-----  00000000  0  0
log4    sys  sds01    0  os-----  00000000  0  0
log5    sys  sds01    0  os-----  00000000  0  0
log6    sys  sds01    0  cn-----  00000000  0  0
```

The status of the system log files is shown in the second and third columns in this area.

Explanation

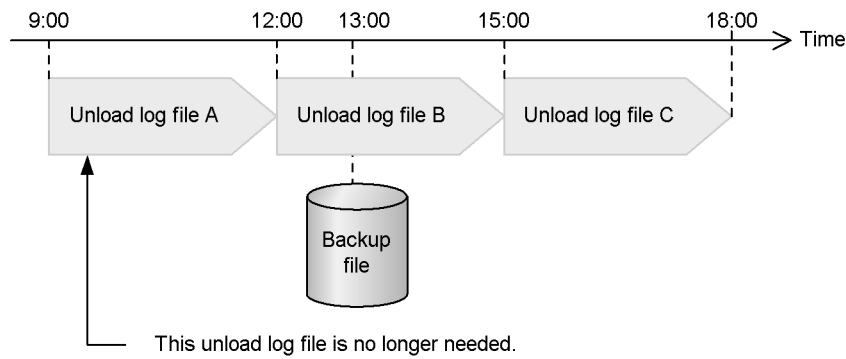
- The file (`log1`) for which there is a "u" in column 3 is an unload-wait file. This file should be unloaded.
- The file (`log2`) for which there is a "c" in column 2 is the current file.

(3) Retaining unload log files

Repeated unloading of system log information results in a proliferation of unload log files, which occupy disk space. The HiRDB administrator should delete unload log files as soon as they are no longer needed.

When a backup is made, unload log files containing only data produced prior to the backup are no longer needed. Figure 7-4 shows the relationship between a backup and unload log files.

Figure 7-4: Relationship between backup and unload log files



Explanation

- The system log information that was output from 9:00 to 12:00 is unloaded to unload log file A.
- The system log information that was output from 12:00 to 15:00 is unloaded to unload log file B.
- The system log information that was output from 15:00 to 18:00 is unloaded to unload log file C.
- If a backup is made around 13:00, the backup and unload log files B and C will be needed for any recovery of the database, but unload log file A will not be needed.

Note:

Unload log files should be stored on a different disk from the disk containing the database. If they are all stored on the same disk, an error affecting the disk will prevent recovery of the database.

(4) Automatic log unloading facility for system log files

As explained above, the HiRDB administrator must use the `pdlogunld` command to unload any unload-wait system log files. If this operation is not performed, you may run out of system log files that can be swapped in when swapping occurs, causing HiRDB to terminate abnormally.

HiRDB provides a facility called the *automatic log unloading facility* that automates the unloading of system log files. When the unloading operation must be performed frequently, consider automating it.

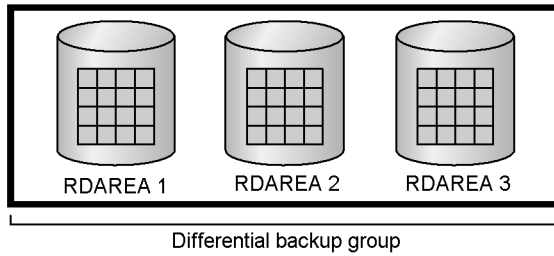
For details about the automatic log unloading facility, see the *HiRDB Version 8 System Operation Guide*.

7.2.3 Differential backup facility

Because backups are normally made on a per-RDAREA basis, the backup includes pages that have been updated as well as pages that have not been updated since the last backup. By using the differential backup facility, only the pages that have been updated since the last time a backup was made are backed up. Making a backup of only the differences since the last time a backup was made allows you to significantly reduce the time required to make a backup. Use of the differential backup facility should be considered in situations where the database is large and the amount of data updating is relatively small. Figure 7-5 provides an overview of the differential backup facility.

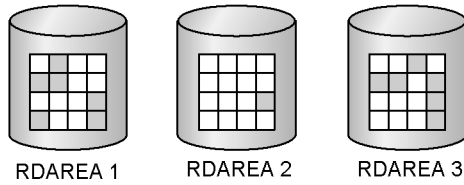
Figure 7-5: Overview of the differential backup facility

1. Backup made on Sunday (full backup)



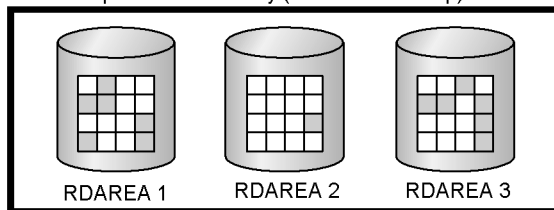
A full backup is performed on the used pages in RDAREAs 1 to 3 (shaded pages).

2. Updates made on Monday



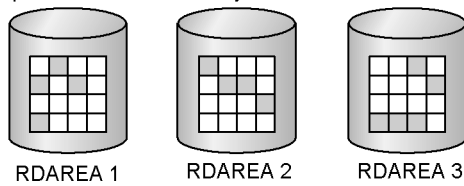
The pages updated since the full backup made on Sunday are shaded.

3. Backup made on Monday (differential backup)



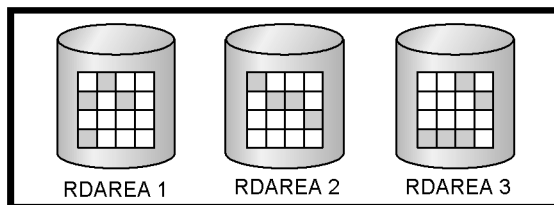
A differential backup is performed on the pages that were updated on Monday.

4. Updates made on Tuesday



The pages updated since the differential backup made on Monday are shaded.

5. Backup made on Tuesday (differential backup)



A differential backup is performed on the pages that were updated on Tuesday.

: Page

Explanation

1. On Sunday, backups are made of RDAREAs 1 through 3. At this time, a backup is made of the used pages in RDAREAs 1 through 3. This kind of backup is called a *full backup*, and the grouped RDAREAs are called a *differential backup group*.
2. Updates are performed with Monday's work.
3. At the end of Monday's work day, a backup is made of RDAREAs 1 through 3. This time, a backup is made only of the pages in RDAREAs 1 through 3 that were updated. This kind of backup is called a *differential backup*.
4. Updates are performed with Tuesday's work.
5. At the end of Tuesday's work day, a backup is made of RDAREAs 1 through 3. At this time, a backup is made only of the pages in RDAREAs 1 through 3 that were updated.

Database recovery procedure

To recover a database up to the point the differential backup was made on Tuesday, as the input information for the database recovery utility, you use the full backup made on Sunday, the differential backup made on Monday, and the differential backup made on Tuesday. For details about how to operate the differential backup facility, see the *HiRDB Version 8 System Operation Guide*.

Note:

A differential backup cannot be made of a LOB RDAREA.

Reference note:

The differential backup facility can also be used during operation without unloading the system log.

7.2.4 Backup-hold

When a backup is made with a command other than the `pdcopy` command (i.e., when another product's backup facility is used), the RDAREAs should be placed on *backup-hold*. An RDAREA on backup-hold can be backed up using the backup facility of another product while HiRDB is running. To place an RDAREA on backup-hold, you specify the `-b` option in the `pdhold` command.

(1) Application example of backup-hold

If you execute the `pdcopy` command in updatable mode, you must place the RDAREA to be backed up on backup-hold (`-b u` or `-b wu`).

(2) Backup-hold types

Table 7-3 lists and describes the four types of backup-hold.

Table 7-3: Types of backup-hold

Backup-hold type	Explanation
Referencing-permitted backup-hold	During backup-hold, RDAREAs that have been backed up and held can be referenced; an updating attempt will result in an SQL error (-920).
Referencing-permitted backup-hold (update WAIT mode)	During backup-hold, RDAREAs that have been backed up and held can be referenced; an updating attempt goes onto lock-release wait status until the backup-hold is released.
Updatable backup-hold	During backup-hold, RDAREAs that have been backed up and held can be referenced and updated. Even while an updating transaction is being executed, an RDAREA is placed immediately on updatable backup-hold status without placing the <code>pdhold</code> command on wait status.
Updatable backup-hold (WAIT mode)	During backup hold, RDAREAs that have been backed up and held can be referenced and updated. If an updating transaction is being executed, the <code>pdhold</code> command is kept waiting until the transaction terminates.

For details about how to make a backup using the backup shutdown, see the *HiRDB Version 8 System Operation Guide*.

Reference note:

Reference-possible backup hold and reference-possible backup hold (update WAIT mode) are also referred to as *committing a database*.

7.2.5 Reducing the time needed to make backups of user LOB RDAREAs (frozen update command)

Using the frozen update command (`pddbufrz` command) allows you to reduce the time needed to make backups of user LOB RDAREAs. Consider using the frozen update command for the following operations:

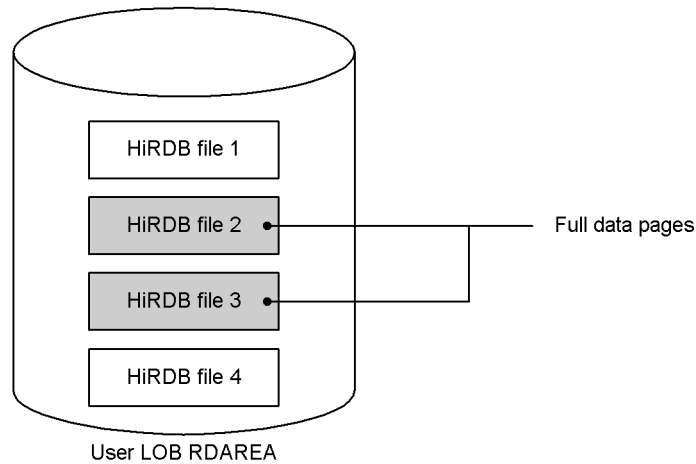
- Making a backup of user LOB RDAREAs using a method other than the database recovery utility (`pdcopy` command)
- Making a backup of HiRDB files on a file-by-file basis
- Performing operations that do not typically occur when updating or deleting registered LOB data (operations that typically occur when adding LOB data)
- Making a backup of a user LOB RDAREA consisting of multiple HiRDB files

For details about how to make backups using the frozen update command, see the *HiRDB Version 8 System Operation Guide*.

(1) Overview of the frozen update command

When you execute the frozen update command, HiRDB files with data pages that are entirely full (all pages are allocated) in a user LOB RDAREA are placed in *frozen update status*. Data in a HiRDB file that has been placed in frozen update status cannot be updated or deleted. Figure 7-6 provides an overview of frozen update command processing.

Figure 7-6: Overview of frozen update command processing

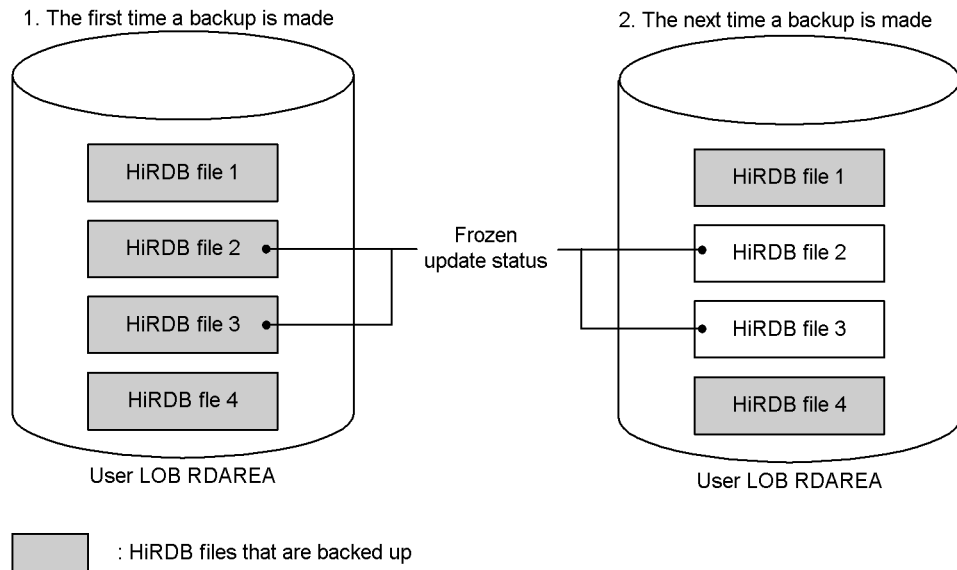
**Explanation**

- A user LOB RDAREA exists that contains HiRDB files 1 through 4. All data pages in HiRDB files 2 and 3 are full.
- Executing the frozen update command on this user LOB RDAREA places HiRDB files 2 and 3 into frozen update status. HiRDB files that are in frozen update status are indicated in the `KFPH27024-I` message.
- HiRDB files 1 and 4 are placed in permit update status.

(2) How to use the frozen update command to make backups

Figure 7-7 shows how to use the frozen backup command to make backups.

Figure 7-7: How to use the frozen update command to make backups



Explanation

The frozen update command is executed before the backup is made. As a result, HiRDB files 2 and 3 are placed in frozen update status.

1. The first time the backup is made, a backup is made of all HiRDB files (HiRDB files 1 through 4).
2. Because HiRDB files 2 and 3 are in frozen update status, their contents do not change after the first time the backup is made. Therefore, the next time a backup is made, HiRDB files 2 and 3 do not need to be backed up. Only HiRDB files 1 and 4 are backed up.

Remark

Because the first HiRDB file (HiRDB file 1 in Figure 7-7) contains a management record, even if the data area becomes full, the file is still written to on a regular basis. Consequently, backups are always made of the first HiRDB file.

7.2.6 NetBackup linkage facility

Using the NetBackup linkage facility allows you to create backup files for use by the database copy utility (`pdcopy`) or database recovery utility (`pdrstr`) on a medium managed by a NetBackup server. To use the NetBackup linkage facility, you need JP1/VERITAS NetBackup Agent for HiRDB License.

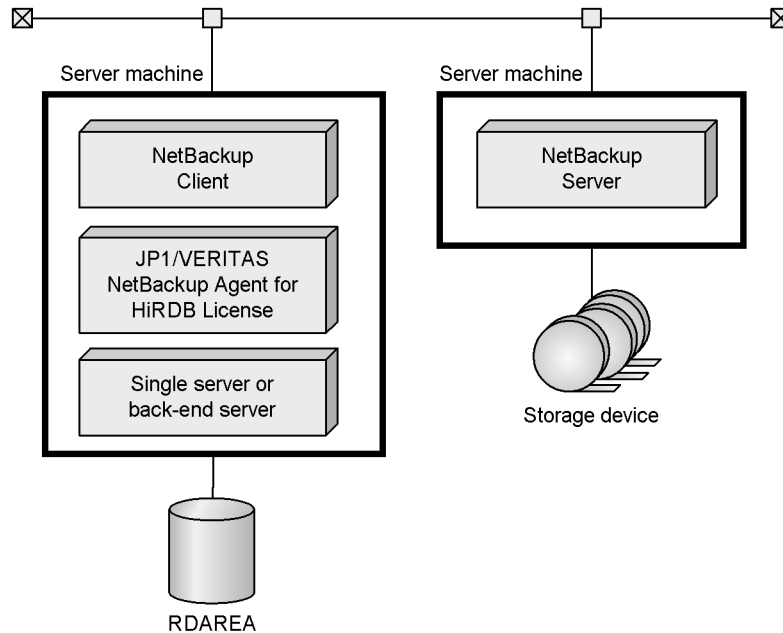
Note:

The NetBackup linkage facility cannot be used with the Windows Server 2003 (IPF) and Windows (x64) versions of HiRDB.

(1) System configuration example of the NetBackup linkage facility

By using the NetBackup linkage facility, you can create backup files on a medium that is managed by the NetBackup server. Figure 7-8 shows a system configuration example of the NetBackup linkage facility.

Figure 7-8: System configuration example of the NetBackup linkage facility



For details about the NetBackup linkage facility, and about the products required to use the NetBackup linkage facility, see *JP1/VERITAS NetBackup v4.5 Agent for HiRDB License Description and User's Guide*.

(2) Advantages of using the NetBackup linkage facility

The following describes the advantages of using the NetBackup linkage facility:

- Ability to manage backup files with NetBackup

Because NetBackup manages the output destination of backup files as well as the backup date and time, you do not need to change the name of the backup file when you make a backup. In addition, when you are recovering (restoring), and simply specifying the volume name and backup date and time, NetBackup selects the

correct backup file for you. Moreover, if you are using the most recent backup file, you do not even have to specify the backup date and time.

- Ability to create backup files on storage devices connected to server machines on which HiRDB is not installed

You can create backup files on a storage device that is connected to a server machine on which HiRDB is not installed. If you do not use the NetBackup linkage facility, you can only create backup files on storage devices connected to server machines on which HiRDB is installed.

- Ability to use a wide variety of media types for backup files

You can create backup files on any media that are supported by NetBackup. For details about the media that are supported by NetBackup, see the NetBackup documentation.

- Ability to use different operating systems for the HiRDB server and the NetBackup server

You can use the NetBackup linkage facility even if the operating systems on which the HiRDB server and the NetBackup server run differ.

7.3 Reorganizing tables and indexes

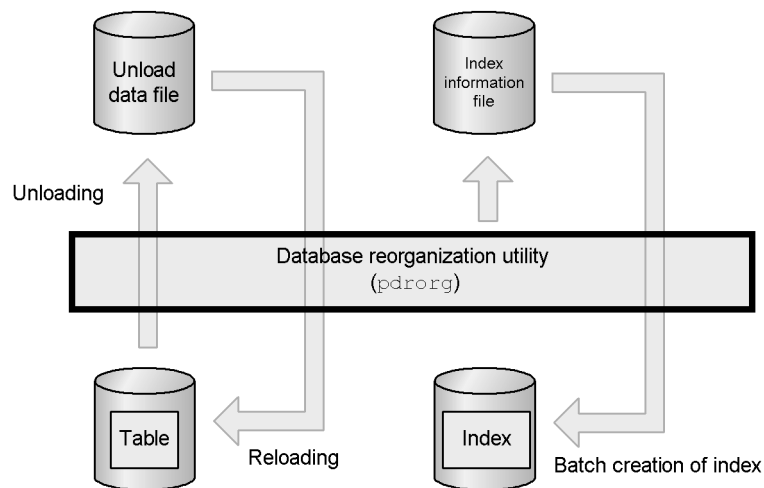
It is recommended that the table owner or the DBA privilege holder execute periodically the *database reorganization utility* (`pdroorg`) to reorganize tables and indexes.

7.3.1 Table reorganization

Deleting data does not release the segments and pages in which the deleted data has been stored. Therefore, when deletion of data has been performed many times, the amount of dead space in the table becomes significant, resulting in a decrease in data storage efficiency. Eventually, this can mean a shortage of space for RDAREAs, even though there has not been an increase in the amount of data.

Similarly, if data is added repeatedly, problems arise with respect to data not being stored in pages near the cluster key and to proliferation in the number of data I/O operations. The result is degradation of data retrieval performance. Executing the database reorganization utility (`pdroorg`) to reorganize a table causes the system to re-store the table's data, which can prevent these problems from arising. Figure 7-9 illustrates table reorganization.

Figure 7-9: Table reorganization



Explanation

- First, the table is stored temporarily in an unload file; this process is called *table unloading*. Subsequently, the data is re-stored in the table; this process is called *table data reloading*. The entire process is called *table reorganization*.

- If an index is defined for the table, the index information is output to an index information file when the data is reloaded. Based upon this information, HiRDB re-creates the index in the batch mode, which reorganizes the index as well.

(1) Execution units for table reorganization

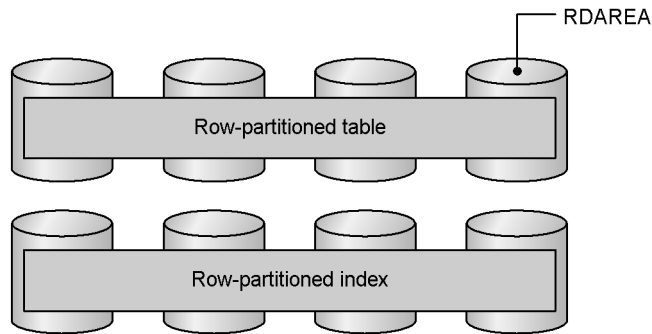
Table reorganization can be executed in the following units:

- By table
- By RDAREA
- By schema

(a) Reorganization by table

Reorganization processing can be performed on an entire table; this is the method that is usually used. You should first execute the database condition analysis utility to determine whether or not the entire table needs to be reorganized. If so, you can then execute reorganization of the entire table. Figure 7-10 illustrates reorganization of an entire table.

Figure 7-10: Reorganization of an entire table



Note

The data indicated by shading is subject to reorganization.

Explanation

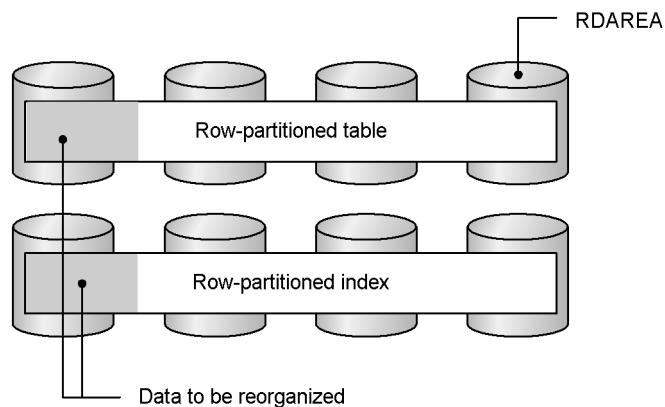
The table to be reorganized is specified in the `-t` option of the database reorganization utility.

(b) Reorganization by RDAREA

Reorganization processing is performed on a per-RDAREA basis. This method can be used only if the table is row-partitioned. Reorganization of RDAREAs is executed when the results of the database condition analysis utility indicate that it would suffice

to reorganize only a portion of a row-partitioned table. This reduces the processing time compared with reorganization of the entire table. Figure 7-11 illustrates reorganization of an RDAREA.

Figure 7-11: Reorganization of an RDAREA



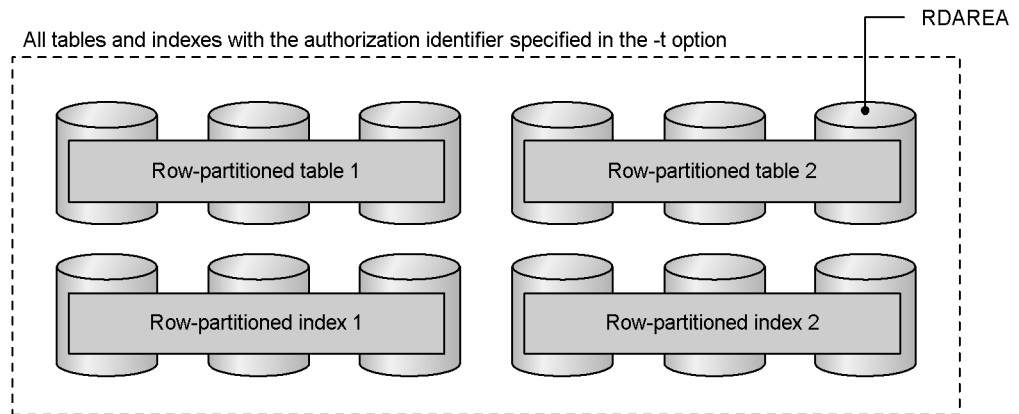
Explanation

The table to be reorganized is specified in the `-t` option and the RDAREAs to be reorganized are specified in the `-r` option of the database reorganization utility.

(c) Reorganization by schema

This processing reorganizes all tables in a schema in the batch mode. Reorganization by schema can be used when you wish to reorganize all the tables you own on a batch basis. Figure 7-12 illustrates reorganization of a schema.

Figure 7-12: Reorganization of a schema



Note: All data is reorganized.

Explanation

The authorization identifier of the schema to be reorganized is specified in the `-t` option of the database reorganization utility. The specification format is: `-t authorization-identifier.all`.

(2) Reorganizing a table containing a large quantity of data

When a table containing a large quantity of data is to be reorganized, you must consider whether or not *reorganization with synchronization points set* should be executed.

Normally, while a table is being reorganized, transactions cannot be reconciled until storage processing of all the data has been completed. This means that synchronization point dumps cannot be obtained during execution of the database reorganization utility. If HiRDB terminates abnormally during reorganization of a large quantity of data, it will take a long time to restart HiRDB. To resolve this problem, you can set synchronization points at intervals of any number of data items during storage of the data (reload processing) in order to reconcile transactions. This is called *reorganization with synchronization points set*.

To perform reorganization with *synchronization points set*, you must specify a *synchronization point lines count*, which is the number data items to be stored before a synchronization point is set. This value is specified in the `option` statement of the database reorganization utility.

Synchronization point setting can also be specified in the database load utility; this is called *data loading with synchronization points set*.

(3) Facility for predicting reorganization time

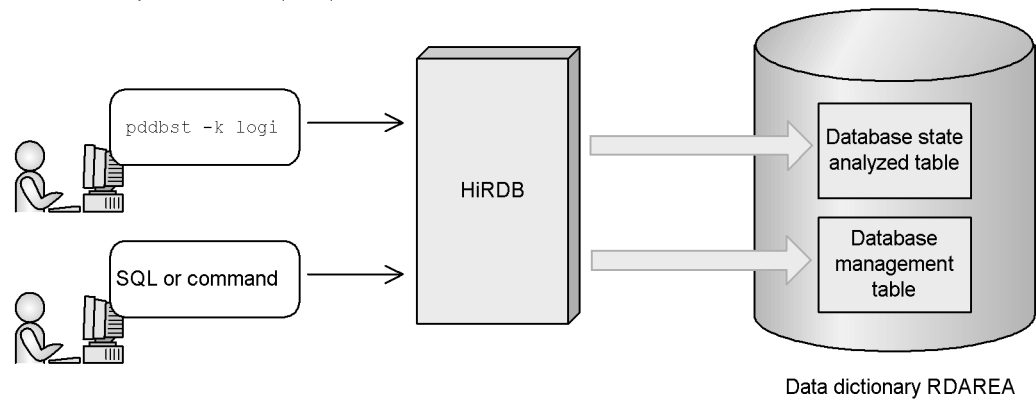
The decision on whether to reorganize tables or indexes or whether to extend an

RDAREA must be made by the user based on using messages that are output and the execution results of the `pddbst` command to make a comprehensive evaluation of which tables to reorganize and when to reorganize them. It is possible that the user may reorganize a table that does not need to be reorganized or may neglect to reorganize a table that does need reorganizing because of an overlooked message that was output.

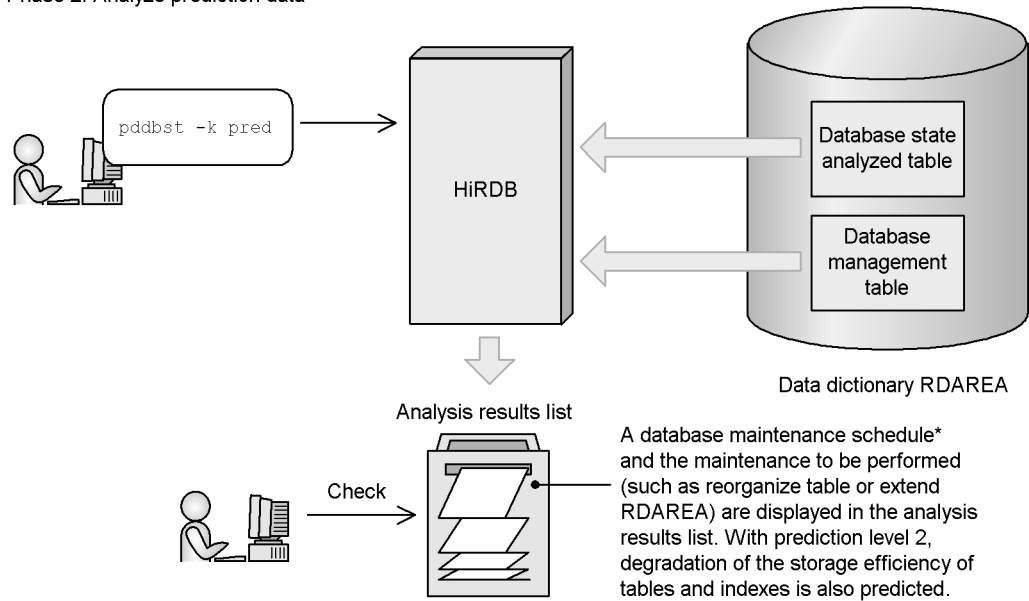
To simplify this operation, HiRDB is now able to predict when reorganization will be necessary. The function that performs this prediction is called the facility for predicting reorganization time. Figure 7-13 provides an overview of this facility.

Figure 7-13: Overview of facility for predicting reorganization time

Phase 1: Get prediction data (store)



Phase 2: Analyze prediction data



* A day on which RDAREA maintenance needs to be performed is called a *scheduled database maintenance day*.

Predicting when reorganization will be needed is divided into two phases:

- Phase 1: Get reorganization prediction data
 - The `pddbst` command is executed periodically, and analysis results from the

database are accumulated in the *database state analyzed table*.

- Whenever SQL code or a command is executed, an operation log from the database is output to the *database management table*.
- Phase 2: Analyze reorganization prediction data

Using the database state analyzed table and database management table as the input information, the `pddbst` command is used to analyze the reorganization prediction data. The user checks the execution results of the `pddbst` command and performs one of the following operations as needed:

 - Uses the `pdrorg` command to reorganize tables or indexes
 - Uses the `pdreclaim` command to release used free pages and free segments
 - Uses the `pdmod` command to extend an RDAREA
 - Uses the `pdmod` command to automatically extend an RDAREA
 - Uses the `pdmod` command to reinitialize an RDAREA.

The facility for predicting reorganization time also provides the two levels described below. Prediction level 1, which predicts the time required for `pddbst` to accumulate database analysis results, can be run in a relatively short amount of time. However, Prediction level 2 may require a much longer time to run.

- Prediction level 1

This level predicts when an RDAREA will no longer have sufficient free space. This level is designed for users who do not want to reorganize frequently while there is still sufficient free space for the RDAREA.

- Prediction level 2

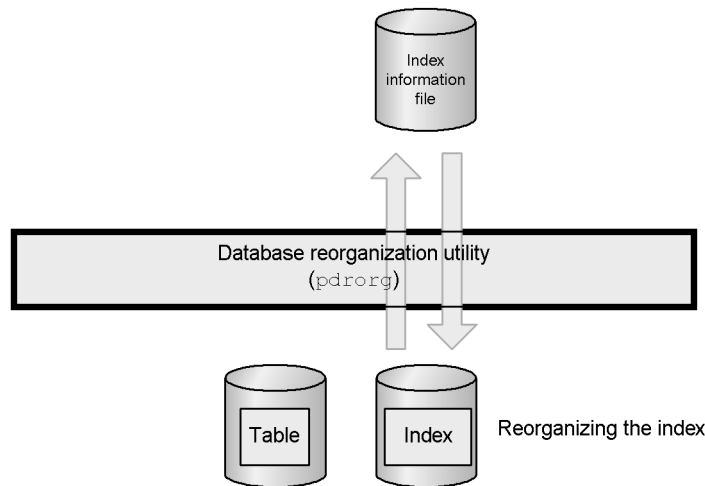
In addition to predicting when an RDAREA will no longer have sufficient free space, this level predicts degradation of storage efficiency in tables and indexes. This level is designed for users who want to reorganize as soon as table or index access performance is reduced, regardless of the amount of free space for the RDAREA.

For details about the facility for predicting reorganization time, see the *HiRDB Version 8 System Operation Guide* and the section about the database condition analysis utility in the manual *HiRDB Version 8 Command Reference*.

7.3.2 Index reorganization

It is possible to reorganize an index only. Figure 7-14 illustrates index reorganization of an index.

Figure 7-14: Index reorganization



Explanation

Index reorganization involves retrieving index key information, creating an index information file, and rearranging the index based on this information. Index reorganization can be executed by index or by RDAREAs that store an index.

(1) Applicability and application criteria

Index reorganization can be applied to normal indexes only; it cannot be used to reorganize plug-in indexes. Index reorganization should be executed in order to release dead space in index storage pages that has been generated by addition, deletion, or updating of large amounts of data.

(2) Index reorganization and its use

- When data updating (`UPDATE`) has been performed frequently, reorganization of the indexes only is recommended.
- When data deletion (`DELETE`) or addition (`INSERT`) has been performed frequently, reorganization of the table is recommended.
- If there is not enough time to reorganize a table, you can reorganize the index only in order to reduce index search time.

(3) Differences from re-creation of an index

Index re-creation involves a search of the table data; index reorganization does not involve a search of the table data. For this reason, index reorganization requires less processing time than re-creation,* sorting is not required, and processing performance is improved.

* Processing time is reduced if the following condition is satisfied:

Number of pages used in table storage RDAREA > Number of pages used in index storage RDAREA

(4) Notes on index reorganization

Before reorganizing simultaneously multiple indexes that are stored in the same RDAREA, you should use the `pdhold` command to place the RDAREA on shutdown status. When reorganization processing has been completed, you can use the `pdrels` command to release the RDAREA from shutdown status.

(5) Technique for reducing reorganization execution time

The processing time involved in index reorganization can be reduced by not collecting database update log information (setting either the no-log mode or the pre-update log acquisition mode in the `-l` option of the database reorganization utility (`pdrorg`)).

(6) Reorganizing an index in an RDAREA that has insufficient free space

The amount of unused area per page is specified in the `PCTFREE` operand of `CREATE TABLE` or `CREATE INDEX` when the index is reorganized is applied. However, if the index being reorganized is in an RDAREA that has insufficient free space to accommodate the specified percentage of unused area, the RDAREA can run out of space during index reorganization processing. To prevent this, you should specify the `idxfree` operand in the `option` statement of the database reorganization utility (`pdrorg`), and use the `PCTFREE` operand of `CREATE TABLE` or `CREATE INDEX` to change the percentage of unused area per page.

It must be noted that this is a temporary measure that is used during reorganization. In normal operations, the RDAREA should be expanded by executing the database structure modification utility (`pdmmod`).

