For UNIX Systems
Scalable Database Server

HiRDB Version 8

System Operation Guide Part I

3000-6-354(E)

HITACHI

■ **Relevant program products**

List of program products:

For the HP-UX 11.0, HP-UX 11i, or HP-UX 11i V2 (PA-RISC) operating system:

P-1B62-1182 HiRDB/Single Server Version 8 08-00
P-1B62-1382 HiRDB/Parallel Server Version 8 08-00
P-1B62-1582 HiRDB/Single Server Version 8(64) 08-00
P-1B62-1782 HiRDB/Parallel Server Version 8(64) 08-00
P-1B62-1B82 HiRDB/Run Time Version 8 08-00
P-1B62-1C82 HiRDB/Developer's Kit Version 8 08-00
P-1B62-1D82 HiRDB/Run Time Version 8(64) 08-00
P-1B62-1E82 HiRDB/Developer's Kit Version 8(64) 08-00
P-F1B62-11823 HiRDB Staticizer Option Version 8 08-00
P-F1B62-11825 HiRDB Non Recover Front End Server Version 8 08-00
P-F1B62-11826 HiRDB Advanced High Availability Version 8 08-00
P-F1B62-11827 HiRDB Advanced Partitioning Option Version 8 08-00

For the HP-UX 11i V2 (IPF) operating system:

P-1J62-1582 HiRDB/Single Server Version 8(64) 08-00
P-1J62-1782 HiRDB/Parallel Server Version 8(64) 08-00
P-1J62-1D82 HiRDB/Run Time Version 8(64) 08-00
P-1J62-1E82 HiRDB/Developer's Kit Version 8(64) 08-00
P-F1J62-11823 HiRDB Staticizer Option Version 8 08-00
P-F1J62-11825 HiRDB Non Recover Front End Server Version 8 08-00
P-F1J62-11826 HiRDB Advanced High Availability Version 8 08-00
P-F1J62-11827 HiRDB Advanced Partitioning Option Version 8 08-00

For the Solaris 8, 9 or 10 operating system:

P-9D62-1182 HiRDB/Single Server Version 8 08-00
P-9D62-1382 HiRDB/Parallel Server Version 8 08-00
P-9D62-1582 HiRDB/Single Server Version 8(64) 08-00
P-9D62-1782 HiRDB/Parallel Server Version 8(64) 08-00
P-9D62-1B82 HiRDB/Run Time Version 8 08-00
P-9D62-1C82 HiRDB/Developer's Kit Version 8 08-00
P-9D62-1D82 HiRDB/Run Time Version 8(64) 08-00
P-9D62-1E82 HiRDB/Developer's Kit Version 8(64) 08-00
P-F9D62-11823 HiRDB Staticizer Option Version 8 08-00
P-F9D62-11825 HiRDB Non Recover Front End Server Version 8 08-00
P-F9D62-11826 HiRDB Advanced High Availability Version 8 08-00
P-F9D62-11827 HiRDB Advanced Partitioning Option Version 8 08-00

For the AIX(R) 5L V5.1, V5.2 or V5.3 operating system:

P-1M62-1182 HiRDB/Single Server Version 8 08-00
P-1M62-1382 HiRDB/Parallel Server Version 8 08-00
P-1M62-1582 HiRDB/Single Server Version 8(64) 08-00
P-1M62-1782 HiRDB/Parallel Server Version 8(64) 08-00
P-1M62-1B82 HiRDB/Run Time Version 8 08-00
P-1M62-1C82 HiRDB/Developer's Kit Version 8 08-00

P-1M62-1D82 HiRDB/Run Time Version 8(64) 08-00

P-1M62-1E82 HiRDB/Developer's Kit Version 8(64) 08-00

P-F1M62-11823 HiRDB Staticizer Option Version 8 08-00

P-F1M62-11825 HiRDB Non Recover Front End Server Version 8 08-00

P-F1M62-11826 HiRDB Advanced High Availability Version 8 08-00

P-F1M62-11827 HiRDB Advanced Partitioning Option Version 8 08-00

For the Red Hat Linux 7.1, Red Hat Linux 7.2, Red Hat Enterprise Linux AS 2.1, Red Hat Enterprise Linux AS 3 (x86), Red Hat Enterprise Linux ES 3 (x86), Red Hat Enterprise Linux AS 4 (x86), Red Hat Enterprise Linux ES 4 (x86), Red Hat Enterprise Linux AS 3 (AMD64 & Intel EM64T),* Red Hat Enterprise Linux AS 4 (AMD64 & Intel EM64T), or Red Hat Enterprise Linux ES 4 (AMD64 & Intel EM64T) operating system:

P-9S62-1182 HiRDB/Single Server Version 8 08-00

P-9S62-1382 HiRDB/Parallel Server Version 8 08-00

P-9S62-1B82 HiRDB/Run Time Version 8 08-00

P-9S62-1C82 HiRDB/Developer's Kit Version 8 08-00

P-F9S62-11823 HiRDB Staticizer Option Version 8 08-00

P-F9S62-11825 HiRDB Non Recover Front End Server Version 8 08-00

P-F9S62-11826 HiRDB Advanced High Availability Version 8 08-00

P-F9S62-11827 HiRDB Advanced Partitioning Option Version 8 08-00

* Only operating systems that run on the Intel EM64T are supported.

For the Red Hat Enterprise Linux AS 3 (AMD64 & Intel EM64T),* Red Hat Enterprise Linux AS 4 (AMD64 & Intel EM64T), or Red Hat Enterprise Linux ES 4 (AMD64 & Intel EM64T) operating system:

P-9W62-1182 HiRDB/Single Server Version 8 08-00

P-9W62-1382 HiRDB/Parallel Server Version 8 08-00

P-9W62-1B82 HiRDB/Run Time Version 8 08-00

P-9W62-1C82 HiRDB/Developer's Kit Version 8 08-00

* Only operating systems that run on the Intel EM64T are supported.

For the Red Hat Enterprise Linux AS 3 (IPF) or Red Hat Enterprise Linux AS 4 (IPF) operating system:

P-9V62-1182 HiRDB/Single Server Version 8 08-00

P-9V62-1382 HiRDB/Parallel Server Version 8 08-00

P-9V62-1B82 HiRDB/Run Time Version 8 08-00

P-9V62-1C82 HiRDB/Developer's Kit Version 8 08-00

P-F9V62-11823 HiRDB Staticizer Option Version 8 08-00

P-F9V62-11825 HiRDB Non Recover Front End Server Version 8 08-00

P-F9V62-11826 HiRDB Advanced High Availability Version 8 08-00

P-F9V62-11827 HiRDB Advanced Partitioning Option 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 *Software Information* or *Before Installing*.

■ Trademarks

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.

The following program products include material copyrighted by Sun Microsystems, Inc.: P-9D62-1182, P-9D62-1382, P-9D62-1582, P-9D62-1782, P-9D62-1B82, P-9D62-1C82, P-9D62-1D82, P-9D62-1E82, P-F9D62-11823, P-F9D62-11825, P-F9D62-11826, and P-F9D62-11827.

The following program products include material copyrighted by UNIX System Laboratories, Inc.: P-9D62-1182, P-9D62-1382, P-9D62-1582, P-9D62-1782, P-9D62-1B82, P-9D62-1C82, P-9D62-1D82, P-9D62-1E82, P-F9D62-11823, P-F9D62-11825,

P-F9D62-11826, and P-F9D62-11827.

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.

Printed in Japan.

■ Edition history

Edition 1 (3000-6-354(E)): March 2007

■ Copyright

Preface

This manual describes the operating procedures for the HiRDB Version 8 scalable database server system.

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 UNIX or Linux systems
- A basic knowledge of SQL

This manual is based on the *HiRDB Version 8 Description*, which should be read before reading this manual.

Organization of this manual

This manual is organized as follows:

Chapter *1. HiRDB Startup and Termination*

Explains the procedures for starting and terminating HiRDB, as well as the procedures for starting and terminating units and servers.

Chapter *2. Security Definition*

Explains the HiRDB-provided security features and their operating procedures.

Chapter *3. Handling System Log Files*

Explains the handling of system log files.

Chapter *4. Handling Synchronization Point Dump Files*

Explains the handling of synchronization point dump files.

Chapter *5. Handling Status Files*

Explains the handling of status files.

Chapter *6. Backup Procedures*

Explains the procedures for making backups.