7.4 Reusing used free pages and used free segments

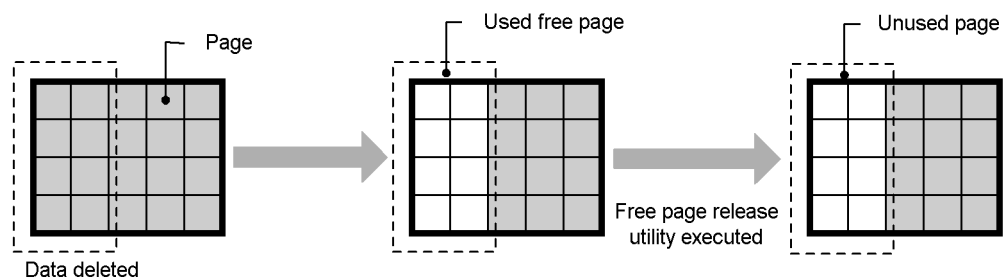
You can reuse used free pages in tables and indexes by converting them to unused pages. Similarly, you can reuse used free segments by converting them to unused segments. Before reading the descriptions in this section, however, you must be familiar with page statuses and segment statuses. For details about page statuses, see 4.3 *Page design*. For details about segment statuses, see 4.2 *Segment design*.

7.4.1 Reusing used free pages

(1) Releasing used free pages

When a large amount of table data is deleted by a batch job or some other operation, some of the pages in which that table data is being stored (data pages) may become used free pages. Similarly, when an index is defined, some of the pages in which index key values are being stored (index pages) also become used free pages. By executing the *free page release utility* (`pdreclaim`), you can convert used free pages to unused pages, and reuse them. This is called *releasing used free pages*. Figure 7-15 provides an overview of releasing used free pages.

Figure 7-15: Releasing used free pages



Hint:

- You cannot release used free pages of data stored in LOB RDAREAs.
- You cannot release used free pages of plug-in indexes.

For details about releasing used free pages, see the *HiRDB Version 8 System Operation Guide*.

(2) Benefits gained from releasing used free pages

(a) Benefits gained from releasing used free pages of tables

Table 7-4 lists and describes the benefits gained from releasing used free pages of a

table.

Table 7-4: Benefits gained from releasing used free pages of a table

Benefit	Description	Degree of benefit
Ability to increase the table reorganization cycle	The ability to reuse used free pages improves data storage efficiency. In turn, this can increase (improve) the table reorganization time cycle.	Good
Improvement in performance when searching large data sets	Because they are a type of used page, used free pages are searched. However, searching is not performed on unused pages (they are skipped by the search). Thus, converting to unused pages improves the search performance in direct proportion to the ratio converted. The benefits are particularly evident when large data sets are searched.	Varies
Improvement in performance when INSERT and UPDATE are used.	If sufficient contiguous free space cannot be allocated when an attempt is made to save data to a used page, HiRDB performs a process called <i>page compaction</i> . Page compaction refers to a process whereby data in the affected page is repacked to secure sufficient contiguous free space in which to store the new data. When free used pages are released, page compaction is also performed at the same time. This eliminates the need to perform page compaction — which prolongs INSERT and DELETE processing — and improves performance by the corresponding amount. The pages on which page compaction is performed are used pages other than full pages and used free pages.	Varies
Potential reduction in errors when INSERT and UPDATE are performed on branch rows.	If you execute INSERT or UPDATE on a branch row when there are no unused pages, an error occurs (KFPA11756-E message). The increase in unused pages due to the release of used free pages tends to reduce the frequency of this error.	Varies

Legend:

Good: Always beneficial.

Varies: The degree of benefit varies depending on conditions.

(b) Benefits gained from releasing used free pages of indexes

Table 7-5 lists and describes the benefits gained from releasing used free space of an index.

Table 7-5: Benefits gained from releasing used free space of an index

Benefit	Description	Degree of benefit
Potential reduction of insufficient capacity in RDAREAs storing indexes	If capacity runs out even though free pages (used free pages) exist, release the used free pages. Note that this tends to reduce the occurrence of insufficient capacity in RDAREAs that store indexes with key values that are frequently updated or deleted.	Great
Ability to increase the index reorganization cycle	The ability to reuse used free pages improves data storage efficiency. In turn, this can increase (improve) the index reorganization time cycle.	Good
Improvement in performance when searching large data sets that use indexes	Because they are a type of used page, used free pages are searched. However, searching is not performed on unused pages (they are skipped by the search). Thus, converting to unused pages improves the search performance in direct proportion to the ratio converted. The benefits are particularly evident when large data sets are searched.	Varies

Legend:

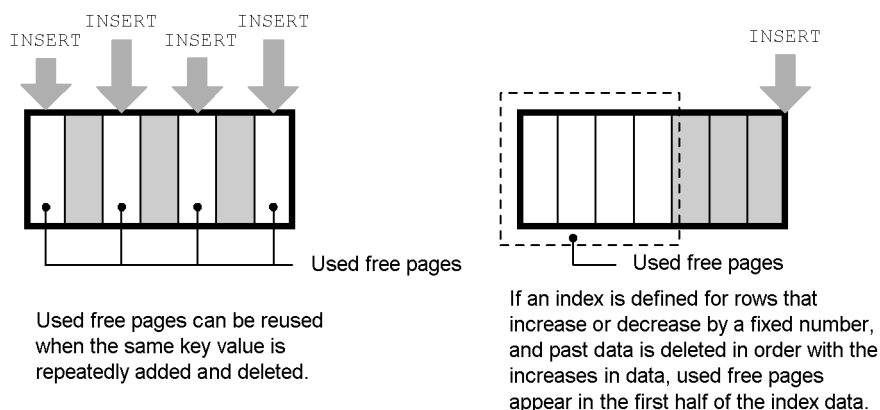
Great: Particularly beneficial.

Good: Always beneficial.

Varies: The degree of benefit varies depending on conditions.

Using this utility is particularly beneficial when deleted key values are not re-registered. Because used free pages are reused when the same key value is added or deleted repeatedly, large numbers of used free pages do not appear. However, for indexes that are defined for rows that increase or decrease by a fixed number (such as date, sequence number, and so on), and if past data is deleted in order with the increases in data, a large number of used free pages that are not reused appear in the first half of the index pages. Figure 7-16 shows the processing of a used free page being created for index pages.

Figure 7-16: Process of used free pages being created for index pages



Note that, after the used free pages are released, the key values are stored in the released pages, which improves data storage efficiency.

(3) Differences from table or index reorganization

From the standpoint of performance and data storage efficiency, reorganizing tables or indexes is superior to releasing used free pages. However, while releasing used free pages, you can still access the tables or indexes on which the utility is operating. With reorganization, you cannot access the tables or indexes on which the utility is operating. This means that you do not need to interrupt normal operations when releasing free pages.

Use the execution results of the database condition analysis utility to evaluate whether to reorganize tables or indexes, or to release used free pages. The following lists the evaluation criteria:

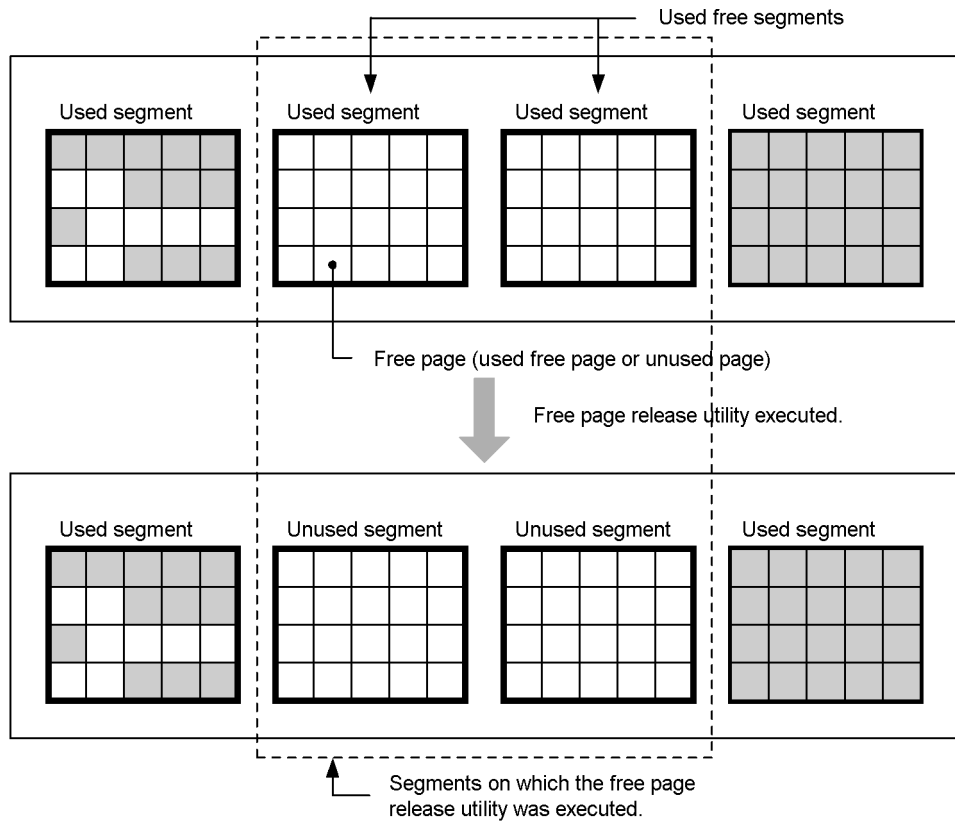
- If there is a large number of used free pages, release the used free pages.
- If the page utilization ratio of the used pages differs significantly from the segment free page ratio (value of the `PCTFREE` operand of `CREATE TABLE`), perform reorganization.

7.4.2 Reusing used free segments

(1) Releasing used free segments

By executing the free page release utility, you can convert used free segments to unused segments, and reuse them. This is called *releasing used free segments*. Figure 7-17 provides an overview of releasing used free segments.

Figure 7-17: Releasing used free segments



For details about releasing used free segments, see the *HiRDB Version 8 System Operation Guide*.

(2) Benefits gained from releasing used free segments

Once segments are allocated, only the table (or index) that is assigned to use a particular segment can use it; no other table can do so. Releasing used free segments converts used free segments to unused segments, which allows other tables to use them.

7.5 Adding, expanding, and moving RDAREAs

As the scale of operations increases, the amount of data in your databases increases as well. However, even if a database becomes larger than your original estimate, with HiRDB, you can subsequently add, expand, or move the database. With HiRDB/Parallel Server, you can also add a server machine and add or move a database onto the new server machine.

HiRDB provides the following functions to support increases in the amount of data in your databases:

- Adding RDAREAs
- Expanding RDAREAs
- RDAREA automatic extension
- Moving RDAREAs (HiRDB/Parallel Server only)

7.5.1 Adding RDAREAs

RDAREAs can be added with the `create rdarea` statement of the database structure modification utility (`pdmod`). RDAREAs should be added when a new table is created.

To use a newly added RDAREA, it is necessary to allocate a global buffer to it. For this purpose, you should use the `pdbuf1s` command to check for an available (previously defined) global buffer.

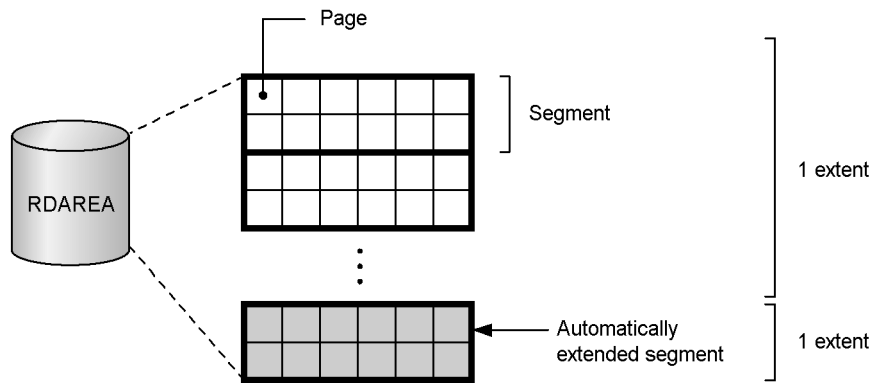
7.5.2 Expanding an RDAREA

As more and more data is added to a table, the available space in its RDAREAs becomes smaller and smaller. When the amount of remaining space becomes too small, it is possible to expand RDAREAs with the `expand rdarea` statement of the *database structure modification utility* (`pdmod`).

7.5.3 RDAREA automatic extension

When a space shortage occurs in an RDAREA, the size of the RDAREA can be expanded by means of automatic addition of segments, provided that there is sufficient free space in the HiRDB file system area. This process is called *RDAREA automatic extension*. Figure 7-18 illustrates RDAREA automatic extension.

Figure 7-18: RDAREA automatic extension



A set of contiguous areas in a HiRDB file system area is called an *extent*. The maximum number of extents for a single HiRDB file is 24. With automatic extension, the number of extents does not increase if contiguous free space can be secured in the last extent allocated in the target HiRDB file. However, if non-contiguous free space is allocated, the number of extents is increased. Extent information can be checked with the `pdf1s` command. Fragmentation of free space in a HiRDB file system area can occur because previously allocated extents may be deleted or their sizes may be reduced when an RDAREA is deleted, reinitialized (its allocation size is reduced or the `reconstruction` operand is specified), or integrated. Caution must be exercised if you add, extend, or reinitialize an RDAREA while it is in this status, because multiple extents may be allocated to a single HiRDB file regardless of whether or not automatic extension is being performed.

(1) RDAREAs eligible for automatic extension

Automatic extension can be applied to the following RDAREAs:

- Data dictionary RDAREAs
- User RDAREAs
- Registry RDAREA
- Data dictionary LOB RDAREAs
- User LOB RDAREAs
- Registry LOB RDAREA

(2) Setting automatic extension

Following is the procedure for setting automatic extension:

1. When the HiRDB file system area is created with the `pdfmkfs` command, specify the maximum number of extensions (in the `-e` option).

- When an RDAREA is created, use a utility control statement* to specify use of the automatic extension facility.

* This can be specified in the `create rdarea`, `expand rdarea`, `initialize rdarea`, or `alter rdarea` statement of the database initialization utility, the database structure modification utility, or the registry facility initialization utility.

For details about RDAREA automatic extension, see the *HiRDB Version 8 System Operation Guide*.

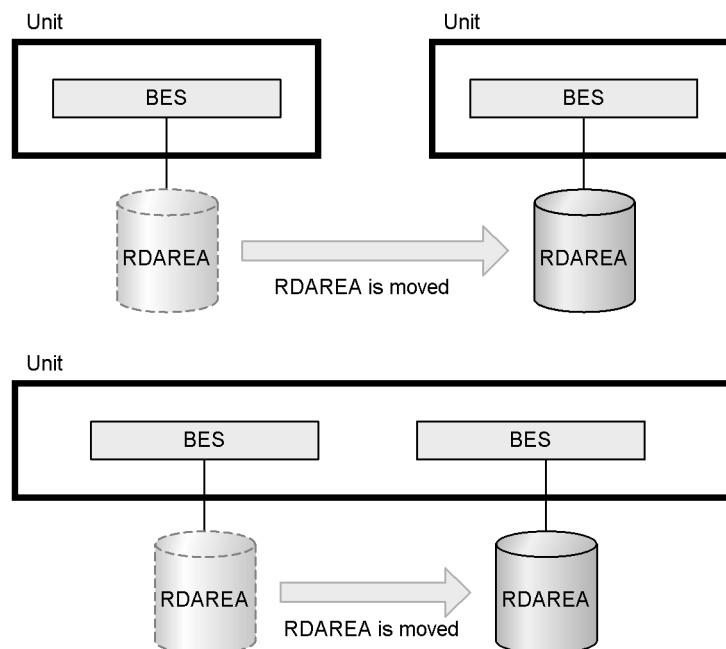
7.5.4 Moving RDAREAs (HiRDB/Parallel Server only)

You can use the `move rdarea` statement of the database structure modification utility (`pdmod` command) to move an RDAREA to another back-end server. The capability of moving an RDAREA is provided only by HiRDB/Parallel Server. The following lists the RDAREAs that can be moved:

- User RDAREAs
- User LOB RDAREAs

Figure 7-19 provides an overview of moving RDAREAs.

Figure 7-19: Moving RDAREAs



For details about moving RDAREAs, see the *HiRDB Version 8 System Operation Guide*.

7.6 Space conversion facility

In a data comparison, one double-byte space character and two single-byte space characters will be recognized as different data. Therefore, table data containing a mixture of double-byte and single-byte space characters can produce incorrect retrieval results.

Example

The following data are recognized as being different:

TV		21-inch
TV		21-inch

where

 : indicates two single-byte spaces

 : indicates one double-byte spaces

The space conversion facility enables double-byte space characters and single-byte space characters to be intermixed in table data.

The double-byte space character being discussed here is coded as shown as follows. Two single-byte space characters are coded as X'2020'.

- Shift-JIS Kanji Code: X'8140'
- Unicode (UTF-8)*: X'E38080'

* NCHAR and NVARCHAR cannot be used if the character encoding is Unicode (UTF-8).

(1) Space conversion levels

As shown in Table 7-6, three levels of space character conversion are provided by the space conversion facility.

Table 7-6: Space conversion levels

Level	Explanation
Level 0	No space conversion.

Level	Explanation
Level 1	<p>Converts as follows data spaces that occur in literals, embedded variables, and ? parameters in the data manipulation SQL or data spaces that are stored by utilities:</p> <ul style="list-style-type: none"> • When a character string literal is being handled as a national character string literal, two single-byte space characters in succession are converted into one double-byte space character; in the case of a single occurrence of a single-byte space, no conversion is performed. • When a character string literal is being handled as a mixed character string literal, one double-byte space character is converted into two single-byte space characters. • During storage of data in a national character string-type column or during comparison of data with a national character string-type value expression, two single-byte space characters in succession in an embedded variable or ? parameter are converted into one double-byte space character; in the case of a single occurrence of a single-byte space, no conversion is performed. • During storage of data in a mixed character string-type column or during comparison of data with a mixed character string-type value expression, one double-byte space character is converted into two single-byte space characters.
Level 3	<p>Adds the following processing to the processing of space conversion level 1:</p> <ul style="list-style-type: none"> • During retrieval of data in a national character string-type value expression, one double-byte space character is converted into two single-byte space characters.

Figure 7-20 illustrates Level 1 processing. Figure 7-21 illustrates Level 3 processing.

Figure 7-20: Level 1 processing

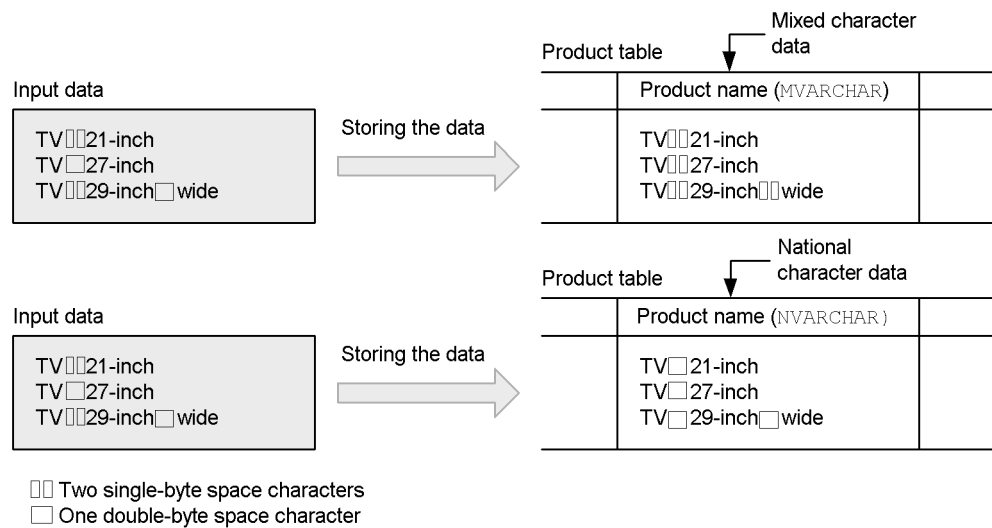
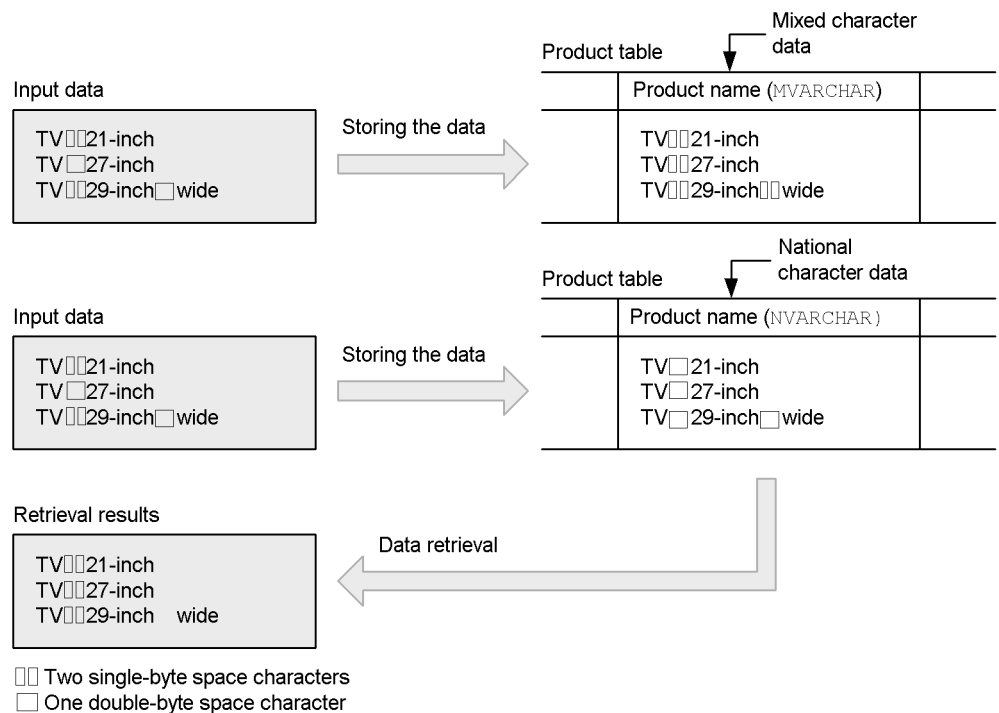


Figure 7-21: Level 3 processing



(2) Setting the space conversion level

The desired space conversion level can be specified in the following operands:

- `pd_space_level` operand in the system common definition
- `PDSPELVL` operand in the client environment definition
- `spacelvl` operand in the `option` statement of the database load utility (`pdload`)
- `spacelvl` operand in the `option` statement of the database reorganization utility (`pdrorg`)

For details about using the space conversion facility, see the *HiRDB Version 8 System Operation Guide*.

7.7 Facility for conversion to a decimal signed normalized number

The decimal, date interval, and time interval data formats are signed packed formats consisting of a value's integer and sign parts. Normally, HiRDB handles X'C' (positive), X'D' (negative), and X'F' (positive) as the sign parts of signed packed-format data as valid values, and stores directly in the database any signs that are input from a UAP or utility.* In addition, HiRDB treats +0 (sign part X'C' or X'F') and -0 (sign part X'D') as distinct values.

Use of the facility for conversion to a decimal signed normalized number enables conversion of the sign part of a signed packed decimal, date interval, or time interval data format.

* Type conversions and operations during the execution of SQL statements sometimes result in conversion of signs. In addition, the use of multicolumn indexes can result in conversion of signs.

(1) Specification of the sign part of signed packed format

In HiRDB, specification of the sign part of signed packed format is as listed in Table 7-7.

Table 7-7: Specification of the sign part of signed packed format

Sign part	Meaning
X'C'	Indicates a positive value.
X'D'	Indicates a negative value.
X'F'	Indicates a positive value.

(2) Rules on conversion of the sign part of signed packed format

When the facility for conversion to a decimal signed normalized number is used, HiRDB converts during data input the sign part of signed packed format according to the rules in Tables 7-8 and 7-9. The process of converting a sign part is called *normalization*. When the sign part is converted, the codes +0 and -0 can be treated as being the same value.

Table 7-8: Rules for conversion of the sign part of signed packed format (other than "0" data)

Sign part of embedded variable	Not normalized	Normalized
X'A'	Error	Converts to X'C'.
X'B'	Error	Converts to X'D'.

Sign part of embedded variable	Not normalized	Normalized
X'C'	No conversion	No conversion
X'D'	No conversion	No conversion
X'E'	Error	Converts to X'C'.
X'F'	No conversion	Converts to X'C'.
X'0' - X'9'	Error	Error

Table 7-9: Rules for conversion of the sign part of signed packed format ("0" data)

Sign part of "0" data	Not normalized	Normalized
X'A'	Error	Converts to X'C'.
X'B'	Error	
X'C'	No conversion	
X'D'	No conversion	
X'E'	Error	
X'F'	No conversion	

(3) Application criteria

When UAPs with different sign part specifications are used, it may be beneficial to use the facility for conversion to a decimal signed normalized number. In such a case, this facility should be used after the sign conversion rules are adequately confirmed.

For example, if a UAP is transferred from XDM/RD E2 to HiRDB, it may be beneficial in some situations to use the facility for conversion to a decimal signed normalized number, because XDM/RD E2 and HiRDB have different specifications for the sign part of the decimal type.

(4) Environment setup

To use the facility for conversion to a decimal signed normalized number, you must specify `pd_dec_sign_normalize=Y` in the system common definition.

The facility for conversion to a decimal signed normalized number should be specified as soon as HiRDB is installed. If sign parts are to be normalized once operation of HiRDB is underway, it may be necessary to reload all table data in which the decimal type is defined.

For details about using the facility for conversion to a decimal signed normalized

number, see the *HiRDB Version 8 System Operation Guide*.

Chapter

8. Error-handling Facilities

This chapter describes error-handling facilities.

8.1 System switchover facility

8.2 Recovery-unnecessary front-end servers

8.1 System switchover facility

Linking to a cluster software product allows you to use the *system switchover facility*, which is designed to improve system reliability and availability. This section provides an overview of the system switchover facility. For details about how to operate the system switchover facility, see the *HiRDB Version 8 System Operation Guide*.

Note that, to use the system switchover facility, you must use Microsoft Cluster Server (MSCS) as your cluster software.

8.1.1 Overview of the system switchover facility

The system switchover facility includes the standby system switchover facility and the standby-less system switchover facility.

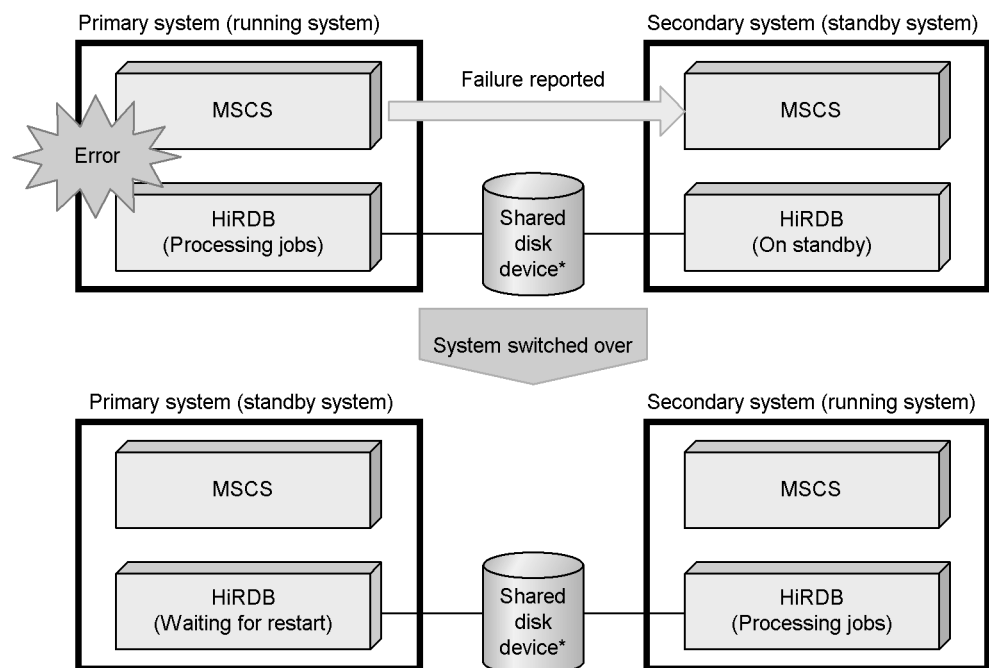
(1) Overview of the standby system switchover facility

By deploying a standby HiRDB separate from the HiRDB that is actively processing jobs, if a failure occurs on the server machine or on HiRDB, job processing can be automatically switched over to the standby HiRDB. This ability is called the *system switchover facility (standby system switchover facility)*. Job processing is interrupted from the time the failure occurs to the time processing is switched over to the standby HiRDB. The system switchover facility is used to keep system downtime to a minimum when a failure occurs.

You implement the system switchover facility with a cluster system configuration consisting of multiple server machines. For HiRDB/Single Server, the system is switched over on a per-system basis. For HiRDB/Parallel Server, the system is switched over on a per-unit basis.

The system on which jobs are currently being processed is called the *running system*, and the system that is currently in reserve is called the *standby system*. Whenever a system switchover occurs, the running system and the standby system are swapped. In addition, to distinguish between the two systems while you are building the systems and configuring the environments, the system that is initially started as the running system is called the *primary system*, and the system that is first started as the standby system is called the *secondary system*. Although the running system and the standby system change when a system switchover occurs, the primary system and secondary system do not. Figure 8-1 provides an overview of the system switchover facility (standby system switchover facility).

Figure 8-1: Overview of the system switchover facility (standby system switchover facility)



* For details about shared disk unit, see (3) *Shared disk unit*, as follows.

Explanation

If a failure occurs on the running system while it is processing jobs, the failure occurrence is reported to the standby system, and the system is switched over. The standby system becomes the running system, and resumes job processing.

(2) Overview of the standby-less system switchover facility

The system switchover facility includes the previously described standby system switchover facility and the standby-less system switchover facility. The standby-less system switchover facility is further classified as follows:

- Standby-less system switchover (1:1) facility
- Standby-less system switchover (effects distributed) facility

The standby-less system switchover facility can be used in a back-end server unit of a HiRDB/Parallel Server; it cannot be used in a unit in which a server other than a back-end server resides.

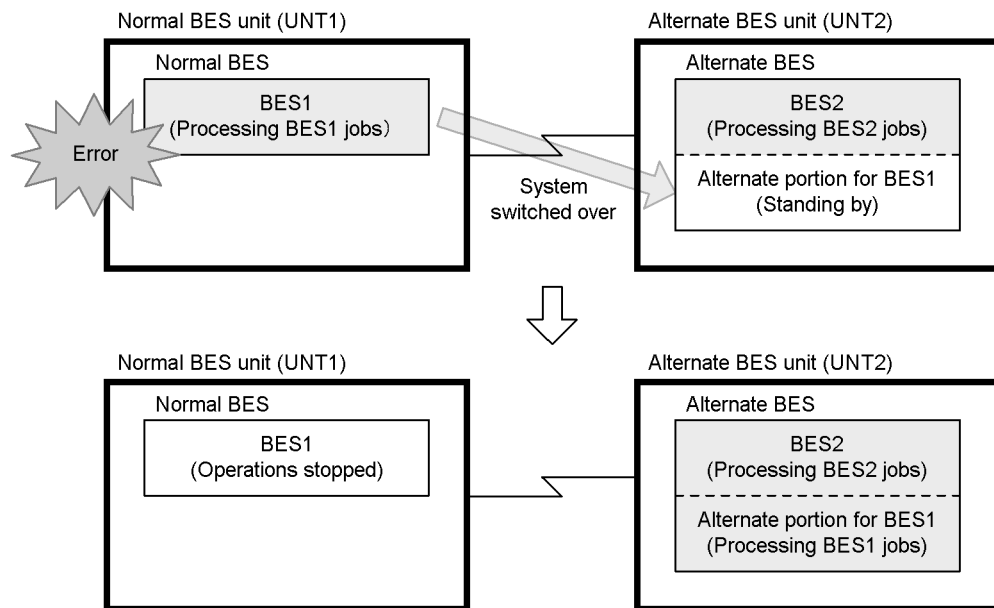
Unlike the standby system switchover facility, with the standby-less system switchover facility you do not have to allocate a standby unit. When a failure occurs, the system does not switch over to a standby unit; instead, the system switches over to another unit whose currently running back-end server takes over the back-end server processing of the failed unit. This is called the *standby-less system switchover facility*.

(a) Standby-less system switchover (1:1) facility

With the *standby-less system switchover (1:1) facility*, there is a one-to-one relationship between the unit on which the failure occurs and the unit to whose back-end server processing is switched.

A back-end server whose processing is transferred to another unit when a failure occurs is called a *normal BES*, and a back-end server that takes over processing is called an *alternate BES*. Similarly, the unit containing the normal BES is called the *normal BES unit*, and the unit containing the alternate BES is called the *alternate BES unit*. Figure 8-2 provides an overview of the standby-less system switchover (1:1) facility.

Figure 8-2: Overview of the standby-less system switchover (1:1) facility



Explanation

- Normally, both BES1 and BES2 perform processing.
- If a failure occurs on the normal BES unit (UNT1), the system switches over, and processing is taken over by the alternate BES. The area in which

processing is taken over is called the *alternate portion* and, when the alternate portion is performing processing, it is said to be *alternating*.

- After the failure is resolved and the normal BES unit is started, the processing taken over by the alternate BES is switched over to the normal BES, and returned to normal status. This is called *reactivating the system*.

Remarks

Using the concepts of the primary and other systems in the standby system switchover facility, consider the following with respect to the standby-less system switchover (1:1) facility:

- Think of the normal BES unit as the primary system, and the alternate BES unit as the secondary system.
- Under normal conditions, think of the normal BES unit as the running system, and the alternate portion as the standby system. During alternating, think of the alternate portion as the running system, and the normal BES unit as the standby system.

Prerequisites

To use the standby-less system switchover (1:1) facility, all of the following must be satisfied:

- HiRDB Advanced High Availability is installed.
- Hitachi HA Toolkit Extension is installed.
- The system switchover facility is running in server mode.

Advantages of the standby-less system switchover facility

The following describes the advantages of the standby-less system switchover facility over the standby system switchover facility:

- You do not need to set aside a standby system unit, which means you can use system resources more efficiently. However, remember that the load increases on the back-end server that takes over processing when the system is switched over, which may adversely affect processing performance.
- The server processes and system servers are already running, which allows you to reduce system switchover time to be about the same as when the rapid system switchover facility is used. For details about the rapid system switchover facility, see *8.1.5 Functions that reduce system switchover time (user server hot standby and the rapid system switchover facility)*.

(b) Standby-less system switchover (effects distributed) facility

When a failure occurs, processing requests directed to back-end servers in the failed unit can be distributed to and executed in multiple active units. This is called the

standby-less system switchover (effects distributed) facility. This facility enables you to use your system resources more efficiently, without having to allocate a standby server machine or a standby unit. Of course, there may be adverse effects on transaction processing performance because of the increased processing load on the units that have taken over processing for the servers in the failed unit. However, because the processing requests directed to the failed servers are distributed to and executed in a number of units, the workload increase for each unit is minimized, reducing overall degradation of system performance.

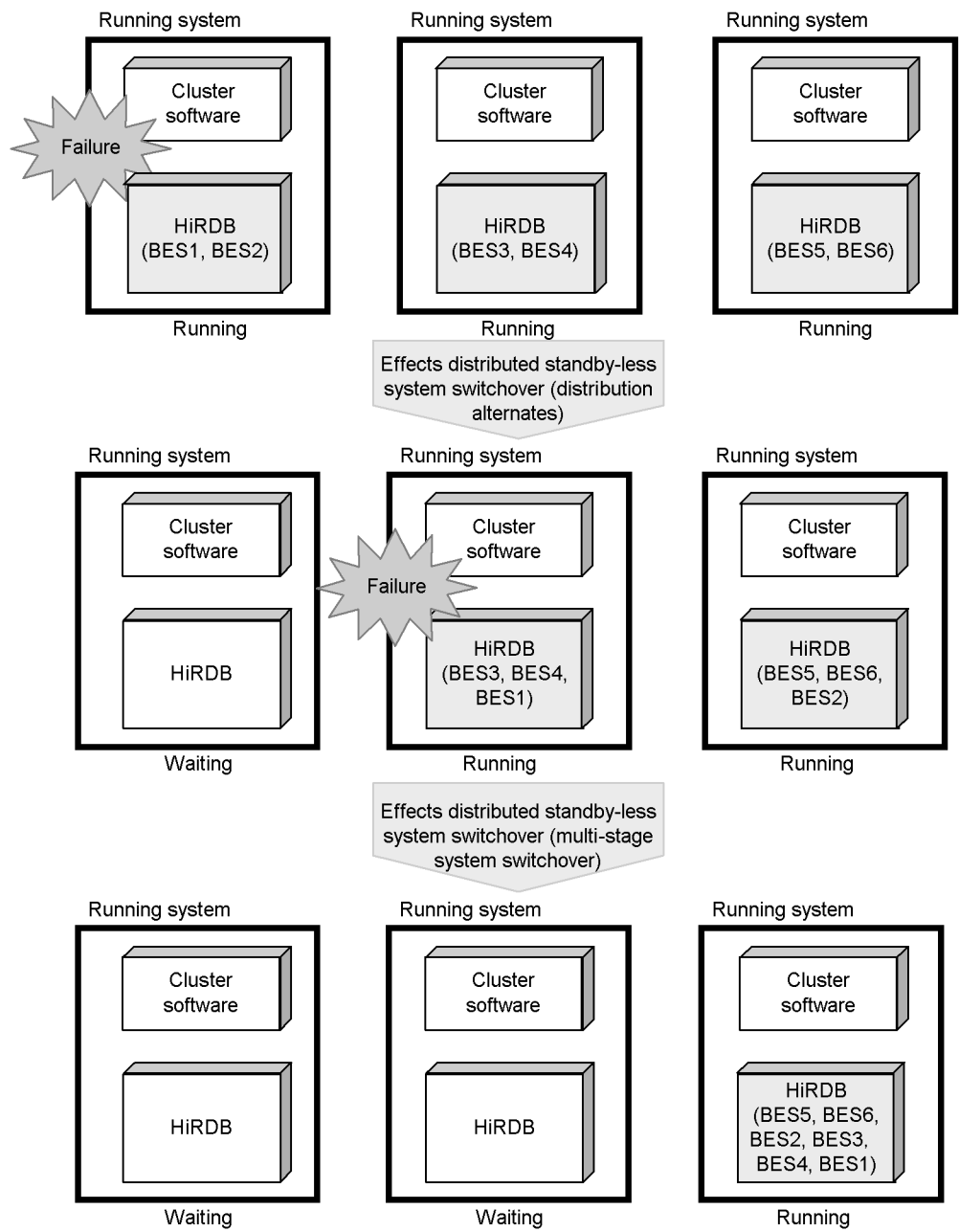
The standby-less system switchover (effects distributed) facility distributes the workload to and switches over among multiple back-end servers. The workload can also be distributed among multiple units. If another failure occurs, this time on a unit that was a switchover destination from the previous error, processing can be continued by again switching to a running unit (this is called *multi-stage system switchover*). Multi-stage system switchover cannot be performed with the standby-less system switchover (1:1) facility, so if a failure occurs at a switchover destination under that facility, processing for the failed unit cannot be continued.

It is appropriate to use the standby-less system switchover (effects distributed) facility in a system in which system resources must always be used at high efficiency and for which degradation of system performance must be minimized.

With the standby-less system switchover (effects distributed) facility, a back-end server that relinquishes processing when a failure occurs is called a *host BES*, and a back-end server that takes over processing is called a *guest BES*. The unit containing the host BESs is called the *regular unit*, and a unit containing a guest BES is called an *accepting unit*. All accepting units must be pre-defined as an *HA group*. The resources for back-end servers associated with guest BESs are called *guest areas*.

Figure 8-3 provides an overview of the standby-less system switchover (effects distributed) facility (with distribution alternates and multi-stage system switchover).

Figure 8-3: Overview of standby-less system switchover (effects distributed) facility (with distribution alternates and multi-stage system switchover)



Prerequisites

To use the standby-less system switchover (effects distributed) facility, the following conditions must be satisfied:

- HiRDB Advanced High Availability is installed
- The standby-less system switchover (effects distributed) facility can switch only to a unit dedicated to back-end servers (a unit that consists only of back-end servers).
- A unit that uses the standby-less system switchover (effects distributed) facility must consist of one or more back-end servers for the primary system. It cannot be used as a dedicated accepting unit.

(3) Shared disk unit

System switchover requires that there be an external hard disk that is shared by the primary and secondary systems. This hard disk is called the *shared disk unit*; it is used to transfer information from the running system to the standby system when system switchover occurs. The following HiRDB files must be created on the shared disk unit:

- HiRDB files that comprise RDAREAs
- System files (system log files, synchronization point dump files, status files)
- Back-up files
- Unload log files
- HiRDB file system area for audit trail files (if the security audit facility is being used)

8.1.2 Monitor mode and server mode**(1) Functional difference between the monitor mode and the server mode**

You can operate the system switchover facility in either the *monitor mode* or the *server mode*. In the monitor mode, only system failures are monitored. In the server mode, both system failures and server failures are monitored. In addition, system switchover can take less time in the server mode than in the monitor mode. Table 8-1 indicates the functional differences between the monitor mode and the server mode.

Table 8-1: Functional differences between the monitor mode and the server mode

Item or function		Monitor mode	Server mode
Monitored failure	System failure ¹	Y	Y
	Server failure ²	N	Y

Item or function		Monitor mode	Server mode
Functions provided to system switchover time	User server hot standby ³	N	Y
	Rapid system switchover facility ³	N	Y
Standby-less system switchover facility	Standby-less system switchover (1:1) facility	N	Y
	Standby-less system switchover (effects distributed) facility	N	Y

Legend:

Y: Item is monitored, or the function can be used.

N: Item is not monitored, or the function cannot be used.

¹ In this table, the following failures are assumed to be system failures; however, system failure conditions differ depending on the cluster software used. We recommend you to check your cluster software documentation to verify.

- Hardware failure
- OS failure
- Power outage
- Cluster software failure
- System slowdown

² In this table, the following failures are assumed to be server failures; however, server failure conditions differ depending on the cluster software used. We recommend you to check your cluster software documentation to verify.

- Abnormal termination of HiRDB (or unit if HiRDB/Parallel Server)
- Slowdown of HiRDB (or unit if HiRDB/Parallel Server)
- Database path errors

³ Functions that reduce the system switchover time. For details about user server hot standby and the rapid system switchover facility, see *8.1.5 Functions that reduce system switchover time (user server hot standby and the rapid system switchover facility)*.

(2) Products needed to operate in the server mode

Table 8-2 lists the products you need to operate the system switchover facility in the server mode.

Table 8-2: Products needed to operate in the server mode

Function	HiRDB Advanced High Availability	Hitachi HA Toolkit Extension
Server mode	—	Y
User server hot standby	—	Y
Rapid system switchover facility	—	Y
Standby-less system switchover (1:1) facility	Y	Y
Standby-less system switchover (effects distributed) facility	Y	Y

Legend:

Y: The indicated product is needed to use the facility.

— : Not required.

As of the publication date, Hitachi HA Toolkit Extension runs on Windows 2000 Advanced Server only, which means that this functionality can be used only in a Windows 2000 Advanced Server environment.

8.1.3 System switchover facility configurations

The system switchover facility provides the following three switchover modes:

- Automatic system switchover

The running system is switched over automatically when a failure occurs on it.

- Planned system switchover

The cluster administrator moves a HiRDB group to switch over the system intentionally.

- Grouped system switchover

This is the system switchover mode used when HiRDB is linked to another product, such as an OLTP product. When a failure occurs on the running system, the OLTP product and HiRDB (unit) are grouped, and the grouped system is switched over. With grouped system switchover, you can use either automatic system switchover or planned system switchover. To determine whether you can use grouped system switchover, check your cluster software documentation.

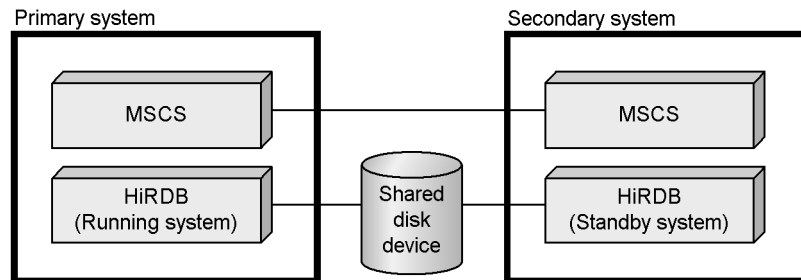
8.1.4 System configuration examples

This subsection provides system configuration examples when the system switchover facility is being used.

(1) System configuration examples of the standby system switchover facility**(a) One-to-one system switchover configuration**

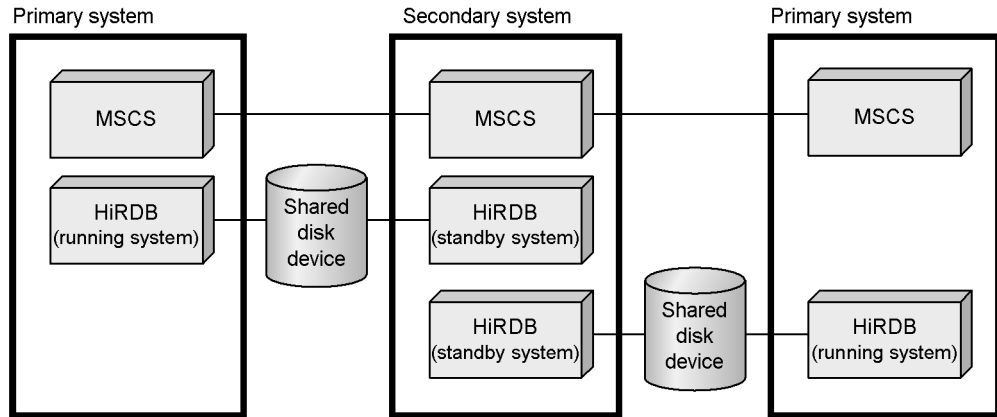
This configuration provides a one-to-one correspondence between running systems and standby systems. Use this configuration when you wish to guarantee response times, even when a system has been switched over. However, you cannot use the resources on the standby server machines (one set of server machine resources cannot be used for every two server machines). Figure 8-4 illustrates a one-to-one system switchover configuration.

Figure 8-4: One-to-one system switchover configuration

**(b) Two-to-one system switchover configuration**

This configuration provides a two-to-one correspondence between running systems and standby systems. The secondary system is configured as a multi-HiRDB system. Use this configuration with operations for which you wish to guarantee response times, even when a system has been switched over (response times deteriorate, though, if the two running systems have both been switched over). However, you cannot use the resources on the standby server machines (one set of server machine resources cannot be used for every three server machines). Figure 8-5 illustrates a two-to-one system switchover configuration.

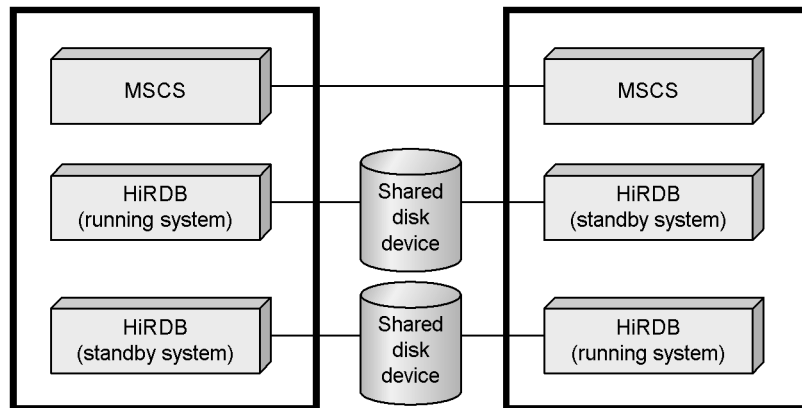
Figure 8-5: Two-to-one system switchover configuration



(c) Mutual system switchover configuration

This configuration provides an alternating standby system on each server machine while, at the same time, the server machine is operating as a running system. Every server machine is configured as a multi-HiRDB system consisting of a HiRDB running system and a HiRDB standby system. Use this configuration when you wish to utilize server machine resources most efficiently. However, response times deteriorate when a system has been switched over. Figure 8-6 illustrates a mutual system switchover configuration.

Figure 8-6: Mutual system switchover configuration



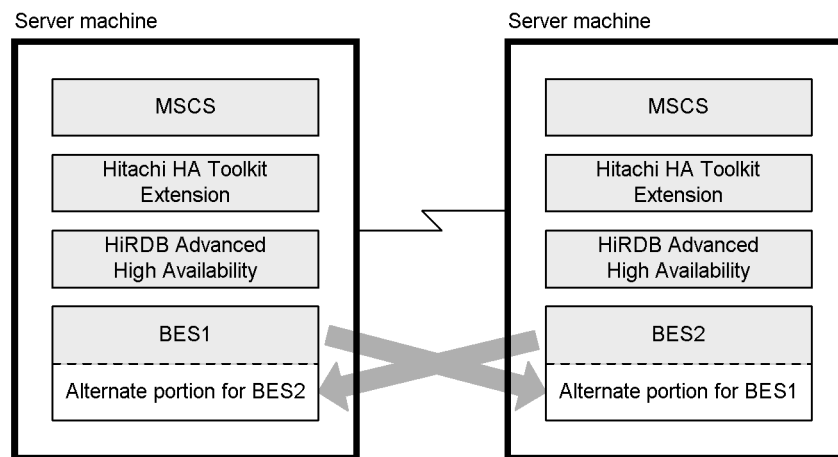
(2) System configuration examples of standby-less system switchover (1:1) facility

This subsection provides examples of typical system configurations when the standby-less system switchover (1:1) facility is used.

(a) Mutual alternating configuration

This configuration example employs reciprocal alternate BESs on two back-end servers using standby-less system switchover (1:1). Figure 8-7 illustrates a system configuration example of such a mutual alternating configuration.

Figure 8-7: System configuration example of a mutual alternating configuration



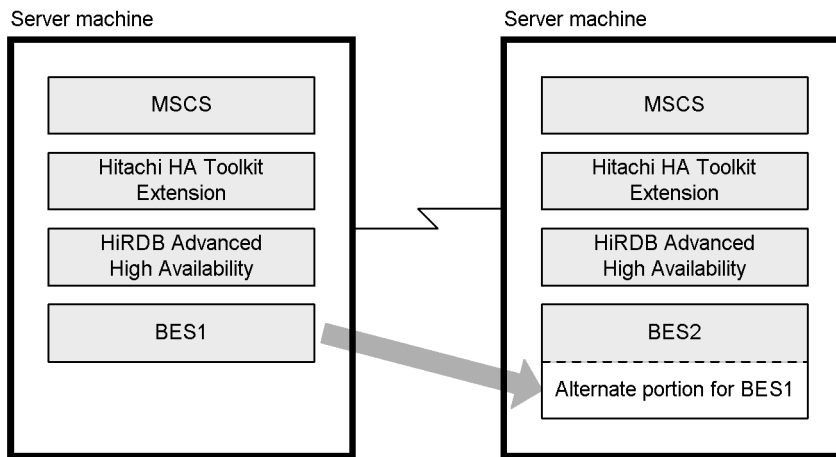
Explanation

- BES1 is the alternate BES for BES2. If a failure occurs on BES2, the alternate portion for BES2 takes over the BES2 processing.
- BES2 is the alternate BES for BES1. If a failure occurs on BES1, the alternate portion for BES1 takes over the BES1 processing.

(b) One-way alternating configuration

This configuration example employs an alternate BES on one back-end server only using standby-less system switchover (1:1). Figure 8-8 illustrates a system configuration example of a one-way alternating configuration (2-node configuration).

Figure 8-8: System configuration example of a one-way alternating configuration (2-node configuration)



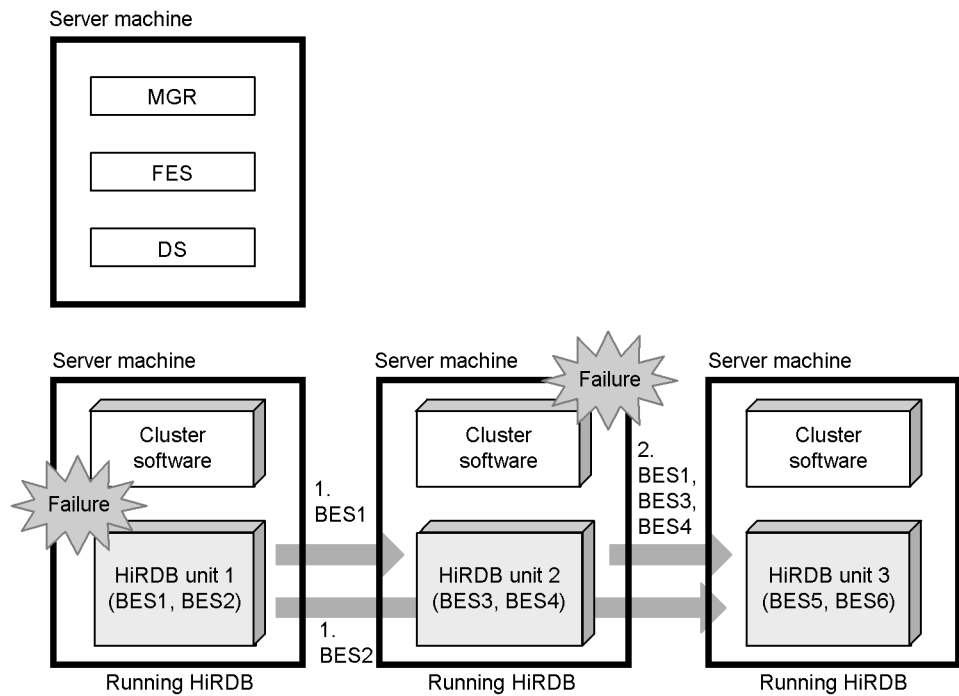
Explanation

BES2 is the alternate BES for BES1. If a failure occurs on BES1, the alternate portion for BES1 takes over the BES1 processing. If a failure occurs on BES2, BES1 does not take over its processing.

(3) System configuration examples of the standby-less system switchover (effects distributed) facility

Figure 8-9 illustrates a system configuration example of the standby-less system switchover (effects distributed) facility. When a failure occurs in a regular unit, processing directed to the back-end servers of the failed primary system is distributed to and executed on multiple active server machines at their back-end servers.

Figure 8-9: System configuration example of the standby-less system switchover (effects distributed) facility



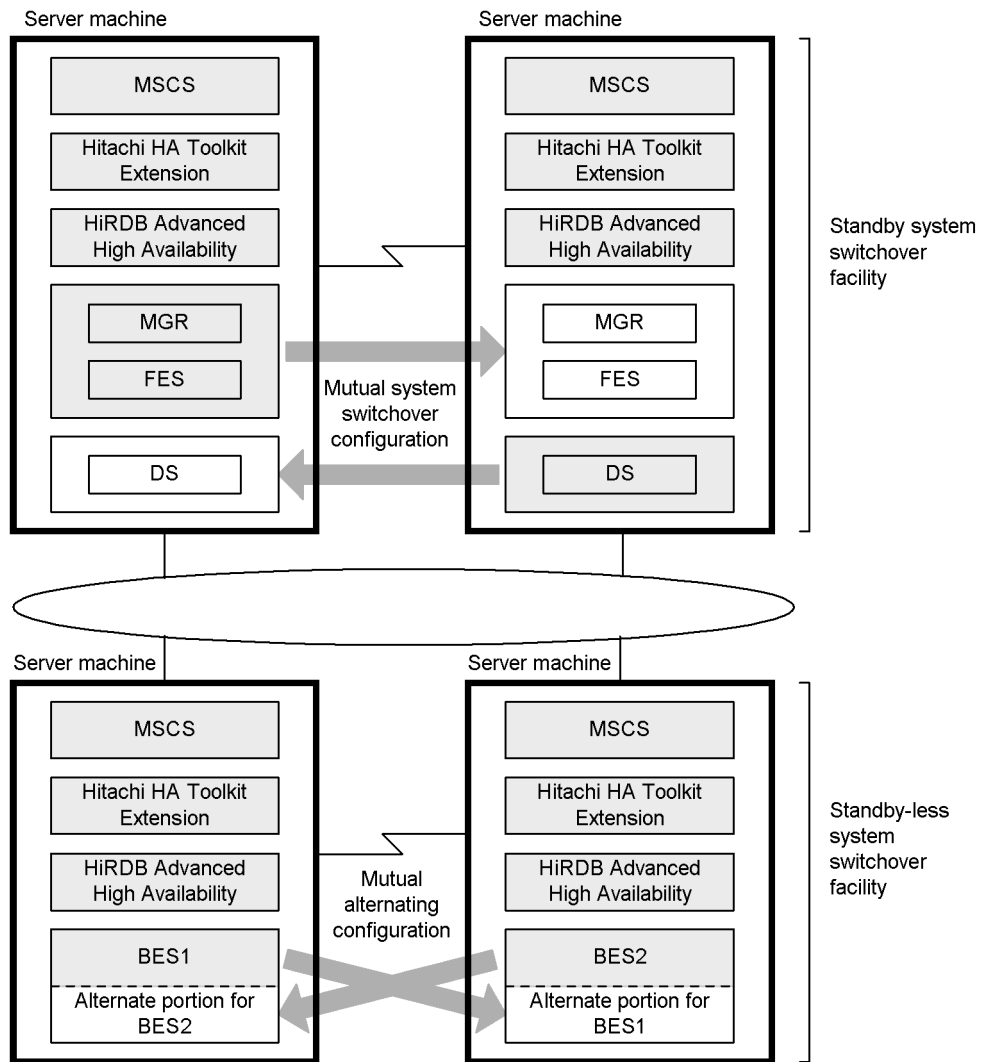
Explanation

1. If a failure occurs in unit 1, unit 2 executes the processing as a guest BES for BES1 and unit 3 executes the processing as a guest BES for BES2.
2. If a failure occurs in unit 2 while unit 1 is still down, unit 3 executes the processing as guest BESs for BES1, BES2, BES3, and BES4.

(4) System configuration example of a mixed standby-less (1:1) and standby type setup

Figure 8-10 shows a configuration example of a mixed standby-less (1:1) and standby type setup.

Figure 8-10: System configuration example of a mixed standby-less (1:1) and standby type setup



Explanation

- The units containing a MGR (system manager), FES (front-end server), and DS (dictionary server) employ the standby system switchover facility in a mutual system switchover configuration.
- The units containing a BES (back-end server) employ the standby-less system switchover (1:1) facility in a mutual alternating configuration.

- HiRDB Advanced High Availability is needed on all server machines. HiRDB Advanced High Availability is also required on server machines for which neither the standby-less system switchover facility nor the standby system switchover facility is employed.

8.1.5 Functions that reduce system switchover time (user server hot standby and the rapid system switchover facility)

HiRDB offers the following functions to reduce system switchover time:

- User server hot standby
- The rapid system switchover facility

Note that, to use these functions, the system switchover facility must be operating in the server mode. You cannot use these functions if the system switchover facility is operating in the monitor mode.

(1) *User server hot standby*

When a system switchover occurs, the following processing is performed to start the standby HiRDB:

- System server startup processing
- System file transfer processing
- Server process startup processing
- Roll forward processing

Of the preceding, the time needed for the server process startup processing accounts for a large proportion of the entire system switchover time. The time required for server process startup processing is directly proportional to the number of resident server processes, which means that system switchover time increases as the number of resident processes increases. You can reduce this time, however, by starting the server processes of the standby HiRDB beforehand, so that server process startup processing is not performed when the system is switched over. This allows you to reduce the system switchover time by the amount of time required for system process startup processing. This concept is called *user server hot standby*. For example, on a server machine that runs at a speed of approximately 100 MIPS, one second is required to start a single server process. In this case, user server hot standby would reduce the system switchover time by about one second for each resident server process.

For details about user server hot standby, see the *HiRDB Version 8 System Operation Guide*.

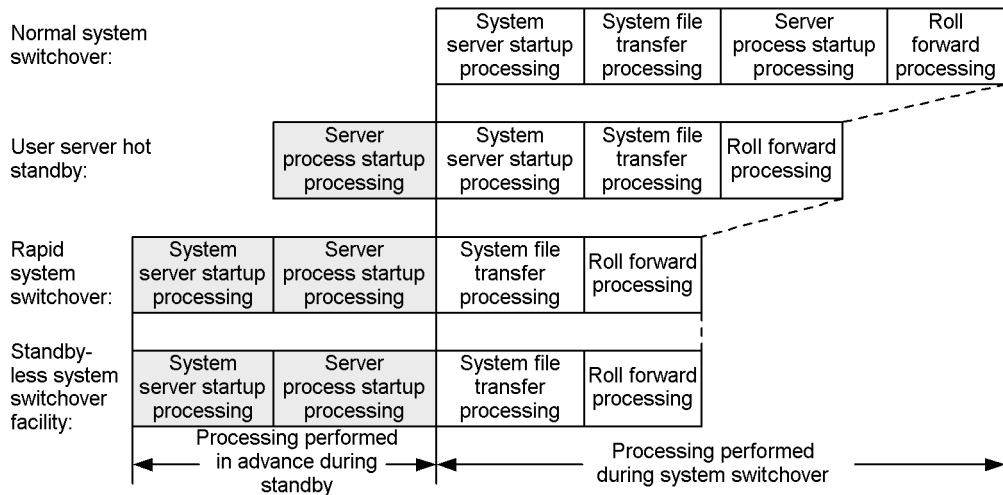
(2) *Rapid system switchover facility*

A facility is provided to start the server processes and the system server on the standby HiRDB beforehand, so that startup processing of the system processes and system

server need not be performed when the system is switched over. This facility is called the *rapid system switchover facility*. With it, you can reduce the system switchover time by the amount of time required to perform startup processing of system processes and the server system. For details about the rapid system switchover facility, see the *HiRDB Version 8 System Operation Guide*.

Note that the rapid system switchover facility can reduce the system switchover time more than user server hot standby can (the rapid system switchover facility includes the user server hot standby function). Figure 8-11 compares the system switchover times of various operating modes.

Figure 8-11: Comparison of system switchover times



Explanation

The processing indicated in the shaded area is performed in advance, after which the associated processes stand by; therefore, this processing does not need to be performed when a system is switched over. Thus, system switchover time is reduced by the amount of time required to perform the processing indicated in the shaded area.

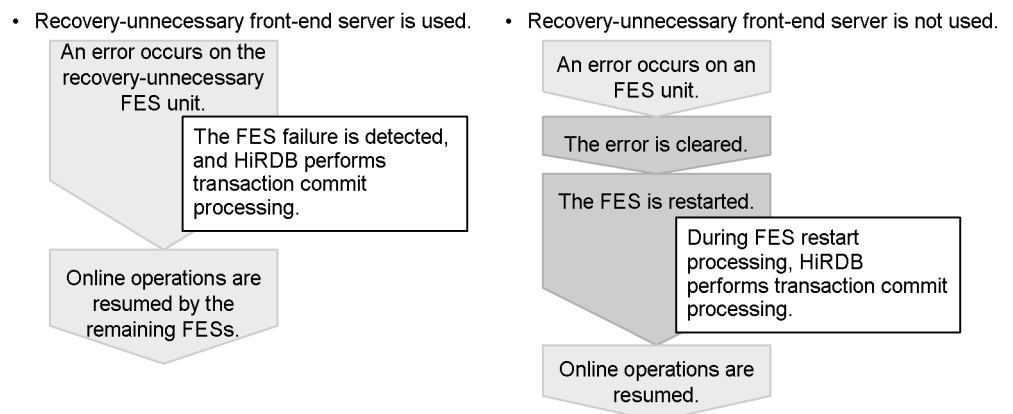
8.2 Recovery-unnecessary front-end servers

Installing HiRDB Non Recover FES enables you to use recovery-unnecessary front-end servers. This section provides a brief description of recovery-unnecessary front-end servers. For details about recovery-unnecessary front-end servers, see the *HiRDB Version 8 Installation and Design Guide*.

8.2.1 Overview of recovery-unnecessary front-end servers

When an error in a unit that contains a front-end server causes the front-end server to terminate abnormally, transactions that were being executed by that front-end server may remain uncommitted. Transactions in uncommitted status maintain an exclusive hold on the database, restricting access to and updating of that locked portion of the database. Normally, the front-end server error must be cleared and the server restarted to commit the uncommitted transactions. However, if a front-end server that is terminated abnormally is a recovery-unnecessary front-end server, HiRDB automatically commits its uncommitted transactions. You can then use another front-end server or back-end server to resume processing on the database. A unit that contains a recovery-unnecessary front-end server is called a *recovery-unnecessary front-end server unit*. Figure 8-12 shows the operation with and without using a recovery-unnecessary front-end server.

Figure 8-12: Operation with and without using a recovery-unnecessary front-end server



Note that HiRDB Non Recover FES must be installed in order to use recovery-unnecessary front-end servers.

Application criteria

This feature enables the front-end servers that remain after an error to resume

online operations without the erroneous front-end server having to be restarted. It is recommended that this feature be used with systems that require continuous, 24-hour per day operation.

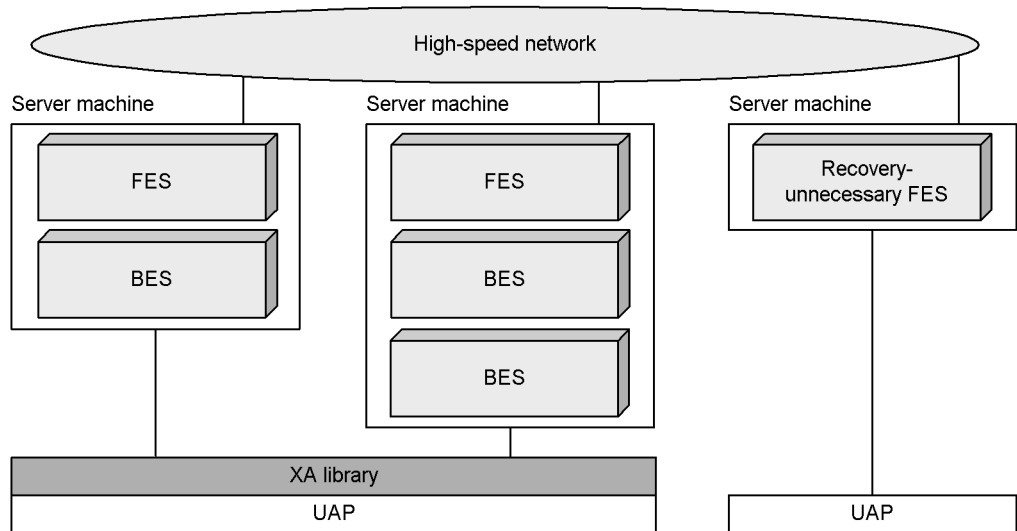
Specification method

To use a recovery-unnecessary front-end server, specify `stls` in the `-k` option of the `pdstart` operand.

8.2.2 Configuration example of a system that uses a recovery-unnecessary front-end server

Figure 8-13 shows a configuration example of a system that uses a recovery-unnecessary front-end server.

Figure 8-13: Configuration example of a system that uses a recovery-unnecessary front-end server



Explanation

- Set up the recovery-unnecessary front-end server in a stand-alone unit.
- UAPs that connect using the X/Open XA interface cannot be executed on a recovery-unnecessary front-end server. Specify `PDFESHOST` and `PDSERVICEGRP` in the client environment definition to direct them to connect to front-end servers other than a recovery-unnecessary front-end server.
- You can still execute the `pdrplstart` and `pdrplstop` commands, even if a recovery-unnecessary front-end server or a recovery-unnecessary front-end server unit is stopped.

Chapter

9. Facilities Related to Security Measures

This chapter describes facilities that provide support for database security measures.

- 9.1 Security facility
- 9.2 Security audit facility
- 9.3 Connection security facility

9.1 Security facility

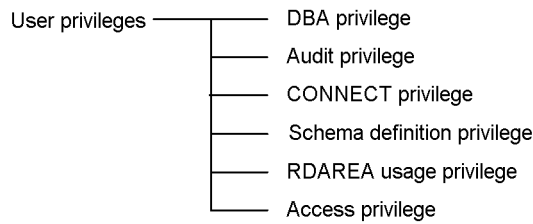
To prevent database access by outsiders, HiRDB provides a *security facility*. Based on a concept called *user privileges*, the security facility allows only users who have an appropriate privilege to access the database.

This section provides an overview of the security facility. For details about how to operate the security facility, see the *HiRDB Version 8 System Operation Guide*.

9.1.1 User privileges

This section explains the user privileges that are set up by HiRDB. Figure 9-1 shows the HiRDB user privileges.

Figure 9-1: HiRDB user privileges



These HiRDB user privileges are granted to various users, such as HiRDB administrators, DBA privilege holders, and schema owners.

Privileges granted to HiRDB administrators

The administrator's own DBA privilege, audit privilege, and RDAREA usage privilege

Privileges granted to DBA privilege holders

DBA privilege, schema definition privilege, RDAREA usage privilege, and CONNECT privilege

Privilege granted to schema owners

Access privilege

(1) DBA privilege

The DBA privilege is required in order to grant and revoke DBA privileges, CONNECT privileges, and schema definition privileges. It permits the following actions:

- Granting DBA privileges, CONNECT privileges, and schema definition privileges to other people

- Revoking DBA privileges, CONNECT privileges, and schema definition privileges granted to other people
- Defining other users' schemas
Defining a schema allows the schema owner to define base tables, view tables, indexes, abstract data types, foreign tables,¹ foreign indexes,¹ stored procedures, stored functions, and triggers.