Chapter *7. Operation Without Acquiring a Database Update Log*

Explains the procedures for executing a UAP or utility in the following modes:

- Pre-update log acquisition mode
- No-log mode

Chapter *8. Obtaining the System Operating Environment (Monitoring the System Status)*

Explains how to obtain information about the system operating environment by monitoring the system status.

Chapter *9. Modifying the System Operating Environment*

Explains how to modify the HiRDB system environment definitions.

Chapter *10. Handling HiRDB File System Areas*

Explains how to create and delete HiRDB file system areas, obtain backups, and how to restore HiRDB file system areas.

Chapter *11. Modifying the System Configuration*

Explains how to modify the unit or server configuration of a HiRDB/Parallel Server; also explains the procedures for migrating from a HiRDB/Single Server to a HiRDB/Parallel Server.

Chapter *12. Migrating Resources Between Systems*

Explains how to migrate tables and stored procedures to another HiRDB system.

Chapter *13. Handling Tables*

Explains the procedures for reorganizing tables, modifying table definitions, deleting tables, deleting abstract data types, and managing lists, as well as the space conversion facility and the facility for conversion to a decimal signed normalized number.

Chapter *14. Handling Indexes*

Explains the procedures for reorganizing and deleting indexes, as well as unbalanced index splitting and delayed batch creation of plug-in indexes.

Chapter *15. Handling RDAREAs*

Explains the procedures for adding, deleting, expanding, reinitializing, and automatically increasing RDAREAs, and how to modify the RDAREA opening trigger.

Chapter *16. Handling Stored Procedures and Stored Functions*

Explains the procedures for registering stored procedures and stored functions, and the handling of stored procedures and stored functions when they are no longer valid.

Chapter *17. Using Java Stored Procedures and Java Stored Functions*

Explains the environment setup and operating procedures for using Java stored procedures and Java stored functions.

Chapter *18. Error Handling Procedures*

Explains the handling of errors.

Chapter *19. Database Recovery Procedures*

Explains the procedures for recovering a database in the event that it is damaged by an error.

Chapter *20. Obtaining Tuning Information*

Explains the procedures for obtaining tuning information.

Chapter *21. Tuning*

Explains the tuning procedures.

Chapter *22. Using the Security Audit Facility*

Explains the environment setup and operating procedures for the security audit facility.

Chapter *23. Using the Connection Security Facility*

Explains how to use the connection security facility, which is designed to enhance the security of HiRDB systems.

Chapter *24. Using the Directory Server Linkage Facility*

Explains the environment setup and operating procedures for the Directory Server linkage facility.

Chapter *25. Using the System Switchover Facility*

Explains the environment setup and operating procedures for the system switchover facility.

Chapter *26. Using the Facility for Monitoring MIB Performance Information*

Explains how to use the facility for monitoring MIB performance information, which uses MIB to collect HiRDB operation information.

Chapter *27. Using a Distributed Database (applicable to HP-UX and AIX 5L only)*

Explains the environment setup and operating procedures for distributed databases.

Appendix *A. Q & A*

Provides in Q&A format the answers to frequently asked questions.

Appendix *B. Operations When Using a DVD-RAM Library Device*

Explains the procedures for using a DVD-RAM library unit as a storage device.

Appendix *C. Information Needed for Troubleshooting*

Explains the information needed to use problem-solving support or Q&A support services.

Appendix *D. Notes on Running HiRDB Around the Clock*

Explains the procedures for and provides notes about running HiRDB continuously around the clock.

Appendix *E. Using Performance Improvement Facilities*

Explains facilities designed to enhance system performance and how to use them.

Related publications

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

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 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 Windows)

- *For Windows Systems HiRDB Version 8 Description* (3020-6-351(E))
- *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 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.