- Deleting other users' schemas, base tables, view tables, indexes, abstract data types, foreign tables,¹ foreign indexes,¹ stored procedures, stored functions, and triggers
- Defining items related to the connection security facility
- Connecting to HiRDB (CONNECT privilege holders²)
- Defining and modifying foreign servers¹
- Defining user mapping¹

¹ Operation that can be performed when using HiRDB External Data Access. For details about HiRDB External Data Access, see *HiRDB External Data Access Version 8*.

² The CONNECT privilege is not needed to use the Directory Server linkage facility. For details about the Directory Server linkage facility, see Section 2.3 *Linkage to directory server products*.

(2) Audit privilege

This privilege is required for auditors. This privilege allows a user to perform the following actions:

- Accessing audit trail tables
- Loading data to audit trail tables
- Granting and revoking the SELECT privilege to audit trail tables
- Deleting audit trail tables
- Changing auditor passwords
- Defining and deleting audit events
- Swapping audit trail files

To use the security audit facility, you need to set the audit privilege. For details about the security audit facility, see 9.2 *Security audit facility*.

(3) CONNECT privilege

The CONNECT privilege is required in order to use HiRDB. This privilege permits a user to connect to the database. An attempt to connect to the database by a user who does not have the CONNECT privilege results in an error.

Using the Hitachi Directory Server linkage facility

When user information (user ID and password) is registered into the Directory Server, the CONNECT privilege is granted to the user. For details about the Directory Server linkage facility, see Section 2.3 *Linkage to directory server products*.

(4) Schema definition privilege

The schema definition privilege is required in order to define a schema. This privilege permits a user to take the following actions:

- Defining the user's own schema
Defining a schema allows the schema owner to define base tables, view tables, indexes, abstract data types, foreign tables, * foreign indexes, * stored procedures, stored functions, and triggers.
- Deleting the user's own schemas, base tables, view tables, indexes, abstract data types, foreign tables, * foreign indexes, * stored procedures, stored functions, and triggers

* Operation that can be performed when using HiRDB External Data Access. For details about HiRDB External Data Access, see *HiRDB External Data Access Version 8*.

(5) RDAREA usage privilege

The RDAREA usage privilege is required in order to use an RDAREA. This privilege permits a user to define tables and indexes in the RDAREA to which the privilege applies. An RDAREA for which the RDAREA usage privilege is granted by specifying an authorization identifier is called a *private user RDAREA*, and an RDAREA for which the RDAREA usage privilege is granted by specifying PUBLIC is called a *public user RDAREA*.

(6) Access privileges

An access privilege is required in order to access a table. Only those users who have an access privilege are allowed to access a table. Access privileges are set for each table. The types of access privilege are listed in Table 9-1.

Table 9-1: Access privilege types

Access privilege type	Explanation
select privilege	Allows retrieval (<code>select</code>) of row data from the table.
INSERT privilege	Allows addition (<code>INSERT</code>) of row data to the table.
DELETE privilege	Allows deletion (<code>DELETE</code>) of row data from the table.
UPDATE privilege	Allows updating (<code>UPDATE</code>) of row data in the table.

9.1.2 Operating the security facility

HiRDB administrators, DBA privilege holders, and schema administrators operate the security facility by assigning various privileges to HiRDB users. For details about assigning privileges, see the *HiRDB Version 8 System Operation Guide*.

9.2 Security audit facility

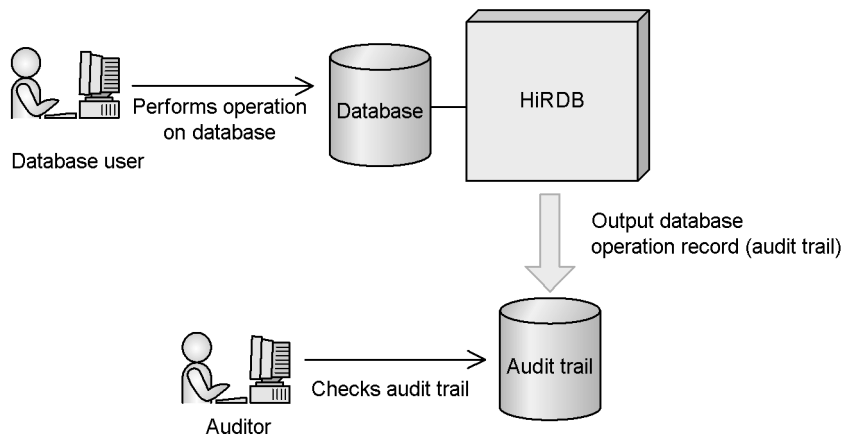
This section describes the security audit facility. For details about how to operate the security audit facility, see the *HiRDB Version 8 System Operation Guide*.

9.2.1 Overview of the security audit facility

(1) Functional overview

HiRDB security is protected by means of privileges. The information that can be accessed or updated, and the objects that can be manipulated (tables, indexes, and so on), are controlled with privileges. To check whether or not these privileges are being applied properly, HiRDB can record a variety of actions that are performed on the databases. This functionality is called the *security audit facility*, and the operations record that it outputs is called an *audit trail*. By examining the output audit trail, you can check whether there has been fraudulent access. This check is performed by users, called *auditors*, who have been assigned the audit privilege. Figure 9-2 provides an overview of the security audit facility.

Figure 9-2: Overview of the security audit facility



The security audit facility collects information about who is using privileges, which privileges they are using, and the objects on which they are using the privileges to perform operations. The auditor uses the `CREATE AUDIT` statement to specify the operations on which the security audit facility is to collect information. Once specified, an audit trail is collected whenever an operation for which an audit trail is specified to be collected is performed.

Reference note:

The purpose of the security audit facility is not to enhance security. It is designed simply to output an operation log that enables checking of whether or not privileges are being used correctly.

(2) Audit trail collection times

HiRDB collects an audit trail when any of the following events occurs:

- When a privilege check is performed because a command or SQL statement is executed
- When an event ends

The security audit facility does not collect an audit trail when an SQL syntax error occurs or when an incorrectly keyed command is entered.

For details about audit trail collection triggers, see the *HiRDB Version 8 System Operation Guide*.

(3) Audit trail collection examples

Examples of audit trail collection are provided in this subsection.

Example 1: Collecting an audit trail when a table is searched

The table access privilege (`SELECT` privilege) is used when a table is searched, so an audit trail is collected.

Item searched (SQL specification)		Contents of audit trail				
		User	Privilege used	Type of manipulated object	Name of manipulated object	Operation type
A user (USR1) issues the following <code>SELECT</code> statement: <code>SELECT C1 FROM USR1.T1</code>	Privilege	USR1	Table access privilege (<code>SELECT</code> privilege)	Table	USR1.T1	Table access (<code>SELECT</code>)
	End	USR1	—	Table	USR1.T1	Table access (<code>SELECT</code>)

Item searched (SQL specification)		Contents of audit trail				
		User	Privilege used	Type of manipulated object	Name of manipulated object	Operation type
A user (USR2) issues the following SELECT statement: SELECT T1.C1, T2.C1 FROM USR1.T1 T1, USR2.T2 T2 WHERE T1.C1=T2.C1	Privilege	USR2	Table access privilege (SELECT privilege)	Table	USR1.T1	Tableaccess (SELECT)
		USR2	Table access privilege (SELECT privilege)	Table	USR2.T2	Tableaccess (SELECT)
	End	USR2	—	Table	USR1.T1	Tableaccess (SELECT)
		USR2	—	Table	USR2.T2	Tableaccess (SELECT)

Legend:

Privilege: Audit trail is collected at time of privilege checking.

End: Audit trail is collected when the event terminates.

—: Not applicable

Example 2: Collecting an audit trail when a table is defined or deleted

The schema owner privilege, table owner privilege, and RDAREA owner privilege are used when a table is defined or deleted, so an audit trail is collected.

Item searched (SQL specification)		Contents of audit trail				
		User	Privilege used	Type of manipulated object	Name of manipulated object	Operation type
A user (USR1) issues the following CREATE TABLE: CREATE TABLE T1 (C1 INT) IN RDAREA1	Privilege	USR1	RDAREA usage privilege	RDAREA	RDAREA1	Definition creation
		USR1	Owner	Schema	USR1	Definition creation
		USR1	Owner	Table	USR1.T1	Definition creation
	End	USR1	—	Table	USR1.T1	Definition creation

Item searched (SQL specification)		Contents of audit trail				
		User	Privilege used	Type of manipulated object	Name of manipulated object	Operation type
A user (USR2) issues the following DROP TABLE: DROP TABLE T1	Privilege	USR2	Owner	Table	USR2.T1	Definition deletion
	End	USR2	—	Table	USR2.T1	Definition deletion

Legend:

Privilege: Audit trail is collected at time of privilege checking.

End: Audit trail is collected when the event terminates.

—: Not applicable

(4) Information collected in audit trails

Table 9-2 lists and describes the information collected in audit trails.

Table 9-2: Information collected in audit trails

Information collected	Description
User identifier	Authorization identifier of the executor of the audit event
Event execution date	Year, month, and date the event was executed
Event execution time	Time the event was executed
Event execution duration	Amount of time it took for the event to execute (in microseconds)
Event type	Event type
Event subtype	Event subtype
Event result	Execution results of the event (whether or not the privilege check was successful)
Privilege used	Privilege used when the event was executed
UAP name	UAP name specified in the PDCLTAPNAME operand of the client environment definition
Service name	Service name requested by the UAP that issued the event. This is the item that corresponds to the service name when an OpenTP1 SUP (service using program) requests a service from an SPP (service providing program), or when TP1/Message Control requests a service from an MHP (message handling program).

9. Facilities Related to Security Measures

Information collected	Description
IP address	Client IP address at which the UAP that issued the event is running*
Process number	Process ID from the UAP that issued the event*
Thread number	Thread ID from the UAP that issued the event*
Host name	Name of the host to which the UAP that issued the event is connected
Unit identifier	Identifier of the unit to which the UAP that issued the event is connected
User name	Name of the front-end server or single server to which the UAP that issued the event is connected
Connection sequence number	Connection sequence number of the event issuer
SQL sequence number	SQL sequence number of the event
Object owner name	Name of the owner of the object on which the event privilege check is performed
Object name	Name of the object on which the event privilege check is performed
Object type	Type of the object on which the event privilege check is performed
Assigned, revoked, or modified privilege	Privilege that was assigned, revoked, or modified due to the event
Identifier of the user who assigned, revoked, or modified a privilege, and the user identifier for the event	Identifier of the user who assigned, revoked, or modified the privilege with the event and the authorization identifier for the event
Values of security audit facility-related operands	Values of operands related to the security audit facility (values at HiRDB startup)
Audit trail type	Indicator of privilege check or event end
SQL code or end code	Code issued when the SQL, utility, or command ends
Swap source audit trail file name	Name of audit trail file at swap source when a swap occurs
Swap target audit trail file name	Name of audit trail file at swap target when a swap occurs
Configuration change type of connection security facility	Configuration change type set in the connection security facility (a change type is also set when the password is changed)
Values of operands related to connection security facility (before change)	Values of operands related to the connection security facility before they have been changed

Information collected	Description
Values of operands related to connection security facility (after change)	Values of operands related to the connection security facility after they have been changed
Audit trail table options	Flag for handling events that target an audit trail table, a view base table of an audit trail table, or a list base table of an audit trail table

Note

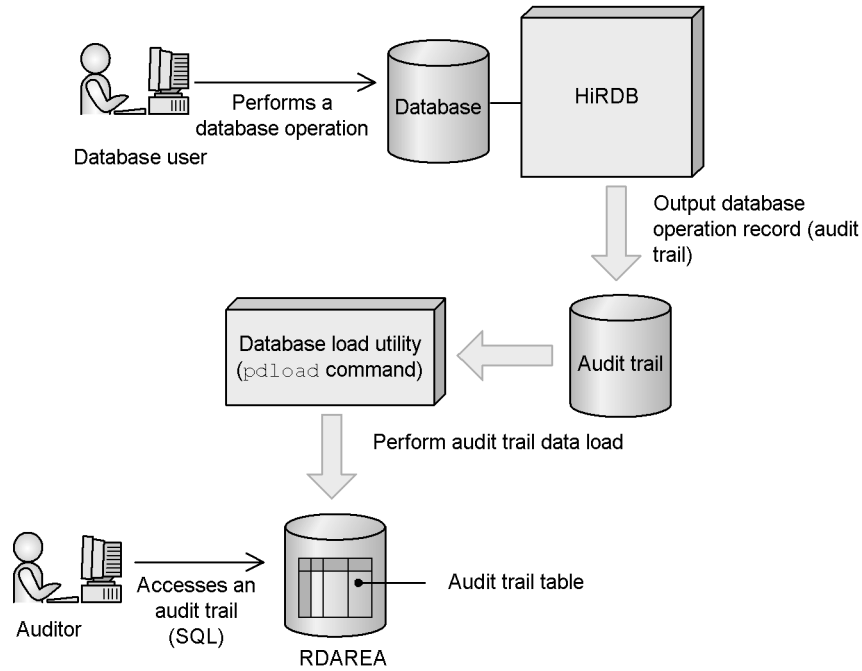
The information items that are collected depends on the event. For a list of the types of information that are collected for each event, see the *HiRDB Version 8 System Operation Guide*.

* For events provided via an application running under OpenTP1 or provided via a Web server or similar product, information is collected from the application to which HiRDB is connected, not from the application that the end user is running.

(5) Accessing an audit trail

Audit trails are output to an *audit trail file*. Data in an audit trail file can be accessed using SQL once the data has been loaded into an *audit trail table* by the database load utility (`pdload` command). Note that the auditor can access (but not update) this audit trail table. Users other than the auditor can access (but not update) an audit trail table only if they are granted access privilege by the auditor. Figure 9-3 shows how to access audit trails.

Figure 9-3: Accessing audit trails



Explanation

1. When an audit event occurs, an audit trail is output to an audit trail file. The audit trail file is created in a HiRDB file system area allocated for audit trail files. For details about audit events, see *9.2.2 Audit events*.
2. Using as the input audit trails that were output to the audit trail file, the database load utility (pdload command) is used to load the data into a table.
3. The auditor uses the audit trail table to perform an audit.

9.2.2 Audit events

Operations that are collected in audit trails are called audit events. Table 9-3 lists and describes audit events.

Table 9-3: Audit events

Event type	Description and list of audit events	Selectable?
System administrator security event	<ol style="list-style-type: none"> 1. Audits security events generated by HiRDB administrators and DBA privilege holders. 2. Audits changes to connection security facility settings. 3. Audits security events generated automatically by the system. An audit trail is output when any of the following events is generated: <ul style="list-style-type: none"> • HiRDB startup (<code>pdstart</code> command)¹ • HiRDB termination (<code>pdstop</code> command)^{1,2} • Auditor registration (<code>pdmod</code> command) • Audit trail table creation (<code>pdmod</code> command) • Audit trail file deletion (<code>pdaudrm</code> command)³ • Start of audit trail collection⁵ • End of audit trail collection⁶ • Start of audit trail file overwrite • Transition to consecutive certification failure account lock state • Release of consecutive certification failure account lock state This occurs at the following times: <ul style="list-style-type: none"> • When <code>CONNECT</code> is executed after the account lock period expires • When <code>DROP CONNECTION SECURITY</code> is executed • When the <code>pdacnlock</code> command is executed • Transition to password invalid account lock state • Release of password invalid account lock state • Change in a connection security facility setting: <ul style="list-style-type: none"> • Permitted number of consecutive certification failures • Account lock period • Items set with password character string restrictions (including pre-checking) • Execution of the <code>pdacnlock</code> command 	No (an audit trail is always output)
Auditor security event	Audits security events generated by the auditor. An audit trail is output when any of the following events is generated: <ul style="list-style-type: none"> • Loading of data into an audit trail table (<code>pdload</code> command) • Swapping of audit trail files (<code>pdaudswap</code> command) • Definition of an audit event (<code>CREATE AUDIT</code>)⁴ • Deletion of an audit trail event (<code>DROP AUDIT</code>)⁴ • Changing an auditor password (<code>GRANT AUDIT</code>)⁴ 	No (an audit trail is always output)
Session security event	Audits events generated by user authentication based on an authorization identifier and password. An audit trail is output when either of the following events is generated: <ul style="list-style-type: none"> • Connection to HiRDB (<code>CONNECT</code> statement) • Changing users (<code>SET SESSION AUTHORIZATION</code> statement) 	Yes

9. Facilities Related to Security Measures

Event type	Description and list of audit events	Selectable?
Privilege control event	<p>Audits events generated by granting and revoking user privileges. An audit trail is output when either of the following events is generated:</p> <ul style="list-style-type: none"> • Granting a user privilege (<code>GRANT</code> statement) • Revoking a user privilege (<code>REVOKE</code> statement) 	Yes ⁷
Object definition event	<p>Audits events generated by definition, deletion, or modification of objects. An audit trail is output when any of the following events is generated:</p> <ul style="list-style-type: none"> • Definition of an object; audits the following SQL statements: <ul style="list-style-type: none"> <code>CREATE FUNCTION</code> <code>CREATE INDEX</code> <code>CREATE PROCEDURE</code> <code>CREATE PUBLIC VIEW</code> <code>CREATE SCHEMA</code> <code>CREATE TABLE</code> <code>CREATE TRIGGER</code> <code>CREATE TYPE</code> <code>CREATE VIEW</code> • Deletion of an object; audits the following SQL statements: <ul style="list-style-type: none"> <code>DROP DATA TYPE</code> <code>DROP FUNCTION</code> <code>DROP INDEX</code> <code>DROP PROCEDURE</code> <code>DROP PUBLIC VIEW</code> <code>DROP SCHEMA</code> <code>DROP TABLE</code> <code>DROP TRIGGER</code> <code>DROP VIEW</code> • Modification of an object; audits the following SQL statements: <ul style="list-style-type: none"> <code>ALTER PROCEDURE</code> <code>ALTER ROUTINE</code> <code>ALTER TABLE</code> <code>ALTER TRIGGER</code> <code>COMMENT</code> 	Yes ⁷

Event type	Description and list of audit events	Selectable?
Object operation event	Audits events generated by object manipulation. An audit trail is output when any of the following events is generated: <ul style="list-style-type: none"> • Searching a table (<code>SELECT</code> statement) • Insertion of rows in a table (<code>INSERT</code> statement) • Updating of rows in a table (<code>UPDATE</code> statement) • Deletion of rows from a table (<code>DELETE</code> statement) • Deletion of all rows from a table (<code>PURGE TABLE</code> statement) • Execution of a stored procedure (<code>CALL</code> statement) • Locking a table (<code>LOCK TABLE</code> statement) • Creation of a list (<code>ASSIGN LIST</code> statement) 	Yes ⁷
Utility operation event	Audits security events generated from operations on objects by utilities or commands. An audit trail is output when any of the following events is generated: <ul style="list-style-type: none"> • Database load command (<code>pdload</code> command) Object: <code>TABLE</code> • <code>pddefrev</code> command Object: <code>PROCEDURE</code>, <code>TABLE</code>, <code>TRIGGER</code>, and <code>VIEW</code> • Database reorganization utility (<code>pdroorg</code> command) Object: <code>TABLE</code> • Dictionary import/export utility (<code>pdexp</code> command) Object: <code>PROCEDURE</code>, <code>TABLE</code>, <code>TRIGGER</code>, and <code>VIEW</code> • Integrity check utility (<code>pdconstck</code> command) Object: <code>TABLE</code> 	Yes ^{7, 8}

¹ In the case of a HiRDB/Parallel Server, startup and termination of a single server are not audit events.

² Normal termination and planned termination are audit events; forced termination and abnormal termination are not audit events. For these cases, check the messages output by HiRDB or the operating system.

The following termination commands are not monitored:

- `pdstop -f`
- `pdstop -f -q`
- `pdstop -f -x host-name`
- `pdstop -f -u unit-identifier`
- `pdstop -f -s server-name`
- `pdstop -f -u unit-identifier -s server-name`
- `pdstop -z`
- `pdstop -z -q`

9. Facilities Related to Security Measures

- `pdstop -z -c`
- `pdstop -z -s server-name`

³ Creation of an audit trail file is not an audit event. To audit creation of audit trail files, use the audit facility provided by the OS.

⁴ You can also output an audit trail by executing the database definition utility (`pddef` command) or the interactive SQL execution utility.

⁵ An audit trail is output when the `pdaudbegin` command is executed or when an audit trail is collected at HiRDB startup.

⁶ An audit trail is output when the `pdaudend` command is executed or when an audit trail is collected during performance of normal or planned termination of HiRDB.

⁷ An audit trail is output unconditionally when the event terminates in the case of privilege control events, object definition events, object operation events, and utility operation events that target an audit trail table, a view base table of an audit trail table, or a list base table of an audit trail table. You can select whether or not to collect an audit trail when a privilege check is performed.

⁸ An audit trail is output unconditionally when the database reorganization utility (`pdorg` command) is used to reload a dictionary table.

9.3 Connection security facility

This section provides a brief description of the connection security facility. For details about operating the connection security facility, see the *HiRDB Version 8 System Operation Guide*.

9.3.1 Overview of the connection security facility

One means of enhancing system security is to use passwords. HiRDB enables you to set a password for each user. However, if simple, easy-to-guess passwords are used (for example, one's authorization identifier or birth date), there is an increased possibility that a fraudulent user could use that password to gain access to the system. To prevent fraudulent use of passwords, use of the connection security facility is recommended. Table 9-4 provides an overview of the connection security facility.

Table 9-4: Overview of connection security facility

Function	Description
Restrictions on password character strings	You can place restrictions on the character strings that can be specified as passwords. For example, you can prohibit passwords such as 012345 or aaaaa. Prohibiting simple passwords tends to increase password security.
Restrictions on the number of consecutive certification failures	If an incorrect password is entered a specified number of times in succession, you can bar that user from connecting (CONNECT) to HiRDB. To do so, you set the maximum number of times an incorrect password can be entered in consecutive connection attempts, and any user who exceeds the set number of attempts can no longer connect to HiRDB. For example, you might permit a user three attempts to enter the correct password. On the fourth unsuccessful attempt, the user would be prohibited from connecting to HiRDB.

By combining these two functions, fraudulent use of passwords based on ease of discovery can be made more difficult, which enhances system security.

Note:

You cannot use a directory server linkage facility and the connection security facility at the same time. If you use a directory server linkage facility, you must clear the settings for the connection security facility.

9.3.2 Restrictions on password character strings

(1) Restrictions that can be specified for passwords

Table 9-5 lists the restrictions that can be specified for passwords.

Table 9-5: Restrictions that can be specified for passwords

Item	Description
Set a minimum password length	You can set a minimum length for a password (minimum number of bytes).
Prohibit specification of authorization identifier	You can prohibit use of one's authorization identifier as a password.
Prohibit only one type of characters*	You can prohibit passwords that consist of only one type of characters (for example, only uppercase alphabetic characters, or only numeric characters, etc.).

* Only the following types of characters can be used in passwords:

- Uppercase alphabetic characters: A to Z, #, @, \
- Lowercase alphabetic characters: a to z
- Numeric characters: 0 to 9

Reference note:

You cannot set password character string restrictions for individuals users. The settings you specify here apply to all users of HiRDB (including DBA privilege holders and auditors).

(2) Effects on existing users

When you set restrictions on passwords, the account of any user who violates a restriction is placed in *password-invalid account lock state*. Users placed in this status can no longer connect (`CONNECT`) to HiRDB.

Such a user's password must be changed in order to release the password-invalid account lock state.

Before setting restrictions on passwords, you should check how many user accounts are likely to be placed in password-invalid account lock state due to restriction violations.

(3) Effects on new users

If a password set for a new user with a `GRANT DBA`, `GRANT AUDIT`, or `GRANT CONNECT` statement violates a restriction, the `GRANT` statement will not execute.

(4) Specification method

You use `CREATE CONNECTION SECURITY` to set restrictions on password character strings.

9.3.3 Restrictions on the number of consecutive certification failures

(1) Restrictions that can be specified

You can specify that when an incorrect password is entered a specified number of times in succession, that user is to be prohibited from connecting (`CONNECT`) to HiRDB. You make this specification by setting a maximum number of consecutive connection failures (*permitted number of consecutive certification failures*). Any user who exceeds this number of unsuccessful password entry attempts will be barred from connecting to HiRDB.

For example, if you specify 3 for the permitted number of consecutive certification failures, any user's fourth consecutive failed attempt to gain user certification will place that user in *consecutive certification failure account lock state*. Users placed in this status can no longer connect (`CONNECT`) to HiRDB.

Reference note:

You cannot set restrictions on the number of consecutive certification failures for individual users. The settings you specify here apply to all users of HiRDB (including DBA privilege holders and auditors).

You can also set a time period during which an account will be kept in consecutive certification failure account lock state (*account lock period*). For example, if you set the account lock period at 1 hour, a consecutive certification failure account lock state will remain in effect for one hour. After one hour, the consecutive certification failure account lock state will be released and the user will be permitted once again to connect to HiRDB.

Reference note:

- You can also set the account lock period to be indefinite (permanent).
- You can release an account from consecutive certification failure account lock state before the account lock period has expired.

(2) Specification method

You use `CREATE CONNECTION SECURITY` to set restrictions on the number of consecutive certification failures.

Chapter

10. Plug-ins

This chapter provides an overview of and explains the functions of the HiRDB plug-ins; it also explains the HiRDB preparations for using plug-ins.

This chapter contains the following sections:

- 10.1 Overview of HiRDB plug-ins
- 10.2 Applying a plug-in to a job
- 10.3 HiRDB plug-in facilities
- 10.4 Preparations for using plug-ins in HiRDB

10.1 Overview of HiRDB plug-ins

HiRDB is based on a *plug-in architecture*, in which *abstract data types* for multimedia data, such as documents and images, and *facilities that enable simple, high-speed operations on data* can be used as a packaged product. With a plug-in architecture, facilities can be expanded to build a system that can handle multimedia data simply by registering plug-ins into HiRDB.

Moreover, application programs can handle multimedia data easily and at high speed without the programmer having to be concerned with complex operational details typical of multimedia data, and application programs can thus provide information services to customers efficiently. Table 10-1 lists the HiRDB plug-ins.

Table 10-1: HiRDB plug-ins

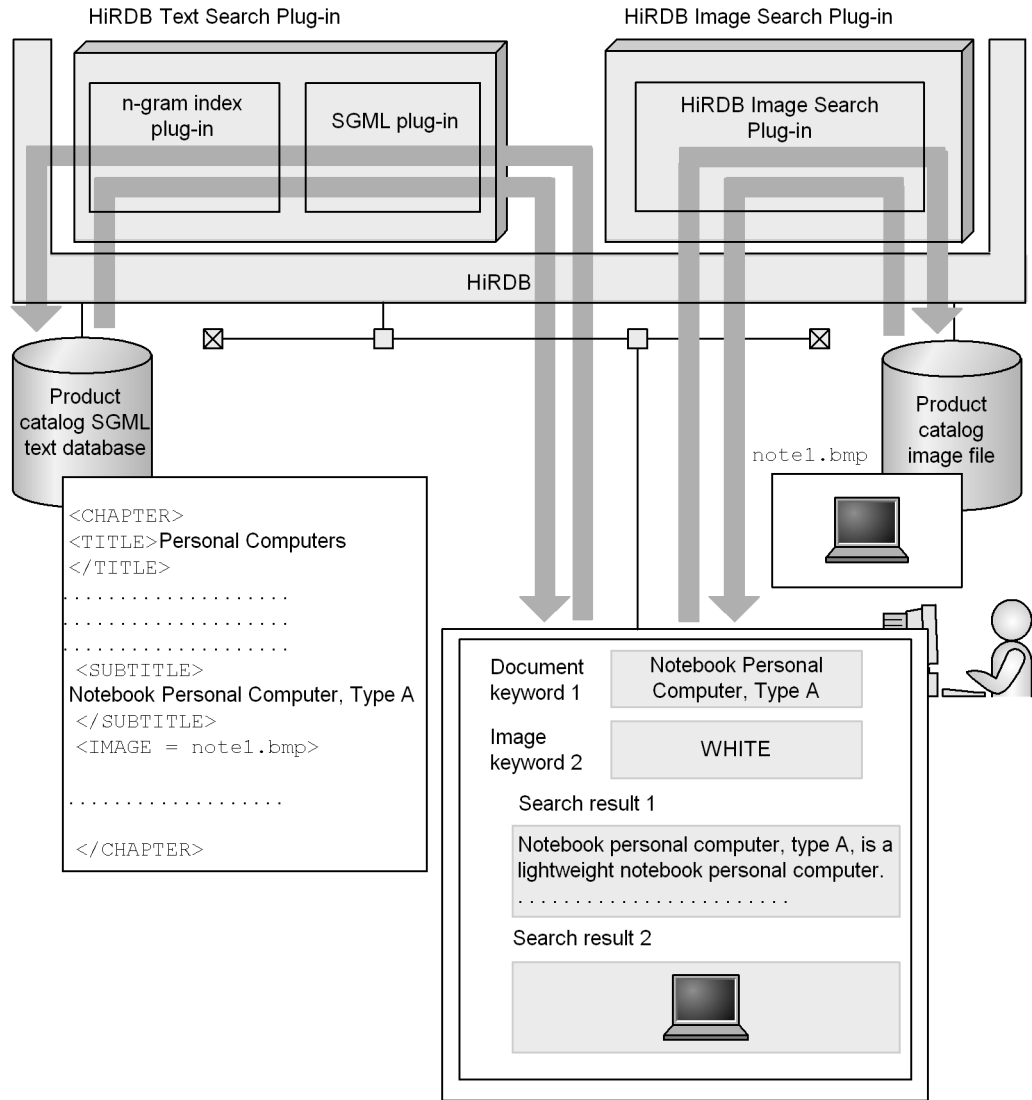
Plug-in	Explanation
HiRDB Text Search Plug-in	Provides a function for retrieving a large volume of document information accurately and quickly.
HiRDB Image Search Plug-in	Provides a function for extracting feature quantities from the colors and shapes of images and a function for retrieving similar images by comparing features.
HiRDB File Link	Implements a mode for storing a large volume of large-scale, data, such as image data, at any server (file server) and registers data linkage information in the HiRDB database. This storage mode minimizes the database workload even when the amount of data increases. When the amount of data increases, this plug-in can increase the disk capacity or the number of file servers.
HiRDB Spatial Search Plug-in	Provides a function for retrieving spatial data (two-dimensional data), such as map information.

For details about the HiRDB plug-in products, see the manuals for the applicable HiRDB plug-ins.

10.2 Applying a plug-in to a job

Let's assume that a company needs to use a database to manage data for its personal computer products catalog. The company's personal computer products catalog may contain document data on desktop machines, notebook computers, CPUs, and memory devices, as well as image data on personal computers. When these documents and image data are stored in a HiRDB database, and the HiRDB Text Search Plug-in and HiRDB Image Search Plug-in are registered into HiRDB, it becomes possible to use keywords to retrieve at high speed both document data and image data. When a large volume of image data must be handled, HiRDB File Link can be used to store the image data at any server (file server) and link this data to columns in a table of the HiRDB database. Figure 10-1 shows the application of plug-ins to a job.

Figure 10-1: Application of plug-ins to a job



Explanation

- This example uses the HiRDB Text Search Plug-in (SGML plug-in and *n*-gram index plug-in) and the HiRDB Image Search Plug-in (features search plug-in and features search index creation plug-in), which are plug-in

package products.

- The product catalog database is built by registering into HiRDB the HiRDB Text Search Plug-in (SGML plug-in and n -gram index plug-in) and the HiRDB Image Search Plug-in (HiRDB Image Search Plug-in and a plug-in for creating an image search index).
- It is possible to retrieve documents using the keyword "notebook personal computer A" and retrieve personal computer image data using "WHITE" as the key.

For details about the function and operation of each plug-in, see the manual for the respective plug-in.

10.3 HiRDB plug-in facilities

HiRDB provides the following plug-in products:

- HiRDB Text Search Plug-in
- HiRDB Image Search Plug-in
- HiRDB File Link
- HiRDB Spatial Search Plug-in

The benefits of each of these plug-ins are explained as follows.

10.3.1 HiRDB Text Search Plug-in

The HiRDB Text Search Plug-in provides the following text structure search facilities:

- SGML and XML document registration
- Flat document registration
- Retrieval with structure specification
- Synonym and spelling variation retrieval
- Score retrieval
- Retrieval result text data extraction
- Embedding of retrieval hit location highlighting tag

Each of these facilities is explained as follows.

(1) SGML and XML document registration

A utility provided by the HiRDB Text Search Plug-in can be used to register into the HiRDB database a DTD file, which defines tag names that indicate the structure and elements of an SGML and XML document. When the SGMLTEXT constructor facility based on the registered DTD file is used, the SGML and XML document together with the document structure information can be registered into the HiRDB database.

(2) Flat document registration

Flat documents that do not have a structure can be registered into the HiRDB database.

(3) Retrieval with structure specification

A full-text search for SGML and XML documents can be performed by using the `contains` abstract data type function to specify the columns to be retrieved and the retrieval condition (document structure name to be retrieved, a conditional expression that specifies the keyword to be retrieved).

(4) Synonym and spelling variation retrieval

A utility provided by the HiRDB Text Search Plug-in can be used to register synonyms and spelling variations into a local file. Based on the registered synonyms and spelling variations, synonyms or spelling variations of a keyword used for a search can also be retrieved during a full-text search for SGML documents. For example, a search for the keyword "computer" can also retrieve a synonym such as "electronic computing machine" and spelling variations such as "COMPUTER" and "Computer".

(5) Score retrieval

The `contains_with_score` and score abstract data type facilities provided by the HiRDB Text Search Plug-in can be used to compute points (scores) based on the frequency of occurrence of the keyword and to display the retrieval results according to the scores.

(6) Note

HiRDB Text Search Plug-in versions 02-02 and earlier do not support UTF-8 as a character code classification. If you use a version of HiRDB Text Search Plug-in that does not support UTF-8, do not specify UTF-8 for the character code classification with the `pdntenv` command. For details about support for UTF-8, see the HiRDB Text Search Plug-in documentation or the *Readme* file.

10.3.2 HiRDB Image Search Plug-in

The HiRDB Image Search Plug-in provides the following facilities:

- Extraction and registration of image data features
- Retrieval based on image features
- Creation of column index for image features

(1) Extraction and registration of image data features

The HiRDB Image Search Plug-in can extract image features from the colors and shapes of image data and register them into the HiRDB database. PBM is the only data type that the HiRDB Image Search Plug-in can handle.

When the digital contents input/output utility provided by Image Database Access is used, image data is first converted to the PBM format before being transferred to the HiRDB Image Search Plug-in, and therefore image features can be extracted from image data in other formats.

(2) Retrieval based on image features

Image Database Access (CORBA object) is used to compare image features that have been registered in the database with the image features of the image to be retrieved and to determine their similarity.

(3) Creation of column index for image features

The index creation plug-in for image searches can be used to create an index for a column that stores image features (ImageFeature-type column). When such an index has been created, images can be retrieved at high speed when image data is specified as the retrieval condition.

10.3.3 HiRDB File Link

When you are handling large volumes of image data, you can use HiRDB File Link to store the image data on a file server and to store, in a table column, the information needed for linking to this image data stored on the file server. This storage mode minimizes the database load even when the amount of data increases. In addition, increases in data volume can be handled by increasing the disk capacity or the number of file servers. Data registration and retrieval are implemented as explained as follows.

(1) Data registration

A utility provided by Image Database Access is used to store image data in a file server and to store information on the linkage to the image data in a FileLink-type column of the HiRDB database.

(2) Data retrieval

Image Database Access (CORBA object) is used to retrieve image data based on the linkage information to image data.

10.3.4 HiRDB Spatial Search Plug-in

The HiRDB Spatial Search Plug-in provides the following facilities:

- Spatial data registration
- Spatial data retrieval

(1) Spatial data registration

The `GEOMETRY` abstract data type facility provided by the HiRDB Spatial Search Plug-in makes it possible to define positional information in a map as spatial data (two-dimensional data with X and Y coordinates specified) and to register it into the HiRDB database.

(2) Spatial data retrieval

The `Within` abstract data type facility provided by the HiRDB Spatial Search Plug-in can be used to evaluate the spatial relationships between the graphic object specified in a retrieval condition expression and a `GEOMETRY`-type graphic object, and to determine whether or not the specified graphic object is included. For example, one could retrieve "banks located within 100 m of the train station."

The `IntersectsIn` abstract data type facility can also be used to evaluate the spatial relationships between the graphic object specified in a retrieval condition expression

and a `GEOMETRY`-type graphic object, and to determine whether or not the specified graphic object is included or intersects. For example, one could retrieve "roads, train tracks, and rivers that pass through a certain area."

10.4 Preparations for using plug-ins in HiRDB

The preparations explained in this section must be made in order to use plug-ins in HiRDB.

- Setup and registration of a plug-in
To use a plug-in, it must be set up and registered into HiRDB.
- Initialization of the registry facility
The registry facility must be initialized before registry information can be registered.
- Table definition for plug-in usage
An object relational database must be created using the abstract data types and index types provided by the plug-in.
- Delayed batch creation of plug-in index
Instead of adding (updating) plug-in index data when row data is added (updated), it is possible to use the database reorganization utility later to add (update) the plug-in index data in a batch.

Each of these preparatory items for using plug-ins is explained as follows.

10.4.1 Setup and registration of a plug-in

To use a plug-in, the plug-in must first be installed, and then it must be set up and registered in the HiRDB environment. For details about how to set up and register plug-ins, see the *HiRDB Version 8 Installation and Design Guide*. For details about installing plug-ins, see the documentation for the applicable plug-ins.

10.4.2 Initialization of the registry facility

An SGML document may have a structure such as chapter, section, item, and list in the document itself. If the individual items of data have structures, the user must enable the plug-in to recognize the each data item's structure when the document is registered. This type of information that is specific to a plug-in and that is to be used by the plug-in for data manipulation is called *registry information*. The facility by which HiRDB maintains registry information is called the *registry facility*.

Before the registry facility can be used for a plug-in, it must be initialized; HiRDB's registry facility initialization utility (`pdreginit`) is used for this purpose.

Initializing the registry facility creates a registry RDAREA and a registry LOB RDAREA and stores in HiRDB RDAREAs a table for managing registry information (registry management table) and data manipulation stored procedures (such as for registering information into the registry management table).

Table 10-2 shows the RDAREAs used for storing information related to the registry facility.

Table 10-2: RDAREAs for storing information related to registry facility

Registry facility information	Storage RDAREA
Registry management table	Registry RDAREA or registry LOB RDAREA*
Data manipulation stored procedures, such as for registering information into the registry management table	Data dictionary LOB RDAREA

* The RDAREA to be used for storing information is determined automatically on the basis of the length of the data to be registered (up to 32,000 bytes is stored in the registry RDAREA; 32,000 bytes or longer is stored in the registry LOB RDAREA).

For details about registry facility initialization, see the *HiRDB Version 8 Installation and Design Guide*. For details about the registry facility setup utility (`pdreginit`), see the *HiRDB Version 8 Command Reference*.

10.4.3 Table definition for plug-in usage

Once a plug-in has been set up and registered, it provides abstract data types and index types, thus enabling the user to easily build an object relational database that can handle complex data, such as multimedia information, without having to define a complex data structure or operations using abstract data types.

To define a table, an abstract data type provided by the plug-in is specified as the data type of the column in the table. To define an index, an index type provided by the plug-in is specified as the index. An index for which an index type provided by a plug-in is specified is called a *plug-in index*. The specifics of the plug-in index facility depend on the particular plug-in; for details, see the manual for the applicable plug-in.

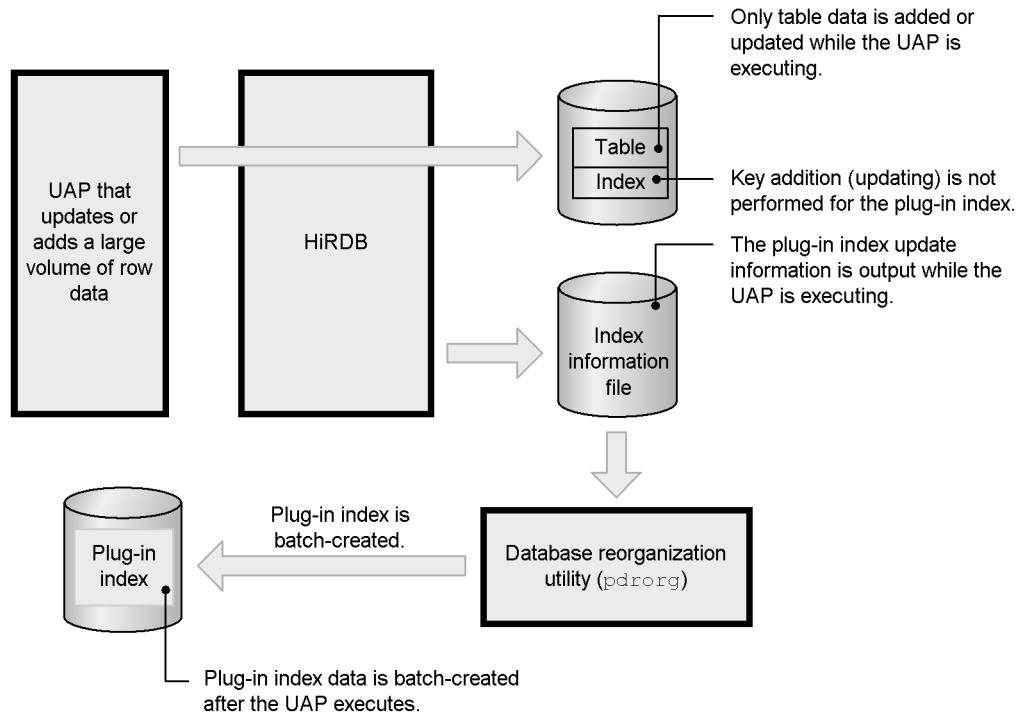
For details about designing and creating an object relational database to be used with plug-ins, see the *HiRDB Version 8 Installation and Design Guide*.

10.4.4 Delayed batch creation of plug-in index

Addition or updating of row data in a table for which a plug-in index is defined requires that keys be added to the plug-in index. When a large amount of row data is added or updated, the resulting addition of keys to the plug-in index can degrade performance.

Instead of adding the plug-in index data at the time the row data is added, the database reorganization utility (`pdroorg`) can be used at a later time to add the plug-in index data in a batch; this is called *delayed batch creation of a plug-in index*. Figure 10-2 shows delayed batch creation of a plug-in index.

Figure 10-2: Delayed batch creation of plug-in index



Prerequisite

It must be determined whether or not the plug-in being used supports delayed batch creation of plug-in index. Delayed batch creation of plug-in index cannot be used for a plug-in that does not support this facility.

The HiRDB Text Search Plug-in supports delayed batch creation of plug-in index.

Advantage

Because data is not added to the plug-in index, the execution time (table data creation time) for a UAP that adds or updates a large volume of data can be reduced considerably.

Operating procedure

For details about the operating procedure for using delayed batch creation of plug-in index, see the *HiRDB Version 8 System Operation Guide*.

Chapter

11. 64-Bit-Mode HiRDB

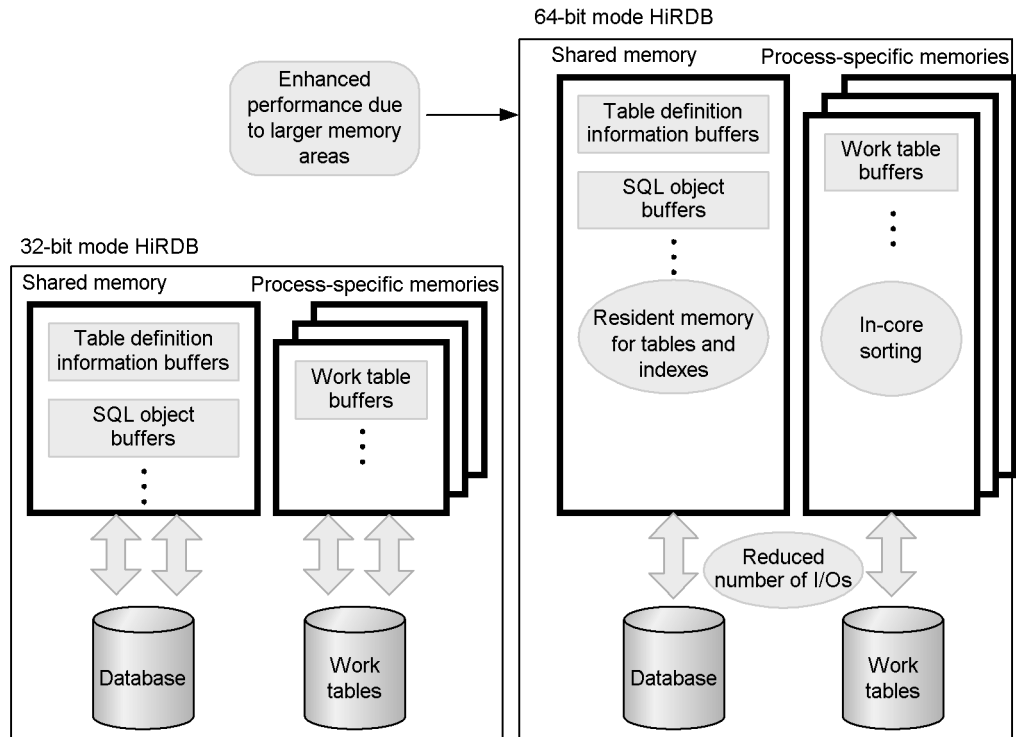
To accommodate larger systems, a 64-bit-mode HiRDB is now supported. This chapter explains the 64-bit-mode HiRDB.

- 11.1 Overview
- 11.2 Functions not available in 64-bit-mode HiRDB
- 11.3 Differences between 32- and 64-bit modes
- 11.4 Migrating from 32-bit mode to 64-bit mode

11.1 Overview

The 64-bit-mode HiRDB can allocate a larger memory area than the 32-bit-mode HiRDB, thus reducing the number of input/output operations on databases and work tables and enhancing performance. Moreover, operating HiRDB in a 64-bit space enables the 32-bit space that was used previously by HiRDB to be allocated as an area for application operations, such as a common library. Figure 11-1 compares the 32- and 64-bit modes.

Figure 11-1: Comparison between 32- and 64-bit-mode HiRDBs



11.2 Functions not available in 64-bit-mode HiRDB

Not all HiRDB-related products are compatible with the 64-bit mode. Functions that require incompatible products cannot be used in the 64-bit-mode HiRDB. Table 11-1 shows the applicability of related products in 64-bit-mode HiRDB.

Table 11-1: Applicability of related products in 64-bit-mode HiRDB

Type	Product	Applicable ?		Related facilities
		IPF	x64	
HiRDB option program products	HiRDB External Data Access	N	N	<ul style="list-style-type: none"> HiRDB External Data Access facility
	HiRDB External Data Access Adapter	N	N	
	HiRDB Advanced High Availability	Y	Y	<ul style="list-style-type: none"> Dynamic update of global buffer System reconfiguration command Standby-less system switchover (1:1) facility Standby-less system switchover (effects distributed) facility
	HiRDB LDAP Option	N	N	<ul style="list-style-type: none"> Sun Java System Directory Server linkage facility
	HiRDB Advanced Partitioning Option	Y	Y	<ul style="list-style-type: none"> Table matrix partitioning Changing the partition storage conditions
	HiRDB Non Recover FES	Y	Y	<ul style="list-style-type: none"> Recovery-unnecessary front-end server settings
HiRDB operation support products	HiRDB SQL Executer	Y	Y	<ul style="list-style-type: none"> SQL interactive execution
	HiRDB Control Manager	Y	Y	<ul style="list-style-type: none"> HiRDB operations support
	HiRDB SQL Tuning Advisor	Y	Y	<ul style="list-style-type: none"> SQL trace information analysis
Plug-ins	HiRDB Text Search Plug-in	N	N	<ul style="list-style-type: none"> HiRDB Text Search Plug-in
	HiRDB Image Search Plug-in	N	N	<ul style="list-style-type: none"> HiRDB Image Search Plug-in

Type	Product	Applicable ?		Related facilities
		IPF	x64	
	HiRDB Spatial Search Plug-in	N	N	• HiRDB Spatial Search Plug-in
	HiRDB File Link	N	N	• HiRDB File Link
Data linkage	HiRDB Datareplicator	N	N	• Data linkage facility
	HiRDB Dataextractor	N	N	• Database extraction/ reflection service facility
JP1	JP1/Base	Y	N	• Linkage with JP1
	JP1/System Event Service	N	N	
	JP1/Automatic Operation Monitor	N	N	
	JP1/Integrated Management	Y	N	
Backup support products	JP1/VERITAS NetBackup Agent for HiRDB License	Y	Y	• NetBackup linkage facility
DABroker	DABroker	Y	Y	• DABroker
	DABroker for C++	Y	Y	
	DABroker for ORB	Y	Y	
	DABroker for COBOL	Y	Y	
	DABroker for Java	Y	Y	
Related products	TP1/Server Base	Y	Y	• OpenTP1
	Sun Java System Directory Server	N	N	• Sun Java System Directory Server linkage facility

Legend:

IPF: Windows Server 2003 (IPF)

x64: Windows (x64)

Y: This product can be used in the 64-bit mode version of HiRDB (the facility can be used).

N: This product cannot be used in the 64-bit mode version of HiRDB (the facility cannot be used).

11.3 Differences between 32- and 64-bit modes

In the 64-bit mode, some HiRDB system definition and client environment definition operands have higher maximum specification values. The two modes also support different UAP languages.

11.3.1 Differences in HiRDB system definition

The 64-bit mode enables higher maximum values to be specified in some definition operands, as shown in Table 11-2.

Table 11-2: Operands with higher maximum value specifications (HiRDB system definition)

Operand	Specification item	Max value in 32-bit mode	Max value in 64-bit mode
pd_hash_table_size	Hash table size when application of hash execution for a hash join or a subquery is specified	524,288 kilobytes	2,097,152 kilobytes
pd_sql_object_cache_size	SQL object buffer size	256,000 kilobytes	2,000,000 kilobytes
pd_table_def_cache_size	Table definition information buffer size	65,535 kilobytes	2,000,000 kilobytes
pd_view_def_cache_size	View analysis information buffer size	32,000 kilobytes	2,000,000 kilobytes
pd_type_def_cache_size	User-defined type information buffer size	65,536 kilobytes	2,000,000 kilobytes
pd_routine_def_cache_size	Routine definition information buffer size	65,536 kilobytes	2,000,000 kilobytes
pd_sds_shmpool_size	Single-server shared memory size	200,000 kilobytes	4,000,000,000 kilobytes
pd_dic_shmpool_size	Dictionary server shared memory size	200,000 kilobytes	4,000,000,000 kilobytes
pd_bes_shmpool_size	Back-end server shared memory size	200,000 kilobytes	4,000,000,000 kilobytes

Operand	Specification item	Max value in 32-bit mode	Max value in 64-bit mode
pd_lck_pool_size	Lock pool size per server	2,000,000 kilobytes	2,000,000,000 kilobytes
pd_fes_lck_pool_size	Lock pool size for front-end servers	2,000,000 kilobytes	2,000,000,000 kilobytes
pd_work_buff_size	Work table buffer size	1,000,000 kilobytes	4,000,000,000 kilobytes
pdbuffer	Number of global buffer definitions	500 per server*	1,000 per server*
pdbuffer -n	Number of global buffer sectors	460,000	1,073,741,824
SHMMAX	Maximum shared memory segment size	2,047 megabytes	4,194,304 megabytes

* Total of 1600 definitions per system.

11.3.2 Differences in client environment definition

The 64-bit mode enables higher maximum values to be specified in some definition operands, as shown in Table 11-3.

Table 11-3: Operands with higher maximum value specifications (client environment definition)

Operand	Specification item	Max value in 32-bit mode	Max value in 64-bit mode
PDAGGR	Number of groups resulting from grouping	30,000,000	2,147,483,647
PDHASHTBLSIZE	Hash table size when application of hash execution for a hash join or a subquery is specified	524,288 kilobytes	2,097,152 kilobytes

11.3.3 64-bit-mode HiRDB client support range

With the 64-bit-mode HiRDB/Developer's Kit, you can create UAPs for both the 64-bit mode and the 32-bit mode. With the 64-bit-mode HiRDB/Run Time, you can create UAPs for both the 64-bit mode and the 32-bit mode. However, not all UAP languages can be used to create and execute UAPs in the 64-bit mode. Table 11-4 shows the client support range of 64-bit-mode HiRDB.

Table 11-4: 64-bit-mode HiRDB client support range

Language/functionality		Creation and execution of UAPs for 32-bit mode	Creation and execution of UAPs for 64-bit mode
Language	C	Y	Y
	C++	Y	Y
	COBOL	Y	N
	OOCOBOL	Y	N
XA interface		Y	N
Multi-connection facility		Y	Y*

Y: Can be created and executed.

N: Cannot be created or executed.

* Because, in 64-bit mode, kernel threading (native threading) is supported instead of DCE threading, pay particular attention to which libraries you link. For details about the libraries to link, see the *HiRDB Version 8 UAP Development Guide*.

11.4 Migrating from 32-bit mode to 64-bit mode

This section provides notes on migrating an application from the 32-bit mode to the 64-bit mode. For details on migrating from 32-bit-mode to 64-bit-mode HiRDB, see the *HiRDB Version 8 Installation and Design Guide*.

In order to migrate a HiRDB client from the 32-bit mode to the 64-bit mode, you must first install the 64-bit-mode HiRDB and set the environment appropriately; For details on HiRDB client environment setup, see the *HiRDB Version 8 UAP Development Guide*.

(1) Incompatible file types

For the most part, files used in the 32-bit mode can be used in the 64-bit mode. However, the following types of files created in the 32-bit mode are incompatible with and cannot be used in the 64-bit-mode HiRDB:

- Backup files
- HiRDB files comprising the master directory RDAREA and data directory RDAREA

(2) Operands with different default values

Changing HiRDB from the 32-bit mode to the 64-bit mode results in changes in the default values for the HiRDB system definition operands shown in Table 11-5.

Table 11-5: Operands with different default values

Operand	Specification item	Default in 32-bit mode	Default in 64-bit mode
pd_work_buff_size	Work table buffer size	<ul style="list-style-type: none"> • HiRDB/Single Server: 384* • HiRDB/Parallel Server: 1024* 	5120*
pd_fes_lck_pool_size	Lock pool size for front-end servers	$(\text{Value of } pd_max_users + 3) \times (\text{value of } pd_max_access_tables + 4) \div 6$	$(\text{Value of } pd_max_users + 3) \times (\text{value of } pd_max_access_tables + 4) \div 4$
SHMMAX	Maximum shared memory segment size	200 megabytes	1024 megabytes

* Value if the `pd_work_buff_mode` operand is omitted, or the default value if `pool` is specified. The default value does not change if `each` is specified in `pd_work_buff_mode`.

(3) Differences in memory requirements

Changing from the 32-bit mode to the 64-bit mode increases HiRDB's memory requirements. For details on the memory requirement calculations, see the *HiRDB Version 8 Installation and Design Guide*.

(4) Differences in UOC interface

Changing to the 64-bit mode requires changes in the UOC interface for the database load utility (`pdload`), with the result that the UOC must be re-created. For details on the UOC interface, see the section on the database load utility (`pdload`) in the *HiRDB Version 8 Command Reference*.

(5) Differences in SQL linkage areas

Changing to the 64-bit mode changes the configuration and sizes of the SQL linkage areas shown in Table 11-6. For details on the SQL linkage areas, see the *HiRDB Version 8 UAP Development Guide*.

Table 11-6: Changes in linkage areas

Linkage area	Size (bytes)	
	32-bit mode	64-bit mode
SQLCA	336	368
SQLCABC	4	8
SQLCODE	4	8
SQLERRD	4 × 6	8 × 6

(6) Changes in SQL descriptor areas

Changing to the 64-bit mode changes the configuration of the SQL descriptor areas, as well as the sizes and data types of the linkage descriptor areas, as shown in Table 11-7. For details on the SQL descriptor areas, see the *HiRDB Version 8 UAP Development Guide*.

Table 11-7: Changes in descriptor areas

Descriptor area	32-bit mode		64-bit mode	
	Size (bytes)	Data type	Size (bytes)	Data type
SQLDA	16 + 16 × n	—	24 + 24 × n	—
SQLDABC	4	—	8	—
SQLVAR	16 × n	—	24 × n	—

Descriptor area	32-bit mode		64-bit mode	
	Size (bytes)	Data type	Size (bytes)	Data type
SQLVAR_LOB	16 × n	—	24 × n	—
SQLLOBLN	—	long	—	int
SQLDATA	4	—	8	—
SQLIND	4	—	8	—
SQLLOBIND	4	long	8	int

n: Number of SQLVAR variables specified in the SQLN descriptor area.

—: No change in size or data type.

(7) Differences in SQL data types and C-language data descriptions

Because in UAPs written in C that support the 64-bit mode, the long-type is 8-byte long, the embedded variables that had used `long` will use `int` instead, and the C-language data descriptions listed in Table 11-8 will be changed. For details on the C-language data descriptions, see the *HiRDB Version 8 UAP Development Guide*.

Table 11-8: Changed C-language data descriptions

Data type of SQL		Data description in C	Data structure	Remarks
INTEGER	Simple type	<code>int variable-name;</code>	Variable	—
	Array type	<code>int variable-name [n];</code>	Array	$1 \leq n \leq 4096$
BLOB indicator variable		<code>int indicator-variable-name;</code>	—	—
SQL statement		<pre>struct{ int len; char str[n]; } variable-name;</pre>	Structure	—

—: Cannot be coded.

(8) Procedure for converting a 32-bit-mode UAP into a 64-bit-mode UAP

Following is the procedure for converting a 32-bit-mode UAP into a 64-bit-mode UAP.

1. In the declarations of embedded variables, replace all occurrences of the long type with the int type.

2. Execute UAP preprocessing, specifying the `/h64` option, which generates a 64-bit-mode post-source.
3. Compile the UAP, specifying the option that generates 64-bit-mode object code.
4. Link the UAP, specifying the 64-bit-mode HiRDB client library as the library to be linked.

Appendixes

- A. Functional Differences Between HiRDB Versions on Different Platforms
- B. Data Dictionary Tables
- C. HiRDB Client and HiRDB Server Connectivity
- D. Glossary

A. Functional Differences Between HiRDB Versions on Different Platforms

Table A-1 lists the functional differences between HiRDB versions on different platforms. Note that some facilities cannot be used, depending on the operating environment. For details, check the description of each facility.

Table A-1: Functional differences between HiRDB versions on different platforms

Item	Facility, operand, or command	HP-UX	Solaris	AIX 5L	Linux	Windows
Facilities	Facility for monitoring the memory size of server processes	Y	Y	Y	N	N
	Utility special unit	Y	Y	Y	Y	N
	Sharing of HiRDB system definitions	Y	Y	Y	Y	N
	Access from HiRDB VOS3 version clients	Y	Y	Y	Y	N
	Shell script for creating an unload statistics log file on a specific server machine (<code>pdstjacm</code>)	Y	Y	Y	Y	N
	Facility for parallel output of system logs	N	N	Y	N	N
Option program product related	Inner replica facility (HiRDB Staticizer Option)	Y	Y	Y	Y	N
	Updatable online reorganization (HiRDB Staticizer Option)	Y	Y	Y	Y	N
	HiRDB External Data Access facility that uses DB2 as a foreign server	Y	N	Y	N	N
Linkage to related products	Sun Java System Directory Server linkage facility	Y	Y	Y	N	Y
	Real Time SAN Replication	Y	N	Y	N	N
	Distributed database facility	Y	N	Y	N	N

Item	Facility, operand, or command	HP-UX	Solaris	AIX 5L	Linux	Windows
	Linkage to JP1/OmniBack II	Y	N	N	N	N
	HiRDB event registration with JP1/Base (or JP1/System Event Service)	Y	Y	Y	N	Y
	Linkage to JP1/Performance Management - Agent Option for HiRDB	Y	Y	Y	N	Y
	Linkage to OLTP (TPBroker)	Y	Y	N	N	Y
	Linkage to OLTP (TUXEDO)	N	Y	N	N	Y
	Linkage to OLTP (WebLogic Server)	Y	Y	N	N	Y
	Linkage to OLTP (TP1/EE)	N	N	Y	N	N
	EasyMT, MTguide	Y	N	Y	N	N
HiRDB system definitions	Fixing in shared memory pd_shmpool_attribute = fixed	Y	Y	Y	Y	N
	Global buffer page-fixing option function pd_dbbuff_attribute = fixed	Y	Y	Y	N	N
	Specification of the maximum number of consecutive abnormal terminations during HiRDB restart processing pd_term_watch_count	Y	Y	Y	Y	N
	Specification of the maximum number of files and pipes that can be accessed by HiRDB processes pd_max_open_fds	Y	Y	Y	Y	N
	Specification of the range of port numbers to use during communications processing pd_registered_port	N	N	N	Y	Y
	Specification of the maximum memory size to use for a server process pd_svr_castoff_size	Y	Y	Y	N	N

A. Functional Differences Between HiRDB Versions on Different Platforms

Item	Facility, operand, or command	HP-UX	Solaris	AIX 5L	Linux	Windows
	Specification of the maximum size of the communication buffers used for server intra-unit communication (TCP-UNIX domain) pd_ipc_unix_bufsize	Y	Y	Y	Y	N
	Specification of the maximum size of the communication buffers used for communication with the HiRDB clients (TCP-UNIX domain) pd_tcp_unix_bufsize	Y	Y	Y	Y	N
	Use of the no-cache access method pd_ntfs_cache_disable	N	N	N	N	Y
	Use of immediate acknowledgment between HiRDB servers pd_ipc_tcp_nodelayack	N	N	Y	N	N
Commands	pddbaset	Y	N	Y	N	N
	pddbchg	Y	Y	Y	Y	N
	pdgen	Y	Y	Y	Y	N
	pdgeter	Y	Y	Y	Y	N
	pditvstop	Y	Y	Y	Y	N
	pditvtrc	Y	Y	Y	Y	N
	pdlodsv	Y	Y	Y	N	N
	pdmemsv	Y	Y	Y	Y	N
	pdoobjconv	Y	Y	Y	N	N
	pdopsetup	Y	Y	Y	Y	N
	pdorbegin	Y	Y	Y	Y	N
	pdorcheck	Y	Y	Y	Y	N
	pdorchg	Y	Y	Y	Y	N
pdorcreate	Y	Y	Y	Y	N	

Item	Facility, operand, or command	HP-UX	Solaris	AIX 5L	Linux	Windows
	pdorend	Y	Y	Y	Y	N
	pdplgset	Y	Y	Y	Y	N
	pdrpause	Y	Y	Y	Y	N
	pdsetup	Y	Y	Y	Y	N
	pdsq1	Y	Y	Y	Y	N
	pdprgcopy	Y	Y	Y	Y	Y
	pdprgrenew	Y	Y	Y	Y	Y
	pdkill	N	N	N	N	Y
	pdntenv	N	N	N	N	Y
	Acquisition of a core file with the pdcancel command	Y	Y	Y	Y	N
	EasyMT date check with the pdload command	Y	N	Y	N	N
	Use of fixed-length tape with the pdcopy, pdload, pdrstr, and pdrogr commands	Y	Y	Y	N	Y
Client	ODBC Driver	N	N	N	N	Y
	OLE DB Provider	N	N	N	N	Y
	HiRDB.NET Data Provider	N	N	N	N	Y
	Third-party COBOL support facilities	N	Y	N	N	N
	XA interface for the XDM/RD E2 connection facility	Y	N	N	N	N
	Multi-connection facility (kernel threads)	Y	Y	Y	Y	N
	HiRDB Java stored procedure/function distribution wizard	N	Y	N	Y	Y
	Use of immediate acknowledgment between HiRDB servers and clients PDNODELAYACK	N	N	Y	N	N

A. Functional Differences Between HiRDB Versions on Different Platforms

Legend:

Y: Supported

N: Not supported

B. Data Dictionary Tables

HiRDB creates and manages the data dictionary tables that store table, index, and other types of definition information. Data manipulation SQL statements are used to reference data dictionary tables in order to check and confirm table, index, and other types of definition information.

Table B-1 lists the HiRDB data dictionary tables. For examples of SQL coding for referencing data dictionary tables and columns in the data dictionary tables, see the *HiRDB Version 8 UAP Development Guide*.

Table B-1: Data dictionary tables

No.	Table name	Contents	Information coverage (per row)
1	SQL_PHYSICAL_FILES	HiRDB file information (correspondence between HiRDB file system name and RDAREA names)	1 HiRDB file
2	SQL_RDAREAS	Information such as RDAREA name, definition information, RDAREA type, number of tables stored, and number of indexes	1 RDAREA
3	SQL_TABLES	Owner and table names of tables in a database (including dictionary tables)	1 table
4	SQL_COLUMNS	Definition information related to a column, such as column name and data type	1 column
5	SQL_INDEXES	Owner and index names of indexes in a database (including dictionary tables)	1 index
6	SQL_USERS	Authorization identifier of the user who granted a user execution privilege and database access	1 user
7	SQL_RDAREA_PRIVILEGES	RDAREA usage privilege status	1 RDAREA of 1 authorization identifier
8	SQL_TABLE_PRIVILEGES	Table access privilege granting status	1 table of 1 authorization identifier
9	SQL_VIEW_TABLE_USAGE	Name of base table on which a view table is based	1 view table

B. Data Dictionary Tables

No.	Table name	Contents	Information coverage (per row)
10	SQL_VIEWS	View definition information	1 view table
11	SQL_DIV_TABLE	Table division information (division condition and name of RDAREA for storage, as specified in CREATE TABLE)	1 table per <i>n</i> rows
12	SQL_INDEX_COLINF	Name of column to which an index is assigned	1 index per <i>n</i> rows
13	SQL_DIV_INDEX	Index division information (name of RDAREA for storage)	1 index per <i>n</i> rows
14	SQL_DIV_COLUMN	BLOB type column division information (name of RDAREA for storage, as specified in CREATE TABLE)	1 column per <i>n</i> rows
15	SQL_ROUTINES	Routine definition information	1 routine per row
16	SQL_ROUTINE_RESOURCES	Information on resources used in a routine	1 routine per <i>n</i> rows
17	SQL_ROUTINE_PARAMS	Parameter definition information in a routine	1 routine per <i>n</i> rows
18	SQL_ALIASES	Information used by the system (empty)	None
19	SQL_TABLE_STATISTICS	Table statistical information	1 table
20	SQL_COLUMN_STATISTICS	Column statistical information	1 column
21	SQL_INDEX_STATISTICS	Index statistical information	1 index
22	SQL_DATATYPES	User-defined type information	1 user-defined type
23	SQL_DATATYPE_DESCRIPTOR	Information on an attribute that comprises a user-defined type	1 attribute
24	SQL_TABLE_RESOURCES	Resource information to be used in a table	1 resource
25	SQL_PLUGINS	Plug-in information	1 plug-in
26	SQL_PLUGIN_ROUTINES	Plug-in routine information	1 plug-in routine
27	SQL_PLUGIN_ROUTINE_PARAMS	Parameter information of a plug-in routine	1 parameter information item
28	SQL_INDEX_TYPES	Index type information	1 index type

No.	Table name	Contents	Information coverage (per row)
29	SQL_INDEX_RESOURCES	Resource information to be used for an index	1 resource information item
30	SQL_INDEX_DATATYPE	Index item information	1 applicable information item (1 step)
31	SQL_INDEX_FUNCTION	Information on an abstract data type facility to be used for an index	Information on 1 abstract data type facility
32	SQL_TYPE_RESOURCES	Resource information to be used in a user-defined type	1 resource information item
33	SQL_INDEX_TYPE_FUNCTION	Information on an abstract data type facility that can be used by an index for which index type has been defined	1 index per <i>n</i> rows
34	SQL_EXCEPT	Information on index exception key values	Exception key group of 1 index per <i>n</i> rows
35	SQL_FOREIGN_SERVERS	DBMS information on foreign servers accessed by HiRDB when the HiRDB External Data Access facility is used	1 foreign server per row
36	SQL_USER_MAPPINGS	Mapping information on foreign servers accessed by HiRDB when the HiRDB External Data Access facility is used	1 mapping item for each user on HiRDB per row
37	SQL_IOS_GENERATIONS	Information used by the system (empty)	None
38	SQL_TRIGGERS	Trigger information in schemas	1 trigger per row
39	SQL_TRIGGER_COLUMNS	Event column list information on UPDATE triggers	1 event column data item per row
40	SQL_TRIGGER_DEF_SOURCE	Definition resource information on triggers	1 trigger definition source data item per <i>n</i> rows
41	SQL_TRIGGER_USAGE	Resource information referenced by trigger action conditions	1 resource name referenced by the trigger action conditions per row

B. Data Dictionary Tables

No.	Table name	Contents	Information coverage (per row)
42	SQL_PARTKEY	Information on partitioning keys of matrix-partitioned tables	1 partitioning key data item per row
43	SQL_PARTKEY_DIVISION	Information on partitioning condition values of matrix-partitioned tables	1 partitioning condition value data item per row
44	SQL_AUDITS	Information on audits	1 event data item for an object or user per row
45	SQL_REFERENTIAL_CONSTRAINTS	Match state of referential constraints	1 constraint per row
46	SQL_KEYCOLUMN_USAGE	Information on columns comprising foreign keys	1 column per row
47	SQL_TABLE_CONSTRAINTS	Information on integrity constraints in schemas	1 integrity constraint per row
48	SQL_CHECKS	Information on check constraints	1 check constraint per row
49	SQL_CHECK_COLUMNS	Information on columns used by check constraints	1 column used by a check constraint per row
50	SQL_DIV_TYPE	Information on partitioning keys in matrix partitioning tables that combine key range partitioning and hash partitioning	1 partitioning key per row
51	SQL_SYSPARAMS	Information on restrictions on number of consecutive certification failures and on password character string restrictions	1 setting item per row, or one permitted number of consecutive certification failures specifications or one password character string restriction specification per <i>n</i> rows

C. HiRDB Client and HiRDB Server Connectivity

HiRDB clients and HiRDB servers can be connected even though their versions or platforms are different. Tables C-1 and C-2 show HiRDB client and server connectivity.