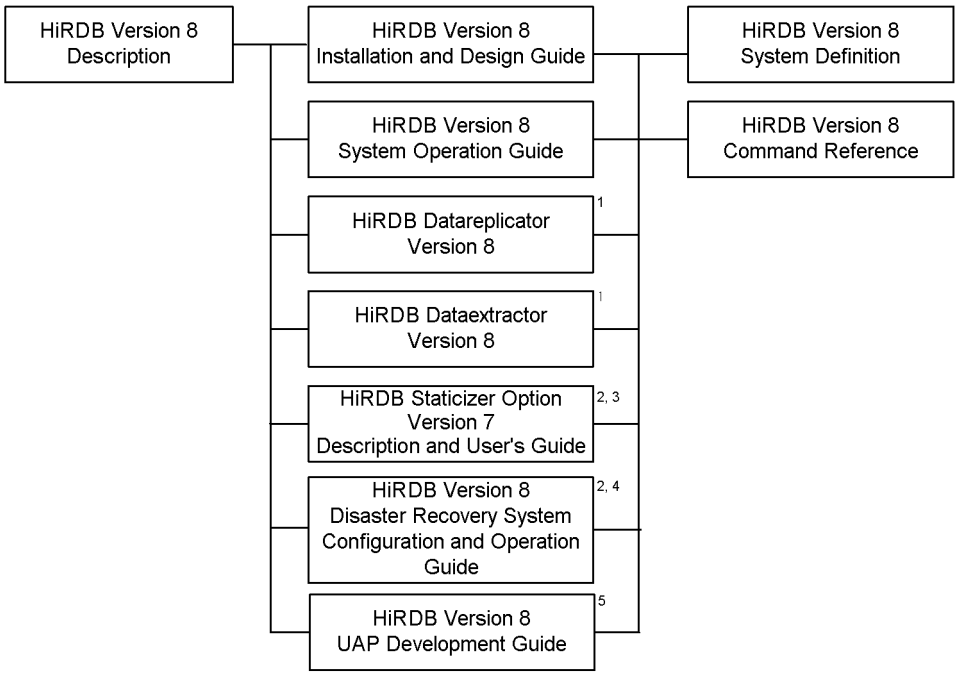
Others

- *HiRDB External Data Access Version 7 Description and User's Guide* (3000-6-284(E))
- *Distributed Database System DF/UX* (3000-3-248(E))
- *JP1 Version 6 JP1/VERITAS NetBackup v4.5 Agent for HiRDB License Description and User's Guide* (3020-3-D79(E))
- *HiCommand Tuning Manager - Agent for RAID* (3020-3-E92(E))
- *HiCommand Tuning Manager - Agent for RAID Map* (3020-3-E93(E))
- *For UNIX Systems Job Management Partner 1/Performance Management - Agent Option for Platform Description, User's Guide and Reference* (3020-3-K65(E))
- *Job Management Partner 1/Performance Management/SNMP System Observer for Extended Resource Management* (3020-3-F70(E))
- *Job Management Partner 1/Consolidated Management 2/Extensible SNMP Agent* (3020-3-F92(E))

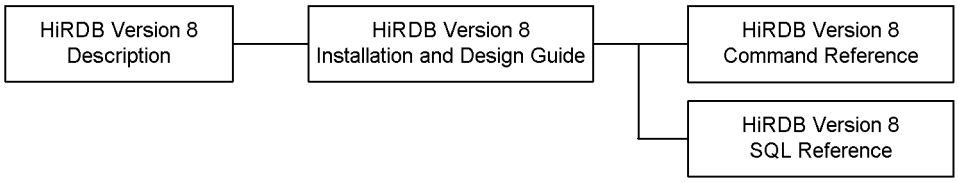
Organization of HiRDB manuals

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™ 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™ 2003, Standard Edition	Windows Server 2003	
Microsoft(R) Windows Server™ 2003, Enterprise Edition		
Microsoft(R) Windows Server™ 2003 R2, Standard Edition	Windows Server 2003 R2 or Windows Server 2003	
Microsoft(R) Windows Server™ 2003 R2, Enterprise Edition		
64 bit Version Microsoft(R) Windows Server™ 2003, Enterprise Edition (IPF)	Windows Server 2003 (IPF) or Windows Server 2003	
Microsoft(R) Windows Server™ 2003, Standard x64 Edition	Windows Server 2003 or Windows Server 2003 x64 Editions	Windows (x64)
Microsoft(R) Windows Server™ 2003, Enterprise x64 Edition		
Microsoft(R) Windows Server™ 2003 R2, Standard x64 Edition	Windows Server 2003, Windows Server 2003 R2 or Windows Server 2003 x64 Editions	
Microsoft(R) Windows Server™ 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

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

- Windows 2000, Windows XP, and Windows Server 2003 may be referred to collectively as *Windows*.
- The HiRDB directory path is represented as `$PDDIR`.
- The hosts file means the `hosts` file stipulated by TCP/IP (including the `/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

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

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

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

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

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

Log representations

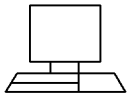
The OS log is referred to generically as *syslogfile*. syslogfile is the log output destination specified in `/etc/syslog.conf`. Typically, the following files are specified as syslogfile.