A HiRDB server and a HiRDB client that have the right connectivity may not be able to actually connect if the client does not support compatible character encodings.

Table C-1: HiRDB client and HiRDB server connectivity (HiRDB server Version 5.0 or later)

HiRDB client version (type)		HiRDB server version (type)										
		Version 6 or later						Version 5.0				
		E	H	S	A	L	W	E	H	S	L	W
Version 6 or later	E	Y	Y	Y	Y	Y	Y	Y	Y	Y	—	—
	H	Y	Y	Y	Y	Y	Y	Y	Y	Y	—	—
	S	Y	Y	Y	Y	Y	Y	Y	Y	Y	—	—
	A	Y	Y	Y	Y	Y	Y	Y	Y	Y	—	—
	L	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	—
	W	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Version 5.0	E	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	H	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	S	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	A	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	L	Y	Y	Y	Y	Y	—	Y	Y	Y	Y	—
	W	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Version 4.0 04-03 or later	E	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	H	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	S	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	A	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	W	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

HiRDB client version (type)		HiRDB server version (type)										
		Version 6 or later						Version 5.0				
		E	H	S	A	L	W	E	H	S	L	W
Version 4.0 04-02 or older	E	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	H	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	W	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Version 03-03 or older	E	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	H	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	S	Y	Y	Y	Y	—	—	Y	Y	Y	—	—
	W	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
VOS3 client	V	Y	Y	—	—	—	—	Y	Y	—	—	—

E: HI-UX/WE2 version

H: HP-UX version

S: Solaris version

A: AIX version

L: Linux version

W: Windows version

V: VOS3 version

Y: Connectable

—: Not connectable

Table C-2: HiRDB client and HiRDB server connectivity (HiRDB server Version 4.0 or older)

HiRDB client version (type)	HiRDB server version (type)												
	Version 4.0									Version 03-03 or older			
	Version 04-03 or later					Version 04-02 or older							
	E	H	S	A	W	E	H	W	E	H	S	W	
Version 6 or later	E	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	H	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	S	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	A	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	L	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	W	Y	Y	Y	Y	Y	—	—	—	—	—	—	—
Version 5.0	E	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	H	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	S	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	A	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	L	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	W	Y	Y	Y	Y	Y	—	—	—	—	—	—	—
Version 4.0 04-03 or later	E	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	H	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	S	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	A	Y	Y	Y	Y	—	—	—	—	—	—	—	—
	W	Y	Y	Y	Y	Y	—	—	—	—	—	—	—
Version 4.0 04-02 or older	E	Y	Y	Y	Y	—	C	C	—	—	—	—	—
	H	Y	Y	Y	Y	—	C	C	—	—	—	—	—
	W	Y	Y	Y	Y	Y	C	C	C	—	—	—	—

HiRDB client version (type)	HiRDB server version (type)												
	Version 4.0									Version 03-03 or older			
	Version 04-03 or later					Version 04-02 or older							
	E	H	S	A	W	E	H	W	E	H	S	W	
Version 03-03 or older	E	Y	Y	Y	Y	—	Y	Y	—	C	C	C	—
	H	Y	Y	Y	Y	—	Y	Y	—	C	C	C	—
	S	Y	Y	Y	Y	—	Y	Y	—	C	C	C	—
	W	Y	Y	Y	Y	Y	Y	Y	Y	C	C	C	C
VOS3 client	V	Y	Y	—	—	—	—	—	—	—	—	—	—

E: HI-UX/WE2 version

H: HP-UX version

S: Solaris version

A: AIX version

L: Linux version

W: Windows version

V: VOS3 version

Y: Connectable

C: Limited connectivity; can be connected if the version of the HiRDB server is the same as or higher than the version of the HiRDB client. In other cases, cannot be connected except in the following cases:

- HiRDB client 04-01 can be connected to HiRDB server 04-00
- HiRDB client 03-02 can be connected to HiRDB server 03-01
- HiRDB client 02-06 can be connected to HiRDB server 02-05
- HiRDB client 02-04 can be connected to HiRDB server 02-03 (but not in the Windows NT version)
- HiRDB clients 02-05 and 02-06 can be connected to HiRDB server 02-03 (Windows NT version only)

—: Not connectable

D. Glossary

This Appendix defines terms used in the HiRDB manuals.

abstract data type

An abstract data type is a unique data type defined by the user so that data with a complex structure and operations involving it can be handled in SQL.

abstract data type column structure base table

A table containing abstract data type columns and from which the abstract data type columns have been removed is called an abstract data type column structure base table.

accepting unit

A unit in which is located a guest BES when the standby-less system switchover (effects distributed) facility is used.

access privilege

An access privilege is a permission that enables you to access a table; access privileges are set on a table-by-table basis. There are four access privileges:

- SELECT privilege
- INSERT privilege
- DELETE privilege
- UPDATE privilege

account lock period

Period during which a consecutive certification failures account lock state is to be in effect (will be detected by the connection security facility).

ADT (Abstract Data Type)

See *abstract data type*.

alias IP address

A technique by which different IP addresses can share the same LAN adapter through assignment of multiple IP addresses to the LAN adapter.

all asynchronous method

One of the processing methods used in Real Time SAN Replication. When an update of a database file or system file occurs at the main site, copying to the corresponding file at the remote site is performed asynchronously.

This term has no practical application for Windows users, because it is related to a

facility that cannot be used with a Windows version of HiRDB.

all synchronous method

One of the processing methods used in Real Time SAN Replication. When an update of a database file or system file occurs at the main site, copying to the corresponding file at the remote site is performed synchronously.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

alternate BES

A back-end server that takes over processing when an error occurs while the standby-less system switchover (1:1) facility is being used. Similarly, a unit containing an alternate BES is called an alternate BES unit.

asynchronous copy

One of the processing methods used to update-copy data to a remote site. Update-copying at the main site is completed without waiting for update-copying at the remote site to be completed.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

asynchronous group

A group made up entirely of asynchronous pair volumes.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

asynchronous pair volume

A pair volume that has been created by specifying `async` as the fence level. When data is written to the P-VOL, it is not mirrored synchronously onto the S-VOL. Thus, differences may arise between the data on the P-VOL and the data on the S-VOL, even though they are paired.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

asynchronous READ facility

When the prefetch facility is being used, the *asynchronous READ facility* sets up two prefetch buffers and, while the DB process is using one of the buffers, a READ process prefetches pages to the other buffer asynchronously with the DB process. By overlapping the DB processing with the prefetch input, you can reduce the processing time.

audit privilege

A privilege required for auditors. A user needs the audit privilege to perform operations such as the following:

- Loading data to audit trail tables
- Swapping audit trail files
- Searching, updating, and deleting audit trail files

audit trail

Operations performed on HiRDB such as audit commands and SQL execution records are output to a file. This record is called an audit trail, and the file to which this record is output is called an audit trail file.

audit trail table

A table used by the auditor to perform audits. It is created by loading into it information saved in an audit trail file.

auditor

A person who audits a HiRDB system. An auditor must be assigned the audit privilege. Only one auditor can be registered.

automatic log unloading facility

Normally, when system log information is unloaded by the HiRDB administrator using the `pdlogunld` command, it is also necessary to unload any system log file that is waiting to be unloaded. The facility that automates the process of unloading system log files is called the automatic log unloading facility.

automatic reconnect facility

When a connection to a HiRDB server is lost due to a server process going down, system switchover, network failure, or other cause, this capability automatically re-establishes the connection. By using the automatic reconnect facility, a user can continue executing the UAP without having to be aware that the connection to the HiRDB server was lost.

back-end server

A back-end server is a constituent element of a HiRDB/Parallel Server that performs database access, locking, and operational processing in accordance with execution instructions received from a front-end server.

back-end server for connecting to foreign servers

A back-end server that connects to foreign servers.

backup acquisition mode

The backup acquisition mode allows you to specify whether or not referencing and updating requests from other UAPs for an RDAREA being backed up are to be accepted while a backup is being made by the database copy utility. The backup acquisition mode is specified in the `-M` option of the database copy utility; there are three modes:

- Referencing/updating-impossible mode (`-M x` specification)
During backup, an RDAREA being backed up cannot be referenced or updated.
- Referencing-permitted mode (`-M r` specification)
During backup, an RDAREA being backed up can be referenced but not updated.
- Updatable mode (`-M s` specification)
During backup, an RDAREA being backed up can be both referenced and updated.

backup file

A backup file stores a copy of an RDAREA that can be used to recover the RDAREA after a failure.

backup-hold

When a backup is made using a command other than the `pdcopy` command (backup using a function provided by another product), an RDAREA to be backed up should be placed on backup-hold. An RDAREA on backup-hold can be backed up by the backup facility of another product while HiRDB is running. To apply backup-hold to an RDAREA, specify the `-b` option in the `pdhold` command. There are four types of backup-hold, as shown in Table D-1.

Table D-1: Types of backup-hold

Types of backup-hold	Explanation
Referencing-permitted backup-hold	During backup-hold, RDAREAs that have been backed up and held can be referenced; an updating attempt will result in an SQL error (-920).
Referencing-permitted backup-hold (update <code>WAIT</code> mode)	During backup-hold, RDAREAs that have been backed up and held can be referenced; an updating attempt goes onto lock-release wait status until the backup-hold is released.
Updatable backup-hold	During backup-hold, RDAREAs that have been backed up and held can be referenced and updated. Even while an updating transaction is being executed, an RDAREA is placed immediately on updatable backup-hold status without placing the <code>pdhold</code> command on wait status.

Types of backup-hold	Explanation
Updatable backup-hold (<code>WAIT</code> mode)	During backup-hold, RDAREAs that have been backed up and held can be referenced and updated. If an updating transaction is being executed, the <code>pdhold</code> command is kept waiting until the transaction terminates.

The following types of backup-hold are referred to collectively as *committing a database*:

- Referencing-permitted backup hold
- Referencing-permitted backup hold (update `WAIT` mode)

base row

The row that contains the base data for all columns. If a branch row has been created, the base row contains the branch target information.

BLOB (Binary Large Object)

BLOB is large data, such as large text, image, and audio data.

BLOB and BINARY data addition update and partial extraction facility

A facility that consists of the two functions described below. Data with a defined length of 32,001 bytes or more is handled as `BINARY` data.

- Function for specifying a concatenation operation in the `UPDATE` statement's `SET` clause in order to add new data to registered `BLOB` or `BINARY` data
- Function for specifying the `SUBSTR` scalar function in order to extract only a specified portion of `BLOB` or `BINARY` data

This facility can reduce the memory requirement to the amount of `BLOB` or `BINARY` data that is to be added or extracted.

BLOB data file output facility

With this facility, instead of retrieved `BLOB` data being returned to the client, it is output to a file in a unit that contains either a single server or a front-end server, and only the name of the file is returned to the client. Use of this facility prevents large increases in memory use during data retrieval operations.

block transfer facility

The block transfer facility transfers data from any number of rows from a `HiRDB` server to a `HiRDB` client.

branch row

Data is partitioned onto another page when its length exceeds one page. Rows that are stored on the other page in this manner are called *branch rows*.

changing the partition storage conditions

Use of `ALTER TABLE` to change the partition storage conditions for a key range partitioned table. By changing a table's partition storage conditions, RDAREAs containing old data can be reused, thereby saving time.

Note:

You can use `ALTER TABLE` to change the partition storage conditions of tables that have been partitioned using one of the following methods:

- Boundary value specification
- Storage condition specification (when the equal sign (=) was used as the storage condition comparison operator)

character code set

The sets of character codes that can be used in the UNIX version of HiRDB and in the Windows version of HiRDB are somewhat different. The following table lists the character code sets that can be used on each platform:

Character code set	UNIX version	Windows version
<code>sjis</code> (Shift JIS Kanji Code)	Y	Y
<code>ujis</code> (EUC Japanese Kanji Code)	Y	N
<code>chinese</code> (EUC Chinese Kanji Code)	Y	Y
<code>utf-8</code> (Unicode)	Y	Y
<code>lang-c</code> (Single Byte Character Code)	Y	Y

Legend:

Y: Can be used

N: Cannot be used

check constraint

A constraint used to maintain data integrity in tables by checking specified constraint conditions when data is inserted or updated and suppressing operations on data that does not meet the conditions.

check pending status

HiRDB limits the data operations that can be performed on referencing tables and check constraint tables when the integrity of table data can no longer be guaranteed due to execution of SQL code or a utility on a table in which a referential constraint or a check constraint is defined. This status indicates such a state, in which data operations have been limited because data integrity cannot be guaranteed.

closing RDAREAs

Process of prohibiting accesses from HiRDB to RDAREAs (closing the RDAREAs). To execute a utility that re-creates RDAREAs, the RDAREAs must first be closed.

When an RDAREA is closed, some RDAREA definition information retained in memory by HiRDB and data storage information are discarded from memory and must be re-created the next time the RDAREA is opened.

For example, when `pdmod` is to be used to re-initialize or delete RDAREAs or `pdrstr` is to be used to recover a database, the RDAREAs must first be closed.

To close an RDAREA whose open attribute is not `SCHEDULE`, the RDAREA must be shut down before it can be closed.

cluster key

A cluster key is a table column that is specified as the key for storing rows in ascending or descending order of the values in the specified column.

compiling

Compiling is the process of converting a post-source program created by an SQL preprocessor in order to create a machine language program (object code).

concurrent connections, maximum number of

The number of connections that can be established at the same time to a HiRDB server (single front-end server in a HiRDB/Parallel Server system); this value is specified in the `pd_max_users` operand. Connection requests that exceed the maximum number of concurrent connections result in a `CONNECT` error.

connection frame guarantee facility for client groups

Facility that groups clients connected to HiRDB and guarantees the maximum number of HiRDB connections allowed for each client group.

connection security facility

Facility for enhancing password security. This facility enables the user to set a minimum size for passwords (in bytes) in order to prohibit use of simple or easily-guessed passwords. It can also set a maximum number of consecutive unsuccessful attempts to enter a password (consecutive certification failures).

consecutive certification failure account lock state

If user authentication fails because of too many consecutive failures to enter the correct password, that user is placed in consecutive certification failure account lock state. A user in this status can no longer establish connection with HiRDB (`CONNECT`).

consistency group

A group that guarantees maintenance of the update order integrity of S-VOLs.

Integrity is maintained between the order in which data is written on the P-VOL and the order in which data is updated to the S-VOL on all asynchronous pair volumes that make up the consistency group.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

constructor function

The constructor function is used to generate an abstract data type value.

cursor

A cursor is used during data retrieval or updating to maintain the most recent extraction position, in order to extract multiple rows of a retrieval result one row at a time.

`DECLARE CURSOR` is used to declare a cursor, the `OPEN` statement is used to open a cursor, the `FETCH` statement is used to retrieve search results and advance the cursor to the next row, and the `CLOSE` statement is used to close a cursor.

database

A database is a system that organizes systematically tables, indexes, user-defined types, stored procedures, stored functions, and access privileges that are created for processing jobs.

data definition language

A language for defining the structure and contents of a database.

data dictionary

A data dictionary stores database design specifications, including database table structures, column definitions, and index definitions. In the case of a HiRDB/Single Server, the single server provides centralized management of the data dictionary. With a HiRDB/Parallel Server, centralized management of the data dictionary is provided by the dictionary server.

data dictionary LOB RDAREA

A data dictionary LOB RDAREA stores either stored procedures or stored functions. The two data dictionary LOB RDAREAs are one for storing the definition sources and one for storing the objects of stored procedures or stored functions.

data directory RDAREA

A data directory RDAREA stores dictionary tables for managing the analysis results of a definition SQL and the dictionary table indexes.

data loading

Data loading is the process of storing data in a table. Data loading is performed by the database load utility (`pdload`).

data loading with synchronization points set

Normally, while data is being loaded, transactions cannot be reconciled until the storage processing of all the data has been completed. This means that synchronization point dumps cannot be obtained during execution of the database load utility. If HiRDB terminates abnormally during loading of a large quantity of data, it will take a long time to restart HiRDB. To resolve this problem, you can set synchronization points for any number of data items during data loading in order to reconcile transactions. This is called data loading with synchronization points set.

To perform data loading with synchronization points set, you must specify a synchronization point lines count, which is the number data items to be stored before a synchronization point is set. This value is specified in the `option` statement of the database load utility. When a large amount of data is to be loaded into a table, you should consider whether or not to execute data loading with synchronization points set.

Synchronization points can also be specified in the database reorganization utility; this is called reorganization with synchronization points set.

data manipulation language

A language for specifying database operations so that a business application program can perform operations on the data in a database.

data warehouse

A data warehouse is the database system concept that enables databases residing at a mainframe, UNIX servers, and PC servers to be linked, so that they can be accessed in a manner transparent to the various different operating systems. With a data warehouse, end-users can access a database from PCs and the system administrator can access it by means of a system analysis tool, so that the data warehouse provides to each the necessary database management functions. The replication functions (HiRDB/Datareplicator and HiRDB Dataextractor) that can be used in data warehouse databases are available in HiRDB.

deadlock

Deadlock occurs when multiple transactions are competing for the same resources and are waiting for each other to release their resources.

default constructor function

The default constructor function is created automatically by HiRDB and identified with the same name as an abstract data type; the default constructor function does not take any arguments.

default RD node

The HiRDB system assumed as the connection target in a client environment definition (when using the `CONNECT` statement to connect without specifying an RD node name). DBMS nodes other than HiRDB are called *distributed RD nodes*.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

deferred write processing

Deferred write processing writes pages that have been updated in the global buffer onto the disk when the number of updated pages reaches a certain value, rather than when a `COMMIT` statement is issued. The point at which the number of updated pages reaches the certain value (as determined by HiRDB) is called the deferred write trigger.

delayed batch creation of plug-in index

Instead of adding plug-in index data when row data is added to a table, the plug-in index delayed batch creation facility can be used to add the plug-in index data in a batch later with the database reorganization utility.

This facility is used for adding (or updating) a large volume of row data to a table for which a plug-in index is defined.

delayed rerun

Delayed rerun is the mode that enables rollback and acceptance of new transactions to be started simultaneously during system recovery.

dictionary server

The server that centrally manages data dictionaries that contain database definition information is called the dictionary server.

differential backup facility

The differential backup facility backs up only the information that has changed since the previous backup was made, which reduces the amount of time required to make a backup. Consider using the differential backup facility when the database is large and the amount of data updating has been relatively small.

differential backup management file

A file required to use the differential backup facility. A differential backup management file stores information obtained when differential backups are made. HiRDB uses this file when backups are made and during recovery of a database using backups.

Directory Server linkage facility

A facility (such as Sun Java System Directory Server) that is used to manage and authenticate HiRDB users. This capability is provided by a process called the Directory Server linkage facility. By using a directory server, you can centrally manage organizational and user information (user IDs, passwords, affiliations, and job titles) that is otherwise managed separately in HiRDB, Groupmax, and other systems.

distributed client

A distributed client is a source of requests to access a remote database (a node with a DBMS that a UAP accesses directly).

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

distributed client facility

A facility that enables the HiRDB system on the local node to function as a distributed client, thus providing remote database access to a distributed server.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

distributed database facility

Use of a remote database access facility provided by the DF/UX distributed database products allows the user to access other database systems to retrieve and update data. There are two remote database access facilities:

- Distributed client facility
- Distributed server facility

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

distributed nest-loop-join

A distributed join in which nested-type loop processing is performed on retrieved rows that match a join condition: more specifically, nested-type processing is conducted by retrieving each row from a particular table, issuing an SQL statement for each row, and matching up the row to the other row of the counterpart table residing on a foreign server.

distributed RD node

RD nodes that are not default RD nodes. Nodes on DBMSs other than HiRDB are referred to as *distributed RD nodes*.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

distributed server

A distributed server is a node (with a database) that accepts requests from distributed clients and performs operations on the database.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

distributed server facility

A facility that enables the HiRDB system on the local node to function as a distributed server, so that it can accept requests from distributed clients and perform database operations.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

dynamic SQL

A method of generating SQL while a UAP is executing. Contrast with *static SQL*, which is a method of coding SQL within a program when a UAP is created.

dynamic update of global buffer

Dynamic update of global buffer refers to the ability to add, change, and delete global buffers with the `pdbufmod` command while HiRDB is running.

embedded UAP

An embedded UAP is a UAP in which an SQL is written directly into a source program that is written in a high-level language (C or COBOL).

encapsulation

The process of separating an abstract data type interface from its implementation method and keeping this implementation method hidden from the user is called encapsulation.

exceptional key value

An exceptional key value is a key value that is to be excluded in order not to create a useless index key during index definition. If an index has the null value in all rows of the column or has key values that are duplicated multiple times, the null value can be specified as the exceptional key value.

facility for conversion to a decimal signed normalized number

The facility for conversion to a decimal signed normalized number converts the sign part of signed packed-format data (decimal, date interval, time interval) that has been input; the conversion is performed in accordance with the rules described in Section 7.7 *Facility for conversion to a decimal signed normalized number*. The process of converting a sign is called *normalization of the sign part*.

facility for monitoring abnormal process terminations

A facility that monitors the number of times a server process terminates abnormally over a given period of time and terminates HiRDB (or the unit in the case of a HiRDB/Parallel Server) abnormally when the number of server abnormal terminations exceeds the value specified in the `pd_down_watch_proc` operand. You should use this facility in conjunction with the system switchover facility. Because this facility

terminates HiRDB abnormally when a specified number of server process abnormal terminations occurs, the system switchover facility will provide rapid system switchover. If you do not use this facility, the system will not be switched over in such a situation because HiRDB will not terminate abnormally.

facility for parallel writes in deferred write processing

A facility for performing deferred write processing by executing multiple processes in parallel, which reduces the amount of time required for write processing.

facility for predicting reorganization time

Typically, the user must determine the necessity for and timing of table or index reorganization or RDAREA expansion on the basis of the results of executing the `pddbst` command. This sometimes results in tables being reorganized unnecessarily or in tables not being reorganized when needed because of human error (such as failure to read messages that have been output).

To simplify these operations, the facility for predicting reorganization time enables HiRDB to predict when reorganization should be performed.

facility using arrays

Array-type variables enable multiple processes to be executed by a single execution of an SQL statement. Use of this facility reduces the number of communications between HiRDB client and server.

The facility using arrays can be applied to the `FETCH`, `INSERT`, `UPDATE`, and `DELETE` statements.

falsification prevention facility

Facility that prevents updating of table data other than the data in updatable columns in order to protect table data from errors or unauthorized tampering. Tables on which the falsification prevention facility is applied are called falsification-prevented tables. When the falsification prevention option is specified, the only operations permitted on the table are row addition, retrieval, deletion of data whose deletion-prevented duration has expired, and updating of updatable columns. A table definition can also be changed to convert an existing falsification-unprevented table into a falsification-prevented table. If no deletion-prevented duration is specified, the table and its data cannot ever be deleted.

fence level

A pair volume copying method or a level for guaranteeing the integrity of data on pair volumes when an update-copy error occurs. The fence levels are `data`, `never`, and `async`. For details about the levels, see the RAID Manager documentation.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

FIX attribute

FIX is the attribute assigned to a table whose row length is fixed.

FIX attribute table

A table whose row length is fixed.

FIX hash partitioning

FIX hash partitioning is a method of row-partitioning a table using a hash function that divides the table by row and stores the values of the columns that comprise the table evenly among RDAREAs. The column specified as the condition for row partitioning of the table is called the partitioning key. When FIX hash partitioning is used, HiRDB recognizes the portion of the table that is partitioned into each RDAREA.

Consequently, during retrieval processing, only those back-end servers in which the applicable data is presumed to be located must become retrieval targets.

flexible hash partitioning

Flexible hash partitioning is a method of row-partitioning a table using a hash function and dividing the values of the columns that comprise the table evenly among RDAREAs. The column specified as the condition for row partitioning of the table is called the partitioning key. When flexible hash partitioning is used for partitioning and storage of a table in RDAREAs, it is not possible to tell which portion of the table is in which RDAREA. Consequently, during retrieval processing, all back-end servers in which the applicable table is located must become retrieval targets.

floatable server

A floatable server is a back-end server that is dedicated to sort and merge operations that involve a heavy processing load; it is not used to access table data. Use of floatable servers improves the overall performance of HiRDB. Table data is not stored in a back-end server that is assigned as a floatable server. A back-end server that does not perform data accesses due to the nature of a particular transaction can be used temporarily as a floatable server. A floatable server can be set up only on a HiRDB/Parallel Server.

foreign index

Definition information of an index that is defined for a table in a foreign database that is mapped to a foreign table. Foreign indexes are used during optimization. The actual index is not on HiRDB. It can be defined when you are using the HiRDB External Data Access facility.

foreign key

One or more columns defined for a referencing table. The value of a foreign key is restricted in such a manner that it takes either the same value as the primary key that references it or a value that includes the null value.

foreign server

An external DBMS that is accessed from HiRDB by using the HiRDB External Data Access facility. Once a foreign server has been defined in HiRDB, HiRDB can access foreign tables.

foreign table

A table defined in HiRDB based on the definition information for a table on a foreign server. This table is needed to access tables on foreign servers. Note that HiRDB manages only the definition information for the table; the foreign server manages the actual table. It can be defined when you are using the HiRDB External Data Access facility.

free page

A page in which no data is stored. A free page may be either a used free page or an unused free page.

free page reuse mode

A method of searching for free space in an RDAREA when HiRDB is saving data. When a segment becomes full, a search is conducted to find free space in used pages before an unused segment is allocated. The next search start position is also remembered, so that, the next time, searching for free space begins at this position.

free segment

A segment in which no data is stored. A free segment may be either a used free segment or an unused free segment.

free space reuse facility

A facility by which free space in used segments is utilized once the number of segments specified by the user is reached and the last segment becomes full. This is done by switching the page search mode to the free page reuse mode. Once the free space in all the segments of the specified number of segments runs out, the free space reuse facility activates the new page allocate mode, and a new unused segment is allocated.

front-end server

A front-end server is a constituent element of a HiRDB/Parallel Server that determines the method of database access to be used from an executed SQL and provides execution instructions to back-end servers.

global buffer

A buffer that is used for input and output of data stored in RDAREAs on disk. Global buffers are allocated in common memory. Global buffers are always allocated to an RDAREA or an index.

A LOB global buffer is allocated in the following cases:

- When a plug-in index is being stored
- When benefits can be expected from buffering because there is a relatively small amount of data
- When LOB data that will be accessed frequently is being stored

global buffer pre-writing

A facility that writes specified tables and indexes into global buffers in advance. This improves the buffer hit rate.

global deadlock

A form of deadlock that occurs between servers in a HiRDB/Parallel Server configuration.

guest area

Resources for a back-end server that is associated with a guest BES.

guest BES

A back-end server whose processing has been moved to another unit by the standby-less system switchover (effects distributed) facility in the aftermath of an error. The unit for a guest BES is called the accepting unit.

HA monitor

A cluster software program for implementing a system switchover facility. It is packaged with the Hitachi HA Toolkit Extension.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

hash facility for hash row partitioning

The addition of RDAREAs to accommodate an increase in the amount of data in a hash-partitioned table (increasing the number of row partitions in a table) can create an imbalance in the amount of data in the existing RDAREAs and in the new RDAREAs. The hash facility for hash row partitioning can correct any such imbalance resulting from an increase in the number of table row partitions.

The hash facility for hash row partitioning can be applied to both FIX hashing and flexible hashing.

heterogeneous system configuration

Typically with a HiRDB/Parallel Server, all units must be based on the same platform. However, if certain conditions are satisfied, a HiRDB/Parallel Server system made up of units on different platforms can be configured. This type of configuration is called a *heterogeneous system configuration*.

HiRDB administrator

Of the users who log onto the operating system using a system administrator user ID, the HiRDB administrator is the user who has the privilege to manipulate HiRDB itself. The HiRDB administrator's privilege permits execution of HiRDB commands and conveys ownership of HiRDB directories and files.

HiRDB client

A workstation or personal computer on which HiRDB/Developer's Kit or HiRDB/Run Time has been installed.

HiRDB External Data Access Adapter

A product that handles the interface differences with a foreign server (an external DBMS). The following HiRDB External Data Access Adapters are available:

- HiRDB External Data Access Adapter for HiRDB and XDM/RD E2
- HiRDB External Data Access Adapter for ORACLE

The HiRDB External Data Access Adapter functionality for HiRDB and XDM/RD E2 is included in HiRDB External Data Access.

HiRDB External Data Access facility

A facility for accessing external DBMSs (ORACLE, and so on) from a HiRDB interface. This facility provides the following benefits:

- Enables you to view and update in a single table information from databases built with multiple DBMSs residing at multiple locations.
- Even when there are multiple environments on which the DBMSs are running, you can access tables from the HiRDB interface.

HiRDB file

A file that is used by HiRDB. A HiRDB file is composed of contiguous segments in a HiRDB file system area. It stores tables, indexes, and information necessary to recover the status of the system in the event of an error.

HiRDB file system area

An area in which HiRDB files are created. Separate HiRDB file system areas are created for different application purposes, for system files, for work table files, and for RDAREA files.

HiRDB.ini file

A file that contains information required by utilities that execute on HiRDB clients and for the HiRDB SQL Executer to connect to a HiRDB server. This file must be available on both the server PC and the client PC.

HiRDB.NET data provider

Data provider required in order to access databases from applications compatible with ADO.NET. The HiRDB.NET data provider complies with the ADO.NET specifications.

HiRDB system definition files

The HiRDB system definition files store the HiRDB system definition information that is used to determine the HiRDB operation environment (system configuration, control information, and each server's execution environment).

host BES

A back-end server that is defined on an applicable unit by the standby-less system switchover (effects distributed) facility. The unit for a host BES is called a regular unit.

Hub optimization information definition

A definition file that specifies whether or not queries that target a foreign server are to be executed on the foreign server and that provides cost parameters related to optimization. Use of this definition file is optional, but it is recommended that the provided sample file be used without modification.

hybrid method

One of the processing methods used in Real Time SAN Replication. When an update of a system file at the main site occurs, copying to the corresponding file at the remote site is performed synchronously. When an update of a database file at the main site occurs, copying to the corresponding file at the remote site is performed asynchronously.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

incorporation during commit

Incorporation during commit is the process of writing pages that have been updated in the global buffer onto the disk when a `COMMIT` statement is issued.

index

An index is assigned to a column as a key for retrieving the table. An index consists of a key and key values. The key is the column name that describes the contents of the column, and the key values are the actual values in the column.

An index can be either a single-column index or a multicolumn index. A single-column index is based on a single column of a table; a multicolumn index is based on more than one column of a table.

index page split

HiRDB indexes have a B-tree structure. Consequently, an index page split occurs when

there is no free area available in an index page to which a key is to be added. An index page split occurs when HiRDB splits the index information of an index page in order to allocate a free area and moves the second half of this information to a new page.

index reorganization

Rearrangement of an index based on information derived by retrieving index key information to create an index information file. Index reorganization can be performed by index or by index RDAREA.

Over time, deleting (DELETE) and updating (UPDATE) data degrades an index's storage efficiency, which reduces performance when the index is used for searches. Index reorganization is performed to minimize such performance degradation.

inheritance

Inheritance is the process by which a lower-order abstract data type inherits attributes and routines defined in a higher-order abstract data type.

inheriting a database

The operations after a site switchover has occurred that involve shutting down HiRDB after the log application processing has been completed and the site status has been changed. Inheriting a database is performed with the `pdrisedbto` command.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

inner replica facility

A facility by which duplicate RDAREAs (replica RDAREAs) are defined and manipulated. Using a disk system or software that provides mirroring, the inner replica facility allows you to access the duplicated database. For details about the inner replica facility, see the *HiRDB Staticizer Option Version 7 Description and User's Guide*.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

integrity constraint

An integrity constraint is a constraint that is available for guaranteeing the validity of data in a database.

There are two integrity constraints:

- NOT NULL constraint (does not permit the null value to be set in a specified column)
- Uniqueness constraint (requires that the values in a specified column be unique and does not permit value duplication in the column)

interface area

An area used for exchanging information between HiRDB and UAPs. HiRDB provides the following seven interface areas:

- SQL Communications Area
- SQL descriptor area
- Column Name Descriptor Area (SQLCNDA)
- Type Name Descriptor Area (SQLTNDA)
- Embedded variables
- Indicator variables
- Parameters

inter-process memory communication facility

When a HiRDB server and a HiRDB client are running on the same server machine, memory is used for inter-process communication to increase communication speed between the HiRDB server and the HiRDB client. This functionality is provided by the *inter-process memory communication facility*, which is used by specifying `MEMORY` in the `PDIPC` operand of the client environment definition.

IP address

An address used by the IP protocol.

Java stored function

A Java stored function is a stored function in which procedures are coded in Java.

Java stored procedure

A Java stored procedure is a stored procedure in which procedures are coded in Java.

JDBC driver

A driver created for HiRDB based on the interface defined in the JDBC standard. The JDBC driver is needed to execute Java stored procedures and Java stored functions. It is installed by selecting it when you install a HiRDB client.

join

Processing that combines two or more tables.

JP1

JP1 is a series of products that provide functions for batch job operations, automatic system operation, document output control, and file backup.

JP1 is used to automate system operations and reduce labor costs.

key range partitioning

Key range partitioning is a method of table partitioning that divides a table into groups of rows, using ranges of values in a column of the table as the partitioning condition. The column specified as the condition for row partitioning is called the partitioning key.

LAN adapter

Hardware that converts data for connecting a computer to a LAN.

language compiler

A program that converts a post-source program created by an SQL preprocessor into a machine-language program (object).

large file

A HiRDB file system area of at least 2 gigabytes in size is called a large file. Normally, the maximum size of a HiRDB file system area is 2047 megabytes (approximately 2 gigabytes). A HiRDB file system area whose size needs to exceed this limit must be created as a large-file HiRDB file system area. The maximum size of a HiRDB file system area created as a large file is as follows:

- HP-UX version: 131,071 megabytes
- Solaris, AIX 5L, or Linux version: 1,048,575 megabytes

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

linkage

The process of creating a single executable program file (load module) by combining the object modules created by a language compiler.

linker

A program that combines the object modules created by a language compiler into a single executable program file (load module).

list

A set of data items saved under a single name (list name). Lists include sets of data items that may be saved with temporary names at intermediate stages during the process of filtering information in a series of hierarchical steps based on specification of conditions that are designed to retrieve an appropriate number of data items.

list RDAREA

Lists created by the `ASSIGN LIST` statement are stored in a list RDAREA. A list RDAREA is created when a narrowed search is performed.

load module

A UAP source that has been preprocessed, compiled, and linked into a single executable file.

LOB column structure base table

A table containing LOB columns from which the LOB data has been removed is called a LOB column structure base table. Data can be loaded and the database can be reorganized by LOB column structure base table or LOB column.

LOB data

Large variable object data such as documents, images, and sounds is called LOB data. LOB data is stored in user LOB RDAREAs. Data loading and database reorganization can be performed separately from the user RDAREA that stores the LOB column structure base table.

local buffer

A buffer that is used for input and output of data stored in RDAREAs on disk. Local buffers are allocated in process private memory.

locator facility

Facility that reduces a client's memory resource overload and data transfer volume during BLOB or BINARY data searches.

A locator is a 4-byte data value that is used to identify data at a server. When a locator-embedded variable is specified, such as in the INTO clause of a single-row SELECT statement or FETCH statement, the value of the locator indicating the data, not the data itself, is received as the search result. Processing that handles the data identified by a locator can be performed when a locator-embedded variable is specified in other SQL statements.

lock

A form of control managed by HiRDB to maintain database integrity.

locking

Locking is a type of control managed by HiRDB in order to maintain database integrity.

log acquisition mode

The log acquisition mode is one of the modes for acquiring a database update log during execution of a UAP or utility. This mode acquires the database update log that will be needed for rollback or rollforward when the UAP or utility is updating the contents of an RDAREA.

log application

In the log-only synchronous method, update processing of a database based on system logs copied from the transaction execution site. Log application is performed at the log application site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

log application not possible status

The state in which integrity of the databases at the transaction execution site and the log application site is not established or the information required for log application cannot be copied correctly. If a natural disaster occurs while the system is in this status, data will be lost if the transaction execution site goes down.

When the preparations for system log application are performed while the system is in log application not possible status, the system is placed in log application possible status.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

log application possible status

The state in which integrity of the databases at the transaction execution site and the log application site is established and the information required for log application can be copied correctly. If a natural disaster occurs while the system is in this status, job processing can be transferred to the log application site without loss of data, even if the transaction execution site goes down.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

log application site

In the log-only synchronous method, the site at which update processing of databases is performed based on system logs copied from the transaction execution site. As with the transaction execution site, the log application site must be operational at all times.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

LVM

A disk device management facility that can create and manage multiple virtual devices from a set of one or more real devices. LVM enables the user to achieve disk device management with high availability and management simplicity, such as by dividing a single real device into multiple small virtual devices or by grouping multiple real devices into a single large virtual device. LVM also provides functions for improving performance and achieving redundancy.

main site

Site where the system normally used by Real Time SAN Replication is located.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

master directory RDAREA

The master directory RDAREA stores information on the RDAREAs that store dictionary tables and user-created tables and indexes; it also stores RDAREA registration location (server) information.

matrix partitioning

A method of partitioning a table by a combination of partitioning methods using two of the table columns as the partitioning key. 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 always 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. A table partitioned by the matrix partitioning method is called a matrix partitioning table.

message queue monitoring facility

A facility that monitors for the server process not responding state. HiRDB uses the message queue during server process allocation processing. If a server process not responding state occurs, messages can no longer be fetched from the message queue. If messages cannot be fetched from the message queue after a certain amount of time has elapsed (called the message queue monitoring time), HiRDB outputs a warning message or error message (KFPS00888-W or KFPS00889-E). Output of one of these messages indicates the possibility that a server process not responding state may have occurred.

module trace

A type of troubleshooting information used for isolating errors. The executed functions and macro history of most HiRDB processes are recorded in a process-specific memory. If a process terminates abnormally and a core file is output, the contents of the process-specific memory are output to the core file, which allows a module trace to be obtained from the core file.

monitor mode

The system switchover facility can be operated in either the monitor mode or the server mode. In the monitor mode, only system failures are monitored. In the server mode, both system failures and server failures (such as HiRDB abnormal termination) are monitored. System switchover can take less time in the server mode than in the monitor mode because a standby HiRDB is started beforehand in the server mode. The

system switchover facility must also be operating in the server mode to use the following functions:

- user server hot standby
- rapid system switchover facility
- standby-less system switchover facility

monitoring free area for system log file

A facility that outputs a warning message or suspends scheduling of new transactions and forcibly terminates all transactions on the server when the percentage of free space in the system log file drops below a warning value.

multi-connection facility

The multi-connection facility enables a UAP running on a HiRDB client to connect simultaneously to multiple HiRDB servers.

multi-HiRDB

Multi-HiRDB is the mode in which multiple HiRDB systems run on a single server machine.

multiple front-end servers

The configuration in which multiple front-end servers are set up is used when the CPU load for SQL processing is too heavy for a single front-end server.

narrowed search

A narrowed search is the process of narrowing in steps the records retrieved in successive searches.

A narrowed search requires creation of a list using the `ASSIGN LIST` statement of the data manipulation SQL. When such a list is created under appropriate conditions, its use can reduce processing time. If there are multiple conditions, multiple lists can be combined for the search.

NetBackup linkage facility

A facility designed to create backup files for use by the database copy utility (`pdcopy`) or database recovery utility (`pdrstr`) on a medium managed by NetBackup Server.

new page allocate mode

A method of searching for free space in an RDAREA when HiRDB is saving data. When a segment becomes full, first, a new unused segment is allocated. If the RDAREA runs out of unused pages, a search is then conducted to find free space in a segment already allocated to that table (in a used segment), starting at the beginning of the first used segment. Once free space is found, the data is saved to that free space.

node

In a distributed system, each communication control device making up the network, and the systems connected to these devices.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

no-log mode

The no-log mode is one of the modes for acquiring a database update log during execution of a UAP or utility. This mode does not acquire a database update log when the UAP or utility is updating the contents of an RDAREA.

normal BES

A back-end server whose processing is transferred to another server in the event of an error, through use of the standby-less system switchover (1:1) facility. Similarly, a unit containing a normal BES is called a normal BES unit.

no-split option

Normally, the data part of a variable-size character string that is at least 256 bytes in length is stored in different pages. The no-split option allows you to store one row of data in one page even when the actual data size of a variable-size character string is 256 bytes or greater. The no-split option improves the data storage efficiency. The no-split option is used by specifying `NO SPLIT` in the `CREATE TABLE` or `CREATE TYPE` statement.

NOT NULL constraint

The `NOT NULL` constraint is a limitation that does not permit the null value to be set in a column.

no transaction loss (no data loss)

With no transaction loss, update processing on transactions committed at the main site is guaranteed to be mirrored to the database at the remote site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

null value

The null value is a special value that indicates that no value has been set.

object

A program that has been translated into machine language by a language compiler.

object relational database

An object relational database is a database that has been expanded by incorporating the concept of object orientation into a relational database model. HiRDB can handle data

that has a complex structure, such as multimedia data, and the operations for such data as a unified object and can manage it in a database.

operation releasing check of unload status

A system log file can become a swap target only if it is in one of the following two statuses:

- Overwriting enabled status
- Extraction completed status (HiRDB Datareplicator)

The unloading status is unrelated to the statuses required in order for a system log file to be a swap target. Performing this operation eliminates the need to unload or release system log files.

operation unloading system log

The operation of unloading system logs. When a database is recovered, unload logs that have been unloaded from the system logs are used as the input to the database recovery utility.

operation without unloading system log

Operating mode in which system log files are not unloaded. When a database is recovered, system logs are entered directly as input information to the database recovery utility, without unload logs being used.

optimizing based on cost

When multiple indexes have been created for a table, HiRDB selects and uses the index that has the least access cost and that it evaluates to be optimal based on the conditions specified for the table retrieval. This process in which HiRDB selects the index it evaluates to be optimal is called *optimizing based on cost*.

overload

Multiple stored functions can be defined with the same name as long as their parameter counts and data types are different. Stored functions that have the same name are said to be *mutually overloaded*.

override

Override is the process of overwriting the definition of a lower-order abstract data type with a routine that has the same name as the routine defined for the higher-order abstract data type.

page

The page is one of the data storage units; it is the smallest unit for database input/output operations. There are three types of pages:

- Data page: Page for storing rows of a table

- Index page: Page for storing index key values
- Directory page: Page for storing RDAREA status management information

pair logical volume group

A group of pair logical volumes. Pair logical volumes on which files have been distributed are handled in units of pair logical volume groups.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

pair logical volumes

Volumes paired between servers that have been logically named and configured.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

pair status

A designation that indicates the status of a pair volume. The pair statuses include PAIR, SMPL, and PSUS. For details about these statuses, see the RAID Manager documentation.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

pair volumes

With Lightning/Thunder series volumes, a volume at the main site and a volume at the remote site that correspond to each other.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

paired

A status in which a pair logical volume or pair logical volume group is paired.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

partitioning key index

An index that satisfies a particular condition becomes a partitioning key index. An index that does not satisfy the condition is called a non-partitioning key index. The condition is explained as follows.

The condition is whether the table involved is partitioned on the basis of single-column partitioning or multicolumn partitioning. When only one column is used in the table partitioning condition, the partitioning is said to be *single-column partitioning*; when multiple columns are used in the table partitioning condition, the partitioning is said to be *multicolumn partitioning*.

(a) Single-column partitioning

An index satisfying one of the following conditions is a partitioning key index:

Conditions

- Single-column index is defined for the column (partitioning key) on the basis of which storage conditions were specified when the table was row partitioned
- Multicolumn index is defined, and the column (partitioning key) on the basis of which storage conditions were specified when the table was row partitioned is the first constituent column

Figure D-1 shows an example (based on the `STOCK` table) of an index that becomes a partitioning key index.

Figure D-1: Partitioning key index: Single-column partitioning

STOCK				
PCODE	PNAME	COL	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29

↑ Column specified for the partitioning condition (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 `PCODE` column, which is a partitioning key, is used as an index, the index becomes a *partitioning key index*. If any other column is used as an index, the resulting index becomes a non-partitioning key index.
2. Specifying the `PCODE` column, which is the partitioning key, as the first constituent column of the index makes the resulting multicolumn index a *partitioning key index*.
3. Specifying the `PCODE` column, which is the partitioning key, as a column other than the first constituent column of the index makes the resulting multicolumn index a *non-partitioning key index*.

(b) Multicolumn partitioning

An index satisfying the following condition is a partitioning key index:

Condition

- Index is created on the basis of multiple columns, beginning with the partitioning key and containing all the columns specified for partitioning from the beginning and without any change in their order.

Figure D-2 shows an example (based on the STOCK table) of an index that becomes a partitioning key index.

Figure D-2: Partitioning key index: Multicolumn partitioning

STOCK				
PCODE	PNAME	COL	PRICE	SQUANTITY
101L	Blouse	Blue	35.00	62
101M	Blouse	White	35.00	85
201M	Polo shirt	White	36.40	29

Columns specified for the partitioning condition (partitioning key)

CREATE TABLE STOCK...
HASH HASH1 BY PCODE,PRICE...

Explanation

```
CREATE INDEX A12 ON STOCK (PCODE ASC, PRICE DESC)      1
CREATE INDEX A12 ON STOCK
    (PCODE ASC, PRICE DESC, SQUANTITY ASC)      2
CREATE INDEX A12 ON STOCK (PRICE DESC, PCODE ASC)      3
CREATE INDEX A12 ON STOCK
    (PCODE ASC, SQUANTITY DESC, PRICE DESC)      4
```

1. All partitioning keys (PCODE and PRICE columns) are specified, and these keys are specified in the same order as in the table definition. Therefore, this multicolumn index is a *partitioning key index*.
2. All partitioning keys (PCODE and PRICE columns) are specified, and these keys are specified in the same order as in the table definition. Therefore, this multicolumn index is a *partitioning key index*.
3. All partitioning keys (PCODE and PRICE columns) are specified, but these keys are specified in an order that differs from the table definition. Therefore, this multicolumn index is a *non-partitioning key index*.
4. All partitioning keys (PCODE and PRICE columns) are specified, but these keys are specified in an order that differs from the table definition. Therefore, this multicolumn index is a *non-partitioning key index*.

password-invalid account lock state

When limitations are set on password character strings by the connection security facility, a user violating any of the limitations is placed in password-invalid account

lock state. A user in this status can no longer establish connection with HiRDB (CONNECT).

plug-in

A plug-in is a HiRDB package product that provides *abstract data types* in which documents, images, and other multimedia entities are defined, and a facility for rapid manipulation of complex data.

plug-in index

A plug-in index is a type of index provided by a plug-in.

POSIX library version

A HiRDB execution environment required for the Directory Server linkage facility, Java stored procedures, Java stored functions, and the HiRDB External Data Access facility. To use the POSIX library version, you specify the `-l` option of the `pdsetup` command.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

post-source

The source program generated by preprocessing embedded SQL statements.

predefined data type

A data type provided by the HiRDB system is called a predefined data type. The predefined data types include `INTEGER`, `CHARACTER`, `DATE`, etc.

prefetch facility

A facility that reads multiple pages at a time from a global buffer or a local buffer. It can reduce the I/O time when searches on large volumes of data are performed in HiRDB file system areas that use the raw I/O facility. It is particularly effective for searches that do not use an index, or for ascending-order searches on large data sets in tables that do use an index.

preparations for system log application

A procedure that enables performance of log application in which database integrity is established and information in files required to perform log application is set in the proper state by temporarily synchronizing the databases at the transaction execution site and the log application site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

preprocessing

Processing performed on code before it is compiled by a language compiler.

Preprocessing converts SQL statements to high-level language code.

pre-update log acquisition mode

Pre-update log acquisition mode is one of the modes for collecting a database update log when a UAP or utility executes. This mode acquires only the database update log needed for rollback when a UAP or utility is updating the contents of an RDAREA.

primary key

The primary key enables you to uniquely identify a row in a table. Columns for which the primary key is defined are subject to the *uniqueness constraint* and the *NOT NULL constraint*. The uniqueness constraint is the limitation that does not permit duplicate data in a key (a column or a set of columns)— i.e., all data in the key must be unique. The NOT NULL constraint is the limitation that does not permit the null value in the key columns.

process

An instance of execution of a program. Each process is allocated a virtual space and CPU resources on a time-sharing basis. Executing more than one process in parallel (multi-processing) is one way to increase throughput.

protection mode

A protection mode can be used to select a processing method when synchronous copying to a remote site cannot be performed. A protection mode can be selected when the all synchronous method or the hybrid method is being used. The protection modes include `data` and `never`, and their processing methods are explained below:

- `data`: Stops update processing at the main site (update processing of volumes containing files that cannot be copied synchronously).
- `never`: Continues update processing at the main site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

RAID Manager instance

A suite of facilities that identify the scope of operations and management that can be performed by RAID Manager. Each instance is identified by an instance number.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

range specification recovery

One of the methods used by the database recovery utility for recovering a database. It is a technique by which the database can be recovered to a user-specified point between the time a backup was made and the time the failure occurred. The `-T` option of the database recovery utility is used to specify the recovery time.

rapid grouping facility

When the `GROUP BY` clause of an SQL is used for grouping, sorting is performed before grouping. This is the rapid grouping facility, and it combines grouping with hashing.

rapid system switchover facility

A facility whereby server processes and the system server are started on a standby HiRDB beforehand, so that startup processing of server processes and the system server is not performed when the system is switched over. The system switchover time is reduced by the amount of time required to perform startup processing of server processes and the server system. In addition to the rapid system switchover facility, user server hot standby is available for reducing system switchover time. The rapid system switchover facility can reduce the system switchover time more than user server hot standby can (the rapid system switchover facility includes the user server hot standby functionality).

RD node

In the distributed database facility, a DBMS on a server connected on a distributed network. In the OSI-RDA protocol, it is called a *resource*.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

RD node name

In the distributed database facility, the node name used to specify a target server DBMS connected on a distributed network. In the OSI-RDA protocol, it is called a *resource name*.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

RDAREA

The RDAREA is the basic unit for storing data; an RDAREA may store 1-16 HiRDB files. Following are the types of RDAREAs:

- Master directory RDAREA
- Data dictionary RDAREA
- Data directory RDAREA
- Data dictionary LOB RDAREA
- User RDAREA
- User LOB RDAREA
- Registry RDAREA

- Registry LOB RDAREA
- List RDAREA

RDAREA automatic extension

When RDAREA automatic extension is being used and a space shortage occurs in an RDAREA, segments are added automatically to expand the capacity of the RDAREA, provided that there is adequate unused free space in the HiRDB file system area.

RDAREAs, shutting down

To limit accesses by UAPs and utilities to an RDAREA, the RDAREA can be shut down (*command shutdown*). If an error (such as an I/O error) occurs on an RDAREA, HiRDB may shut down the RDAREA automatically (*error shutdown*).

There are various shutdown modes, such as one that allows referencing and updating; the user selects the appropriate shutdown mode.

For example, when `pdload` is to be used to load data or `pdrorg` is to be used to re-organize a table, the applicable RDAREAs should first be shut down because such processing usually takes a considerable amount of time.

Real Time SAN Replication

A facility that enables jobs to continue being processed on a standby system set up at a remote site in the event of a disaster (earthquake, fire, etc.) that makes it physically impossible to quickly restore the system that is normally used.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

recovery-unnecessary front-end server

If an error occurs on the unit where the front-end server is located, transactions that were executing from that front-end server may be placed in uncomplete status for transaction determination. Because a transaction in this status has locked the database, some database referencing or updating becomes restricted. Normally, to complete a transaction that is in uncomplete status for transaction determination, the front-end server must be recovered from the error and restarted. However, if the abnormally terminated front-end server is a recovery-unnecessary front-end server, HiRDB automatically completes any transaction that was placed in uncomplete status for transaction determination. This enables other front-end and back-end servers to be used to restart database update processing. A unit containing a recovery-unnecessary front-end server is called a recovery-unnecessary front-end server unit.

reduced activation

Reduced activation is the process of starting HiRDB using only the normal units, when there are units that cannot be started. Normally, a HiRDB/Parallel Server cannot be started if any of its units cannot be started. In such a case, the reduced activation

facility enables the HiRDB/Parallel Server to be started using only the remaining units even if some units cannot be activated due to errors.

reference buffer

When data is referenced, it is referenced in a global buffer. A buffer that is used for referencing data, or a buffer for a database that has been updated, is called a *reference buffer*.

reference-only back-end server

A back-end server in which shared tables and shared indexes of shared RDAREAs can be referenced but not updated.

referential constraint

A constraint that is defined for a specific column (foreign key) during table definition in order to maintain referential conformity in data between tables. A table for which referential constraints and a foreign key are defined is called a referencing table, and a table that is referenced from the referencing table using the foreign key is called a referenced table. For the referenced table, the primary key that is referenced by the foreign key must be defined.

registry facility

The registry facility enables HiRDB to maintain information unique to plug-ins; such information is used by the plug-ins for data manipulation.

registry LOB RDAREA

The registry LOB RDAREA stores a table for managing registry information (registry table). This RDAREA is required in order to use the registry facility. Not all plug-ins use the registry facility. The system determines automatically whether data is to be stored in the registry LOB RDAREA or the registry RDAREA on the basis of the length of the data. Stored procedures for data manipulation, such as ones that register information in the registry management table, are also stored in this RDAREA.

registry RDAREA

The registry RDAREA stores a table for managing registry information (registry table). This RDAREA is required in order to use the registry facility. Not all plug-ins use the registry facility.

regular expression

A method of representation that allows different character strings, such as any character strings, repetitions of the same character string, or any number of characters, to be expressed in a specific sequence (pattern) of characters forming part of a larger character string. Regular expressions are specified in a pattern character string of the `SIMILAR` predicate.

regular unit

The unit where a host BES is located when the standby-less system switchover (effects distributed) facility is used.

remote site

Site where the backup system is located for Real Time SAN Replication.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

reorganization by schema

Reorganization by schema is the process of reorganizing in a batch all the tables in a schema. To reorganize in a batch all the tables you own, you should execute the reorganization by schema by specifying the authorization identifier of the schema to be reorganized in the `-t` option of the database reorganization utility. The specification format is `-t authorization-identifier.all`.

In addition to reorganization by schema, the other units of reorganization are as follows:

- Reorganization by table
- Reorganization by RDAREA

reorganization with synchronization points set

Normally, while a table is being reorganized, transactions cannot be reconciled until the storage processing of all the data has been completed. This means that synchronization point dumps cannot be obtained during execution of the database reorganization utility. If HiRDB terminates abnormally during reorganization of a large quantity of data, it will take a long time to restart HiRDB. To resolve this problem, you can set synchronization points for any number of data items during storage of the data (reload processing) in order to reconcile transactions. This is called reorganization with synchronization points set.

To perform reorganization with synchronization points set, you must specify a synchronization point lines count, which is the number data items to be stored before a synchronization point is set. This value is specified in the `option` statement of the database reorganization utility. When a table containing a large quantity of data is to be reorganized, you should consider whether or not to execute reorganization with synchronization points set.

Synchronization point setting can also be specified in the database load utility; this is called data loading with synchronization points set.

repetition column

A column that consists of multiple elements in a single row is called a repetition column. An element is one of the multiple items in the repetition column row. A

repetition column is defined by `CREATE TABLE` with the maximum number of elements specified. However, the number of elements can be increased later with `ALTER TABLE`. Figure D-3 shows an example of a table that contains repetition columns.

Figure D-3: Example of a 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.

replication facility

The replication facility enhances the efficiency of data utilization in a data warehouse by linking a database in a Hitachi mainframe with a HiRDB database or a HiRDB database with another HiRDB database.

reserved word

A character string that has been registered as a keyword used in SQL statements.

Reserved words cannot be used as table names or column names. Otherwise, a reserved word can be used as a name if it is enclosed in double quotation marks ("").

A keyword that has been registered as a reserved word can be deleted by the SQL reserved word deletion facility; however, some functions may become disabled if a reserved word is deleted.

rollback

Rollback is the process of invalidating the database processing performed by a transaction when an error has occurred in the transaction.

row partitioning

Row partitioning is the process of storing a table, index, or LOB column by dividing it into multiple user RDAREAs or user LOB RDAREAs. When a table is row-partitioned, its indexes can also be row partitioned in correspondence with the row-partitioned table. When a table contains a LOB column, it can be partitioned and stored in multiple user LOB RDAREAs in correspondence with the row-partitioned table. When a table is to be row-partitioned, it is necessary to specify storage conditions for the row partitioning; this is done with the `CREATE TABLE` statement of the definition system SQL. When an index is row-partitioned, it is necessary to specify the user RDAREAs in which the row-partitioned index is to be stored; this is done with the `CREATE INDEX` statement of the definition system SQL.

schema

A concept that encompasses tables, indexes, abstract data types (user-defined types), index types, stored procedures, stored functions, triggers, and access privileges.

schema definition privilege

The privilege required to define a schema.

security facility

The security facility prevents anyone who does not have the required authorization from accessing the database.

segment

The segment is a unit of data storage consisting of multiple contiguous pages. It is the unit of allocation for storage of tables and indexes. A segment can store either one table or one index.

server machine

A workstation or personal computer that runs HiRDB server software.

server mode

The system switchover facility can be operated in either monitor mode or server mode. In monitor mode, only system failures are monitored. In server mode, both system failures and server failures (such as HiRDB abnormal termination) are monitored. System switchover can take less time in server mode than in monitor mode because a standby HiRDB is started beforehand in server mode. In addition, the system switchover facility must be operating in server mode to use the following functions:

- user server hot standby
- rapid system switchover facility
- standby-less system switchover facility

shared index

An index that is stored in a shared RDAREA and can be referenced from all back-end servers. In order to achieve SQL and UAP compatibility with HiRDB/Parallel Servers, shared indexes can also be defined in HiRDB/Single Servers. If defined, however, such shared indexes are stored in user RDAREAs because shared RDAREAs cannot be defined in a HiRDB/Single Server.

shared RDAREA

A user RDAREA that can be referenced from all back-end servers. Only a HiRDB/Parallel Server can define shared RDAREAs.

shared table

A table that is stored in a shared RDAREA and can be referenced from all back-end servers. In order to achieve SQL and UAP compatibility with HiRDB/Parallel Servers, shared tables can also be defined in HiRDB/Single Servers. If defined, however, such shared tables are stored in user RDAREAs because shared RDAREAs cannot be defined in a HiRDB/Single Server.

site status

The status of a site as recognized by the log-only synchronous method. There are four statuses: initial, ready, transaction, and log application.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

skipped effective synchronization point dump monitoring facility

With the `pd_spd_syncpoint_skip_limit` system common definition operand, you can specify the maximum number of times synchronization point dumps can be skipped during a transaction.

If, for example, an infinite loop occurs in a UAP, synchronization point dump processing may not be performed several times in succession (processing may be skipped a number of times). If the number of times this processing is skipped exceeds a value specified by the user, the affected transaction is forcibly suspended and rolled back. This facility keeps track of the number of times in succession that synchronization dumps are skipped.

space conversion facility

The space conversion facility unifies coding of single-byte and double-byte spaces that are intermixed in table data. Double-byte spaces are coded as `X'8140'` in JIS Kanji code and as `X'A1A1'` in EUC Chinese Kanji code. Two single-byte spaces are coded as `X'2020'`.

SQL connection

The logical connection of a UAP to an RD node for the purpose of executing an SQL

statement.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

SQL extension optimizing options

SQL extension optimizing options enable optimization of SQL at execution time by determining the most efficient access paths that can be specified, taking into consideration the status of the database.

The following SQL extension optimizing options are provided:

1. Application of optimizing mode 2 based on cost
2. Hash-execution of a hash join or a subquery
3. Suppression of foreign server execution of SQL statements that include a join
4. Forced foreign server execution of SQL statements that include a direct product
5. Suppression of unconditionally generated derived rapid search conditions that can be executed on foreign servers

SQL object

An SQL object is the object that is compiled by HiRDB when SQL statements are defined and executed.

SQL optimization options

SQL optimization options enable optimization of SQL at execution time by determining the most efficient access paths that can be specified, taking into consideration the status of the database.

The following SQL optimization options are provided:

1. Forced nest-loop-join
2. Creating multiple SQL objects
3. Increasing the target floatable servers (back-end servers for fetching data)
4. Prioritized nest-loop-join
5. Increasing the number of floatable server candidates
6. Priority of OR multiple index use
7. Group processing, ORDER BY processing, and DISTINCT set function processing at the local back-end server
8. Suppressing use of AND multiple indexes
9. Rapid grouping facility

10. Limiting the target floatable servers (back-end servers for fetching data)
11. Separating data collecting servers
12. Suppressing index use (forced table scan)
13. Forcing use of multiple indexes
14. Suppressing creation of update-SQL work tables
15. Derivation of rapid search conditions
16. Application of scalar-operation-included key conditions
17. Batch acquisition from plug-in-provided functions

SQL optimization specification

Optimizations to enhance SQL search efficiency can be specified for SQL statements. The following three SQL optimization specifications are available:

- SQL optimization specification for index utilization
- SQL optimization specification for the join method
- SQL optimization specification for the subquery execution method

SQL optimization specifications take precedence over any specifications of SQL optimization options and SQL extension optimizing options.

SQL preprocessor

A program that converts SQL statements into high-level language code so that they can be compiled by a high-level language compiler.

SQL runtime warning output facility

After SQL code is executed, HiRDB checks its runtime. If this check determines that the SQL code's runtime is greater than the time specified (set as a ratio of the `PDCWAITTIME` value), this facility is used to output the following warning information for that SQL code:

- SQL runtime warning information file
- Warning message (KFPA20009-W)

SQL stored function

An SQL stored function is a stored function in which procedures are coded in SQL.

SQL stored procedure

An SQL stored procedure is a stored procedure in which procedures are coded in SQL.

standby system switchover facility

A standby HiRDB separate from the HiRDB that is actively processing jobs is

deployed, and if a failure occurs on the server machine or on HiRDB, job processing can be automatically switched over to the standby HiRDB. This is called the standby system switchover facility.

standby-less system switchover (effects distributed) facility

A type of standby-less system switchover facility. In the event of an error, this facility distributes to multiple active units the processing requests for back-end servers (BESs) in the erroneous unit; this is called the standby-less system switchover (effects distributed) facility.

standby-less system switchover facility

Unlike a standby system HiRDB that has been prepared for the standby system switchover facility, with the standby-less system switchover facility a standby HiRDB need not be prepared. If an error occurs, there is no system switchover to a standby HiRDB system, but another operating unit takes over the processing. This is called the standby-less system switchover facility.

The standby-less system switchover facility can be further subdivided as follows:

- Standby-less system switchover (1:1) facility
- Standby-less system switchover (effects distributed) facility

standby-less system switchover (1:1) facility

Unlike the standby system switchover facility that provides a standby HiRDB system, the standby-less system switchover facility does not require that a standby HiRDB system be kept in reserve. In the event of an error, the standby-less system switchover facility transfers processing to another active unit without switching over to a standby HiRDB system. This is called the standby-less system switchover facility.

The standby-less system switchover (1:1) facility can perform 1-to-1 unit switchover in the event of an error and assign processing to another, designated back-end server.

static SQL

A method of coding SQL within a program when a UAP is created. Contrast with *dynamic SQL*, which is a method of generating SQL code while a UAP is executing.

statistics log file

A file that stores statistical information (statistics logs) output by HiRDB.

status file

A file that stores system status information that may be needed to restart HiRDB is called a status file. There are two types of status files:

- Server status files
- Unit status files

status file for log application processing

A status file required by the log-only synchronous method. It is used to obtain system status information when log application processing is performed at the log application site. Its counterpart, the status file for transaction processing, is required at the transaction execution site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

status file for transaction processing

A status file required by the log-only synchronous method. It is used when log application processing is performed at the log application site. Its counterpart, the status file for log application processing, is required at the transaction execution site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

stored function

A facility for registering as a function in the database a data process coded in SQL or Java. Because input parameters can be assigned to a stored function and because a stored function can generate return values, a stored function can be called by specifying it as a value expression in an SQL statement. A stored function can be defined in a function in the `CREATE FUNCTION` or `CREATE TYPE` statement. Data processes coded in SQL or Java are compiled during definition, an SQL object coding of the procedure is generated, and the object, together with the definition information, is stored in the database.

stored procedure

A stored procedure enables a sequence of database access procedures coded in SQL or Java to be stored as a procedure in a database. Output or input/output parameters can be assigned to a stored procedure, and a stored procedure can be called by the `CALL` statement in SQL. A stored procedure can be defined in a procedure in the `CREATE PROCEDURE` or `CREATE TYPE` statement. Database operations coded in SQL or Java are compiled at definition time, an SQL object coding the access procedures is generated, and the object, together with definition information, is stored in the database.

substitutability

Substitutability is the capability to substitute a value in a lower-order abstract data type for the value in the higher-order abstract data type.

subtype

A subtype is an abstract data type that is created by customizing an abstract data type.

Sun Java System Directory Server linkage facility

By using the Sun Java System Directory Server, you can manage and authenticate HiRDB users. This capability is provided by a process called the Sun Java System Directory Server linkage facility. By using this facility, you can centrally manage organizational and user information (user IDs, passwords, affiliations, job titles) that is otherwise managed separately in HiRDB, Groupmax, and other systems.

supertype

An abstract data type that is higher in order than a specialized abstract data type (subtype) is called the supertype.

synchronization point

The point at which a transaction is completed is called a synchronization point. Synchronization point processing that validates a transaction-induced update is called a commit, and synchronization point processing that invalidates a transaction is called a rollback.

synchronization point dump file

In the event of abnormal termination of HiRDB, recovering it using system log information only will require system log information covering the entire time since HiRDB was startup, which will require a significant amount of time for recovery processing. Therefore, points can be established at fixed intervals during HiRDB operation, and required HiRDB management information can be saved at those points. This means that all system log information produced prior to the point to be used for recovery is no longer needed, thus reducing the recovery time. The file that stores the HiRDB management information obtained at each such point is called a synchronization point dump file.

synchronization point dump file for log application processing

A synchronization point dump file that is required by the log-only synchronous method. It is used to obtain synchronization points when log application processing is performed at the log application site. Its counterpart, the synchronization point dump file for transaction processing, is required at the transaction execution site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

synchronization point dump file for transaction processing

A synchronization point dump file that is required by the log-only synchronous method. It is used when log application processing is performed at the log application site. Its counterpart, the synchronization point dump file for log application processing, is required at the transaction execution site.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

synchronous copy

One of the processing methods used to update-copy data to a remote site. After update-copy processing at the remote site is completed, update-copy processing at the main site is completed (update-copy processing at the main site waits for completion of the update-copy processing at the remote site).

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

synchronous group

A group made up entirely of synchronous pair volumes.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

synchronous pair volume

A pair volume that has been created by specifying either `data` or `never` as the fence level. When data is written to the P-VOL, it is mirrored synchronously onto the S-VOL. When paired, no differences exist between the data on the P-VOL and the data on the S-VOL.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

system file

A system file stores information that will be needed if it becomes necessary to recover the system in the event of an error. *System file* is a generic term that includes the following files:

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

system generator

The system generator is a facility that enables a HiRDB system to be constructed interactively by selecting displayed values.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

system log file

A system log file stores a history of database updates; it is also called a journal file. Historical information on database updates is called the system log (or system log information), and it is used by HiRDB to recover the database in the event of abnormal termination of either HiRDB or a UAP. The system log is also used as input

information for recovery of the database.

system RDAREA

System RDAREA is a generic term for the following types of RDAREAs:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREA

system switchover facility

When a standby server machine is provided separately from the server machine that is processing jobs, job processing can be switched automatically by HiRDB to the standby server machine in the event the running server machine fails. This is called the system switchover facility.

table reorganization

Over time, additions and updates of data in a table tend to fragment the arrangement of rows, giving rise to unusable free space. Table reorganization reorganizes table data in a user RDAREA or LOB data in a LOB RDAREA to eliminate unusable free space. The database reorganization utility (`pdrorg` command) is used to reorganize a table.

tape device access facility

A facility that enables access to files on DAT, DLT, and LTO. The tape device access facility is used for the following types of files:

- Input data files (input data files specified in the `source` statement of the `pdload` command)
- Unload data files (unload data files specified in the `unload` statement of the `pdrorg` command)
- Unload data files (LOB data unload data files specified in the `lobunld` statement of the `pdrorg` command)
- Backup files (backup files specified with the `-b` option of the `pdcopy` or `pdrstr` command)

This term has no practical application for UNIX users, because it is related to a facility that cannot be used with a UNIX version of HiRDB.

transaction

A transaction is a unit of logical work; for example, a series of database operations. The transaction is also the basic unit for recovery and locking.

transaction execution site

In the log-only synchronous method, the site that accepts transactions.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

trigger

By defining a trigger, you can have SQL statements automatically execute when an operation (updating, insertion, deletion) is performed on a particular table. When a particular table is updated, a trigger based on the associated event allows you to automatically perform operations such as updating another table as well.

unbalanced index split

Unlike normal page split, the unbalanced index split method splits the data in an index page into two unequal parts, rather than into two equal parts. Index storage efficiency is improved by using this method when an ascending or descending middle key is to be added.

uniqueness constraint

The uniqueness constraint is a limitation that prohibits a data value from appearing more than once in a column (all data values in the column must be unique).

unit

A unit refers to the HiRDB operation environment within a single server machine.

unit controller

A unit controller is a system that controls and monitors server execution in a unit and that controls communication between units.

unload log file

A file created by unloading a system log file (system log information) with the `pdlogunld` command is called an unload log file.

unload statistics log file

A file created by unloading the contents of a statistics log file.

unused page

A page that is not being used.

unused segment

A segment that is not being used. Such a segment can be used by any table (or index) in the RDAREA.

updatable back-end server

A back-end server on which shared tables and shared indexes in shared RDAREAs can be updated.

updatable online reorganization

Refers to functionality that allows databases to be accessed and updated during database reorganization. Processes that access or update a database perform the operation on a replica database. To perform updatable online reorganization, you must install HiRDB Staticizer Option, and you must also specify related operands in the HiRDB system definitions.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

update buffer

When data is updated, the data is first updated into a global buffer, before being updated in the database. The buffer that stores this data before the database is updated is called an *update buffer*.

update copy

An operation in which, when updating of data on a primary volume occurs, the updated data is also mirrored (copied) onto a secondary volume. Update copying maintains integrity between the primary volume and the secondary volume.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

used page

A page in which a table or an index is stored. A used page that is completely filled with data such that no more can be added is called a full page, and a used page from which data has been deleted so that it no longer contains data is called a used free page.

used segment

A segment in which table or index data is stored. A used segment that is completely filled with data such that no more can be added is called a full segment, and a used segment from which data has been deleted so that only free pages remain (used free pages or unused pages) is called a used free segment.

user-defined type

A data type that can be defined by the user is called a user-defined type. Abstract data types are user-defined types.

user LOB RDAREA

A user LOB RDAREA stores large variable object data, such as documents, images, and sounds. The following types of data must be stored in a user LOB RDAREA:

- Column with the BLOB type specification (BLOB column)
- Attribute with BLOB specification in an abstract data type

- Plug-in index

user mapping

Information defining the user ID and password required to log into a foreign server. This information is defined for each foreign server and is used to establish connection.

user RDAREA

A user RDAREA stores tables and indexes created by a user.

user server hot standby

A facility that starts server processes on a standby HiRDB beforehand, so that startup processing of server processes is not performed when the system is switched over. The system switchover time is reduced by the amount of time required to perform startup processing of server processes.

In addition to user server hot standby, the rapid system switchover facility is available for reducing system switchover time. The rapid system switchover facility can reduce the system switchover time more than user server hot standby can (the rapid system switchover facility includes the user server hot standby functionality).

utility special unit

A utility special unit is a server machine in which only input/output devices to be used for executing a utility are set up. A utility special unit can be set up only in a HiRDB/Parallel Server. The following utilities can use a utility special unit:

- Database load utility
- Database reorganization utility
- Dictionary import/export utility
- Database copy utility
- Database recovery utility

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

value (instance)

The term value refers to a specific value of an abstract data type.

view table

A newly defined virtual table in which specific rows and columns are selected from a table that actually exists (called a base table).

If you are using HiRDB External Data Access, you can create a view table from foreign tables.

volume attribute

There are three types of volumes, primary volumes (P-VOL), secondary volumes (S-VOL), and simplex volumes (SMPL). The volume attribute refers to the attribute indicating one of these three volume types.

This term has no practical application for Windows users, because it is related to a facility that cannot be used with a Windows version of HiRDB.

work table file

A file that stores temporary information needed in order to execute an SQL statement is called a work table file.

Index

Symbols

-M option (pdcopy command) 272

Numerics

24-hour-per-day continuous operation 10, 225

A

abstract data type 118, 387
 as data type 120
 defining 118
 null value of 124
abstract data type column
 deleting 161
 retrieving 161
 structure base table 387
 updating 161
accepting unit 314, 387
access privilege 332, 387
 granting of (role) 40
access processing method 233
account lock period 347, 387
ADO.NET-compatible application 21
ADT 387
alias IP address 387
all asynchronous method 387
all synchronous method 388
ALTER TRIGGER 175
alternate BES 312, 388
alternate BES unit 312
alternate portion 313
alternating 313
API, compatible with X/Open 259
application
 ADO.NET-compatible 21
 JDBC-compatible 20
 ODBC-compatible 19
 OLE DB-compatible 19
ASSIGN LIST statement 64, 197

asynchronous copy 388
asynchronous group 388
asynchronous pair volume 388
asynchronous READ facility 237, 388
asynchronous XA call 43
audit event 340
audit privilege 331, 334, 389
audit trail 334, 389
 file 339
 table 339, 389
auditor 334, 389
auditor security event 341
automatic extension 299
automatic log unloading facility 276, 389
automatic reconnect facility 259, 389
automatic startup 229
automatic system switchover 318

B

B-tree index 101
B-tree structure 101
back-end server 389
 for connecting to foreign servers 28, 389
 for viewing or updating foreign tables 28
 HiRDB/Parallel Server 13
backup
 acquisition mode 272, 390
 made in unit 272
 making 272
 operation without collecting database update log 268
 reducing time needed to make (user LOB RDAREA) 280
backup file 270, 390
 storage of 273
backup-hold 279, 390
 RDAREA placed on 279
base row 391
base table 98

basic attributes (linkage with JP1) 50
 batch file 9
 BINARY 70
 BINARY data
 addition update of 249
 partial extraction of 249
 binary data 70
 binary data string 70
 BLOB 70, 391
 BLOB and BINARY data addition update and partial
 extraction facility 248, 391
 BLOB data
 addition update of 249
 file output facility of 244, 391
 partial extraction of 249
 block transfer facility 186, 391
 BOOLEAN 70
 boolean data 70
 boolean value 70
 boundary value specification 78
 branch row 391

C

CALL statement 167
 candidate key 73
 CHARACTER 70
 character code set 392
 character data 70
 character string
 fixed-size 70
 variable-size 70
 check constraint 182, 392
 check pending status 184, 392
 Class file 173
 CLOSE statement 152
 cluster key 73, 393
 CLUSTER KEY option (CREATE TABLE) 74
 cluster system 310
 column 68
 commit 256, 271
 commit processing 258
 commitment control 256
 communications overhead 186
 compiling 393

concurrent connections, maximum number of 393
 CONNECT privilege 332
 CONNECT statement 254
 connection frame guarantee facility for client
 groups 393
 connection security facility 345, 393
 consecutive certification failure account lock
 state 347, 393
 consecutive certification failures, restrictions on
 number of 347
 consistency group 393
 constructor function 122, 394
 user-defined 123
 Cosminexus, linkage to 58
 CREATE CONNECTION SECURITY 346
 CREATE INDEX 103
 create rdarea statement 299
 specifying 139
 CREATE SCHEMA 67
 CREATE TABLE 69
 CREATE TRIGGER 175
 CREATE TYPE 71
 CREATE VIEW 98
 current file 274
 cursor 152, 394

D

DABroker 57
 data
 basic manipulation of 152
 deleting 154
 deleting (abstract data type) 160
 inserting 154
 inserting (abstract data type) 160, 162
 manipulation of, in table containing abstract
 data type 159
 operation of 157
 processing 158
 retrieving 152
 retrieving (abstract data type) 159
 searching for specific 155
 updating 153
 updating (abstract data type) 159
 data definition language 394

- data dictionary 394
 - LOB RDAREA 63, 394
 - RDAREA 63
- data dictionary table 379
- data directory RDAREA 63, 394
- data global buffer 233
- data linkage facility 31
- data linkage product, linkage to 31
- data loading 394
 - with synchronization points set 288, 395
- data local buffer 242
- data manipulation language 395
- data page 139
- data type 69
 - predefined data type 69
 - user-defined data type 69
- data warehouse 395
- database
 - access mode 17
 - access tool 57
 - committing 280
 - inheriting 405
 - logical structure of 61
 - management system 2
 - managing 269
 - physical structure of 137
 - preparation for error of 272
 - recovery of 270
- database access
 - improving performance of 186
 - using SQL 149
- database copy utility 272
- database extraction/reflection service facility 31
- database management table 291
- database mapping facility 35
- database recovery utility 270
- database reorganization utility 285
- database state analyzed table 291
- database update log
 - acquiring 265
 - operation without collecting 265
- DATAFRONT 53
- DATE 70
- date 70
 - date data 70
 - date interval data 70
- DB access product 200
- DBA privilege 330
- DBMS 2
- DCE threading 367
- deadlock 264, 395
 - example of (when index key value no-lock option is not used) 115
 - example of preventing 114
- decimal signed normalized number, facility for conversion to 306, 398
- DECLARE CURSOR 152, 191
- default constructor function 123, 395
- default RD node 395
- deferred write processing 238, 396
- deferred write trigger 238
- definition source 63
- delayed rerun 230, 396
- delayed rerun operation 8
- DELETE facility using arrays 190
- DELETE privilege 333
- DELETE statement 154
- deletion-prevented duration 94
- dictionary server 396
 - HiRDB/Parallel Server 13
- dictionary table 63
- differential backup 279
- differential backup facility 277, 396
- differential backup group 279
- differential backup management file 396
- direct disk access (raw I/O) 207
- directory page 139
- Directory Server linkage facility 331, 396
- directory server product, linkage to 37
- directory service 37
- DISCONNECT statement 254
- DISTINCT 158
- distributed client 397
- distributed client facility 397
- distributed database facility 397
- distributed nest-loop-join 397
- distributed RD node 397
- distributed server 397

Index

distributed server facility 398
distributed transaction processing 42
DocumentBroker 56
double-precision floating point 70
DROP TRIGGER 175
DTP 42
duplexed synchronization point dump file 211
duplexed system log file 209
dynamic registration 46
dynamic SQL 398
dynamic transaction registration 43

E

element 96
embedded SQL UAP 151
embedded UAP 398
EMPTY option (CREATE INDEX) 113
encapsulation 121, 132, 398
 level 132
error shutdown 267
event monitoring, using JP1/Integrated
Management 51
events, reporting (linkage with JP1) 50
EX, lock mode 262
EXCEPT 158
EXCEPT VALUES option (CREATE INDEX) 112
exception key value 112, 398
EXISTS predicate 157
expand rdarea statement 299
extended attributes (linkage with JP1) 50
extent 300

F

facility
 for monitoring abnormal process
 terminations 398
 for parallel writes in deferred write
 processing 238, 399
 for predicting reorganization time 288, 399
 using arrays 399
falsification prevention facility 399
falsification-prevented table 93
fence level 399
FETCH facility using array 188

FETCH statement 152
file group
 synchronization point dump file 211
 system log file 209
first dimension partitioning column 85
FIX attribute 73, 400
FIX attribute table 400
FIX hash partitioning 80, 400
FIX option (CREATE TABLE) 73
fixed point 70
flexible hash partitioning 80, 400
FLOAT 70
floatable server 14, 400
foreign HiRDB 28
foreign index 400
FOREIGN KEY 179
foreign key 178, 400
foreign server 27, 401
foreign table 27, 401
free page 401
free page release utility 294
free page reuse mode 142, 401
free segment 401
free space reuse facility 142, 401
front-end server 401
 HiRDB/Parallel Server 13
frozen update command 280
frozen update status 281
full backup 279
function 163
function call 167
functional differences between HiRDB versions on
different platforms 374

G

global buffer 233, 401
 allocation procedure for 236
 dynamic update of 236, 398
global buffer pre-writing 240, 402
global buffer residence utility 241
global deadlock 402
GROUP BY clause 158

- group processing, ORDER BY processing, and DISTINCT set function processing at local back-end server 193
 - grouped system switchover 318
 - guaranteed valid generation, number of 211
 - guest area 314, 402
 - guest BES 314, 402
- H**
- HA group 314
 - HA monitor 402
 - hash facility for hash row partitioning 83
 - hash group 84
 - hash partitioning 80
 - heterogeneous system configuration 14, 402
 - HiRDB
 - 64-bit-mode 361
 - administrator 403
 - advantages of using 6
 - architecture of 203
 - characteristics of 2
 - environment setup for 204
 - file 138, 403
 - functions not available in 64-bit-mode 363
 - migrating from 32-bit mode to 64-bit mode 368
 - option program product of 24
 - HiRDB Adapter for XML 35
 - HiRDB Advanced High Availability 24, 225
 - HiRDB Advanced Partitioning Option 25, 85, 91
 - HiRDB client 4, 403
 - connectivity to HiRDB server 383
 - HiRDB/Developer's Kit 4
 - HiRDB/Run Time 5
 - support range of 64-bit-mode 366
 - HiRDB Dataextractor 31
 - HiRDB Datareplicator 31
 - HiRDB External Data Access 26, 28
 - HiRDB External Data Access Adapter 29, 403
 - HiRDB External Data Access facility 403
 - overview of 26
 - HiRDB file 206
 - HiRDB File Link 56, 350, 356
 - HiRDB file system area 138, 206, 403
 - creation unit of 207
 - maximum size of 208
 - HiRDB Image Search Plug-in 56, 350, 356
 - HiRDB LDAP Option 25, 41
 - HiRDB server 3
 - connectivity to HiRDB client 383
 - HiRDB/Parallel Server 4
 - HiRDB/Single Server 4
 - HiRDB Spatial Search Plug-in 56, 350, 356
 - HiRDB SQL Executer 48, 151
 - HiRDB SQL Tuning Advisor 49
 - HiRDB system 2
 - configurations of 11
 - connection to 254
 - disconnection from 254
 - overview of 2
 - HiRDB system definition 220
 - after creation of 225
 - back-end server definition 222
 - dictionary server definition 221
 - foreign server information definition 222
 - front-end server definition 221
 - hub optimization information definition 222
 - server common definition 220, 221
 - single server definition 220
 - SQL reserved word definition 220, 222
 - system common definition 220, 221
 - UAP environment definition 220, 222
 - unit control information definition 220, 221
 - HiRDB system definition file 404
 - creating 225
 - HiRDB Text Search Plug-in 55, 350, 354
 - HiRDB XA library 42
 - functions provided by 43
 - HiRDB.ini file 403
 - HiRDB.NET data provider 21, 404
 - HiRDB/Developer's Kit 4
 - HiRDB/Parallel Server 4
 - back-end server 13
 - configuration of 11
 - consisting of unit 13
 - front-end server 13
 - system manager 13
 - unit 13

Index

HiRDB/Run Time 5
HiRDB/Single Server 4
 configuration of 11
 consisting of unit 11
holdable cursor 191
host BES 314, 404
Hub optimization information definition 404
hybrid method 404

I

image 56
image data 118
Image Database Access 56
IN SHARE MODE (LOCK TABLE statement) 264
incorporation during commit 239, 404
index 101, 404
 basic structure of 101
 example of row partitioning of (HiRDB/
 Parallel Server) 105
 example of row partitioning of (HiRDB/Single
 Server) 104
 for dictionary table 63
 guidelines for row-partitioning of (HiRDB/
 Parallel Server) 105
 guidelines for row-partitioning of (HiRDB/
 Single Server) 104
 local buffer 242
 reorganization of 291, 405
 row partitioning of 103
index global buffer 233
index key value no-lock 113
index page 139
index page split 109, 404
inheritance 128, 405
inner replica facility 405
INSERT facility using arrays 188
insert history maintenance column 94
INSERT ONLY option 94
INSERT privilege 333
INSERT statement 154
instance 435
INTEGER 70
integer 70
integrity check utility 184

integrity constraint 176, 405
inter-process memory communication facility 406
interface area 406
INTERVAL HOUR TO SECOND 70
INTERVAL YEAR TO DAY 70
IP address 406
iPlanet Console 41

J

JAR file 173
Java stored function 169, 406
Java stored procedure 169, 406
Java stored routine 169
 characteristics of 169
 creation and execution procedure 171
Java virtual machine 173
 position of 169
JDBC driver 20, 406
JDBC-compatible application 20
join 406
journal 209
JP1 406
JP1/Automatic Job Management System 2 51
JP1/Base 50
JP1/Integrated Management 50
JP1/Performance Management - Agent Option for
HiRDB 49
JP1/VERITAS NetBackup Agent for HiRDB
License 282

K

kernel threading 367
key range partitioning 76, 407
 with boundary values specified 78
 with storage condition specified 77
key values 101

L

LAN adapter 407
language compiler 407
LARGE DECIMAL 70
large file 407
large object data 70

LDAP 37
 leaf page 101
 linkage 407
 linker 407
 list 197, 407
 example of search, using 197
 list RDAREA 64, 407
 creating 199
 load module 408
 LOB column structure base table 408
 LOB data 408
 LOB global buffer 234
 LOB RDAREA 63
 local buffer 242, 408
 locator facility 250, 408
 lock 408
 lock mode 262
 exclusive mode 262
 protected update mode 262
 shared mode 262
 shared retrieval mode 262
 shared update mode 262
 LOCK PAGE (CREATE TABLE) 262
 lock period 264
 LOCK statement 264
 LOCK TABLE statement 264
 locked resource 261
 locking 261, 408
 automatic 263
 by page 262
 by row 262
 units of 261
 log acquisition mode 265, 408
 log application 409
 log application not possible status 409
 log application possible status 409
 log application site 409
 logical file 212
 logless shutdown 267
 LRU management method 239
 for global buffer 239
 LVM 409

M

main site 410
 manual startup 229
 master directory RDAREA 63, 410
 matrix partitioning 85, 410
 matrix-partitioned table 85
 MCHAR 70
 message queue monitoring facility 410
 Microsoft Cluster Server 310
 middle page 101
 minimum password length 346
 mixed character data 70
 mixed character string
 fixed-size 70
 variable-size 70
 module trace 410
 monitor mode 316, 410
 monitoring free area for system log file 411
 move rdarea statement 301
 MSCS 310
 multi-connection facility 43, 254, 411
 multi-front-end server 13
 multi-HiRDB 15, 411
 configuration of 15
 multi-stage system switchover 314
 multicolumn index 102
 multicolumn partitioning 107
 multiple front-end server 411
 mutual alternating configuration 321
 mutual system switchover configuration 320
 MVARCHAR 70

N

n-gram index method 55
 n-gram index plug-in 352
 narrowed search 197, 411
 national character data 70
 national character string
 fixed-size 70
 variable-size 70
 native threading 367
 NCHAR 70
 NetBackup linkage facility 282, 411
 new page allocate mode 142, 411

NO SPLIT option

CREATE TABLE 75

CREATE TYPE 75

no transaction loss (no data loss) 412

no-log mode 265, 412

no-split option 74, 412

node 412

non-partitioning key index 107

non-stop service, support for 9

normal BES 312, 412

normal BES unit 312

normalization 71

NOT NULL constraint 176, 412

NOT NULL option (CREATE TABLE) 176

null value 412

numbering 95

numeric data 70

NVARCHAR 70

O

object 412

object definition event 342

object operation event 343

object relational database 412

expansion into 118

ODBC driver 19

ODBC-compatible application 19

OLE DB provider 19

OLE DB-compatible application 19

OLTP product, linkage to 42

one-phase commit 256, 257

one-to-one system switchover configuration 319

one-way alternating configuration 321

OPEN statement 152

OpenTP1 42

operation releasing check of unload status 413

operation unloading system log 413

operation without unloading system log 413

optimizing based on cost 102, 413

option program product 24

ORDER BY clause 158

overload 167, 413

override 129, 130, 413

P

page 138, 413

allocating 147

designing 146

free page 146

full page 146

ratio of unused area in 146

releasing 148

status of 146

unused page 146

used free page 146

used page 146

page compaction 295

page locking 262

pair 414

pair logical volume group 414

pair logical volumes 414

pair status 414

pair volumes 414

partition 206

partition storage conditions, changing 392

partitioning key 76, 80

partitioning key index 107, 414

partitioning storage conditions, changing 91

password character string restrictions 345

password restrictions, specifiable 345

password-invalid account lock state 346, 416

PCTFREE option

page unused area ratio 147

segment free page ratio 141

pd_assurance_table_no operand 144

pd_check_pending operand 179, 182

pd_dbbuff_lru_option operand 239

pd_dbbuff_rate_updpage operand 238

pd_dbsync_point operand 238, 239

pd_dfw_awt_process operand 238

pd_indexlock_mode operand 114

pd_jp1_event_level operand 50

pd_jp1_use operand 50

pd_large_file_use operand 208

pd_max_ard_process operand 238

pd_max_list_count operand 199

pd_max_list_users operand 199

pd_mode_conf operand 229

- pd_pageaccess_mode operand 240
 - pd_trn_commit_optimize operand 257
 - pdbuffer operand 236, 238
 - m option of 237
 - p option of 237
 - w option of 238, 240
 - pdbufs command 299
 - pdbufmod command 236
 - pdchgconf command 225
 - pdconfchk command 225
 - pdconstek command 184
 - pdcopy command (-M option) 272
 - pddbfrz command 280
 - PDDBLOG operand 265
 - pdfmkfs command 208
 - pdlbuffer operand 243
 - p option of 237
 - pdloginit command
 - creating synchronization point dump files 212
 - creating system log files 210
 - pdlogls command 275
 - pdlogunld command 273
 - pdpgbfon 241
 - pdrbal command 84
 - pdreclaim command 294
 - pdstart command 226
 - pdstop command 226
 - pdstsinit command
 - creating server status files 213
 - creating unit status files 213
 - permit update status 281
 - permitted number of consecutive certification failures 347
 - planned system switchover 318
 - plug-in 349, 417
 - setup 358
 - plug-in architecture 350
 - plug-in index 101, 359, 417
 - delayed batch creation of 359, 396
 - POSIX library version 417
 - post-source 417
 - PR, lock mode 262
 - pre-update log acquisition mode 265, 418
 - predefined data type 417
 - prediction level 1 (facility for predicting reorganization time) 291
 - prediction level 2 (facility for predicting reorganization time) 291
 - prefetch facility 236, 417
 - prepare processing 258
 - preprocessing 417
 - primary key 73, 418
 - PRIMARY KEY option (CREATE TABLE) 73
 - primary system 310
 - PRIVATE 133
 - private RDAREA 63, 64
 - private user RDAREA 332
 - privilege control event 342
 - procedure 163, 168
 - process 418
 - products that handle multimedia information, linkage to 54
 - prohibit only one type of characters 346
 - prohibit specification of authorization identifier 346
 - PROTECTED 133
 - protection mode 418
 - PU, lock mode 262
 - PUBLIC 133
 - public RDAREA 63, 64
 - public user RDAREA 332
 - PURGE TABLE statement 154
- Q**
- quantified predicate 157
- R**
- RAID Manager instance 418
 - range specification recovery 418
 - rapid grouping facility 188, 419
 - rapid system switchover facility 325, 419
 - raw I/O facility 207
 - RD node 419
 - RD node name 419
 - RDAREA 62, 419
 - adding 299
 - automatic extension of 299, 420
 - closing 393

Index

- creating 64
 - expanding 299
 - moving 301
 - relationship between RDAREA and index 62
 - relationship between RDAREA and table 62
 - reorganization by 286
 - shutting down 420
 - types of 62
 - usage privilege 332
- reactivating system 313
- read-only 43
- Real Time SAN Replication 420
- rebalancing facility 83
 - for hash row partitioning 402
- rebalancing utility 84
- recovery
 - of database 270
 - to most recent synchronization point before failure occurred 271
 - to the point backup was made 271
- RECOVERY operand (CREATE TABLE) 266
- recovery-unnecessary front-end server 327, 420
- recovery-unnecessary front-end server unit 327
- reduced activation 229, 420
- reference buffer 233, 421
- reference-only back-end server 421
- referenced table 178
- referencing table 178
- referencing-permitted backup-hold 280
 - update WAIT mode 280
- referencing-permitted mode 273
- referencing/updating-impossible mode 273
- referential constraint 178, 421
- registry facility 421
 - initialization of 358
- registry information 64, 358
- registry LOB RDAREA 64, 421
- registry management table 358
- registry RDAREA 64, 421
- regular expression 421
- regular unit 314, 422
- reloading 285
- remote site 422
- reorganization
 - by schema 422
 - with synchronization points set 288, 422
- reorganization processing
 - by RDAREA 286
 - by schema 287
 - by table 286
 - with synchronization point 288
- reorganization time prediction data
 - analyzing 291
 - getting 290
- repetition column 96, 422
- replication facility 31, 423
 - example of using 32
 - products required for 34
- reserved word 423
- resource manager 42
- RETURN statement 167
- RM 42
- role 40
- rollback 230, 256, 271, 423
- rollforward 230
- root page 101
- routine 168
- row 68
- row locking 262
- row partitioning 76, 103, 424
 - among servers 82, 106
 - within server 82, 105
- row-partitioned index 103
- row-partitioned table 76
- running system 310

S

- scheduled database maintenance day 290
- schema 67, 424
 - reorganization by 287
- schema definition privilege 332, 424
- search condition 155
- second dimension partitioning column 85
- secondary system 310
- security audit facility 334
- security facility 330, 424
- segment 138, 424
 - allocating 141

- designing 140
- free segment 140
- full segment 140
- ratio of free pages in 140
- releasing 141
- status of 140
- unused segment 140
- used free segment 140
- used segment 140
- SEGMENT REUSE option 144
- SELECT privilege 333
- SELECT statement 152
- server failure 317
- server machine 424
- server mode 316, 424
- server status file 212
- session security event 341
- SGML document registration 354
- SGML plug-in 352
- SGML structured text data 118
- shared disk unit 316
- shared index 425
- Shared Nothing method 2
- shared RDAREA 64, 425
- shared table 99, 425
- simple setup tool 9
- single column index 102
- single server, HiRDB/Single Server 11
- single-column partitioning 107
- single-phase optimization 43
- single-precision floating point 70
- site status 425
- skipped effective synchronization point dump monitoring facility 425
- SMALLFLT 70
- SMALLINT 70
- snapshot method 240
- space conversion facility 303, 425
- spatial data 56
- SQL 17, 150
 - execution method of 150
 - in HiRDB, use of 150
 - object 63, 168, 426
 - optimization of 191
- SQL connection 425
- SQL extension optimizing option 191, 426
- SQL extension optimizing option facility
 - application of optimizing mode 2 based on cost 195
 - forced foreign server execution of SQL statements that include direct product 195
 - hash-execution of hash join or subquery 195
 - suppression of foreign server execution of SQL statements that include join 195
 - suppression of unconditionally generated derived rapid search conditions that can be executed on foreign servers 196
- SQL optimization option 191, 426
- SQL optimization option facility
 - application of scalar operation-included key conditions 195
 - creating multiple SQL objects 192
 - derivation of rapid search conditions 194
 - for batch acquisition from functions provided by plug-ins 195
 - forced nest-loop-join 192
 - forced table scan 194
 - forcing use of multiple indexes 194
 - increasing number of floatable server candidates 193
 - increasing target floatable servers 192
 - limiting target floatable servers 193
 - prioritized nest-loop-join 192
 - priority of OR multiple index use 193
 - rapid grouping facility 193
 - separating data collecting servers 194
 - suppressing creation of update-SQL work tables 194
 - suppressing index use 194
 - suppressing use of AND multiple indexes 193
- SQL optimization specification 191, 427
- SQL optimization specification facility
 - for index utilization 192
 - for join method 192
 - for subquery execution method 192
- SQL preprocessor 427
- SQL runtime warning output facility 427

- SQL stored function 163, 427
 - SQL stored procedure 163, 427
 - SR, lock mode 262
 - standby system 310
 - standby system switchover facility 427
 - overview of 310
 - standby-less system switchover (1:1) facility 312, 428
 - standby-less system switchover (effects distributed) facility 313, 428
 - system configuration examples of 322
 - standby-less system switchover facility 428
 - overview of 311
 - startup mode 226
 - forced startup 227
 - normal startup 226
 - restart 226
 - static registration 46
 - static SQL 428
 - statistics log file 428
 - status file 212, 428
 - for log application processing 429
 - for transaction processing 429
 - storage condition specification 77
 - stored function 63, 163, 429
 - application of 166
 - invoking 167
 - overloading of 167
 - re-creating 168
 - stored procedure 63, 163, 429
 - calling 167
 - re-creating 168
 - structured repetition predicate 97
 - SU, lock mode 262
 - subquery 157
 - substitutability 128, 429
 - subtype 126, 429
 - Sun Java System Directory Server linkage facility 38, 430
 - Sun ONE Console 41
 - super type 128, 430
 - suppress option 74
 - SUPPRESS option (CREATE TABLE) 74
 - synchronization point 210, 254, 271, 430
 - synchronization point dump 210
 - synchronization point dump file 210, 430
 - for log application processing 430
 - for transaction processing 430
 - synchronization point dump processing 8
 - synchronization point lines count 288
 - synchronous copy 431
 - synchronous group 431
 - synchronous pair volume 431
 - system administrator security event 341
 - system failure 317
 - system file 209, 431
 - component comprising 214
 - configuration of (HiRDB/Parallel Server) 216
 - configuration of (HiRDB/Single Server) 215
 - system generator 431
 - system log 209
 - unloading 273
 - system log application, preparation for 417
 - system log file 209, 431
 - automatic log unloading facility for 276
 - swapping 274
 - system manager, HiRDB/Parallel Server 13
 - system RDAREA 63, 432
 - system reconfiguration command 225
 - system switchover facility
 - configurations of 318
 - overview of 310
 - system switchover time, comparing 326
- T**
- table 68
 - containing large quantity of data, reorganizing 288
 - examples of row partitioning of 82
 - execution unit for reorganization of 286
 - matrix partitioning of 85
 - normalization of 71
 - reorganization by 286
 - reorganization of 285
 - row partitioning definition of 83
 - row partitioning of 76
 - table access privilege (role) 40

table rebalancing 84
 table reorganization 432
 tape device access facility 432
 termination mode 227

- abnormal termination 228
- forced termination 228
- normal termination 228
- planned termination 228

 TIME 70
 time 70
 time data 70
 time interval data 70
 time-stamp data 70
 TIMESTAMP 70
 TM 42
 TPBroker 42
 transaction 432

- control of 254
- startup of 255
- termination of 255
- transfer 43

 transaction execution site 432
 transaction manager 42

- registration in 46

 transaction transfer 259
 trigger 174, 433
 trigger action search condition 174
 trigger event SQL 174
 trigger SQL statement 174
 TUXEDO 42
 two-phase commit 258
 two-to-one system switchover configuration 319

U

UAP transaction management under OLTP
 environment 259
 unbalanced index split 110, 433
 UNBALANCED SPLIT option

- CREATE INDEX 112
- CREATE TABLE 112

 unfinished index 113
 UNION 158
 UNIQUE CLUSTER KEY option (CREATE
 TABLE) 176

UNIQUE option (CREATE INDEX) 176
 uniqueness constraint 176, 433
 unit 433

- HiRDB/Parallel Server consisting of 13
- HiRDB/Single Server consisting of 11

 unit controller 433
 unit status file 212
 unload log file 270, 433

- creating 273
- relationship between backup and unload log
 file 275
- retaining 275

 unload statistics log file 433
 unload-wait 274
 unloading 285
 UNTIL DISCONNECT (LOCK statement) 264
 unused page 433
 unused segment 433
 updatable back-end server 433
 updatable backup-hold 280

- WAIT mode 280

 updatable column 94
 updatable mode 273
 updatable online reorganization 434
 update buffer 233, 434
 update copy 434
 UPDATE facility using arrays 189
 UPDATE privilege 333
 UPDATE statement 153
 used free page

- being created, process of 297
- releasing 148, 294
- reusing 294

 used free segment

- releasing 141, 297
- reusing 297

 used page 434
 used segment 434
 user authentication 39
 user LOB RDAREA 64, 434
 user mapping 435
 user privilege 330
 user RDAREA 63, 435
 user server hot standby 325, 435

Index

user-defined function 168
user-defined type 434
utility operation event 343
utility special unit 435

V

value 435
VARCHAR 70
view table 98, 435
volume attribute 436

W

Web Page Generator 56
WebLogic Server 42
WITHOUT LOCK (SELECT statement) 263
WITHOUT ROLLBACK option (CREATE
TABLE) 95
work table file 217, 436
 creating HiRDB file system area for 218
 operation requiring 218
 SQL statement requiring 217
WRITE specification 245

X

X/Open XA interface 42
XA interface, transaction control based on 254
XDM/RD E2 connection facility 5
XML document registration 354
XML server, linkage to 36
XML structured text data 118

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)