OS	File
HP-UX	`/var/adm/syslog/syslog.log`
Solaris	`/var/adm/messages` or `/var/log/syslog`
AIX 5L	`/var/adm/ras/syslog`
Linux	`/var/log/messages`

Conventions: Diagrams

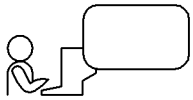
This manual uses the following conventions in diagrams:

- Workstation or personal computer



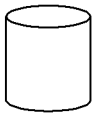
- I/O operation

- Screen display



- Program or server

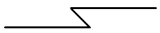
- File or magnetic disk



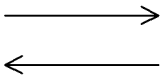
- Magnetic tape

- CMT or DAT

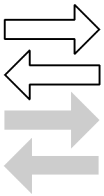
- Communication line



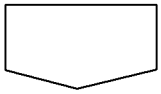
- Flow of control



- Flow of data

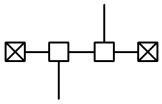


- Work procedure

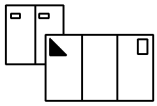


- Network

- LAN



- Mainframe



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: • From the **File** menu, choose **Open**. • Click the **Cancel** button. • In the **Enter name** entry box, type your name.
Italics	Italics are used to indicate a placeholder for some actual text provided by the user or system. Italics are also used for emphasis. For example: • Write the command as follows: `copy` *source-file target-file* • Do *not* delete the configuration file.
`Code font`	A code font indicates text that the user enters without change, or text (such as messages) output by the system. For example: • At the prompt, enter `dir`. • Use the `send` command to send mail. • The following message is displayed: `The password is incorrect.`

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:

Store**D**atabase [temp|perm] (*database-name* ...)

The following table lists the conventions used in syntax explanations. The syntactical characters described below are used to provide a clear explanation of code syntax; you do not actually enter these characters.

Example font or symbol	Convention
`StoreDatabase`	Code-font characters must be entered exactly as shown.
database-name	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.

Example font or symbol	Convention
perm	Underlined characters indicate the default value.
[ ]	Square brackets enclose an item or set of items whose specification is optional. Example: pdbuffer [-p] This example indicates that you can specify either pdbuffer or pdbuffer -p.
|	Only one of the options separated by a vertical bar can be specified at the same time. Example: pdlogadfg -d sys | spd This example indicates that you can specify either sys or spd for the -d option.
. . .	An ellipsis (...) indicates that the item or items enclosed in ( ) or [ ] immediately preceding the ellipsis may be specified as many times as necessary. Example: pdbuffer -r *RDAREA-name* [,*RDAREA-name*] . . . This example indicates that following -r, you can specify *RDAREA-name* as many times as necessary.
( )	Parentheses indicate the range of items to which the vertical bar (|) or ellipsis (...) is applicable.
{ }	A single pair of curly brackets encloses multiple items, one of which you must specify. Example: pdbuffer [{-r *RDAREA-name* | -i *authorization-identifier*.*index-identifier* | -o}] This example indicates that you must specify one of the three options enclosed by the curly braces: -r *RDAREA-name*, -i *authorization-identifier*.*index-identifier*, or -o.
{{ }}	A double pair of curly brackets encloses multiple items, all of which you can specify multiple times as a unit. Example: {{pdbuffer -a *option-name*}} This example indicates that you can specify the above multiple times as follows: pdbuffer -a *option-name* pdbuffer -a *option-name*
~	A swung dash precedes the attributes of a user-specified value.
<< >>	A double pair of angle brackets encloses the default value assumed by the system when the specification is omitted.
< >	A single pair of angle brackets encloses the syntax element notation for a user-specified value.
(( ))	A double pair of parentheses encloses the permitted range of values that can be specified.

Syntactical element symbols

The following syntactical element symbols are used in this manual:

Syntactical element symbol	Meaning
<*alphabetics*>	The alphabetic characters (A-Z and a-z) and the underline (_)
<*alphabetics and special characters*>	The alphabetic characters (A-Z and a-z) and the special characters #, @, and \
<*alphanumerics*>	Alphabetics and the numeric digits (0-9)
<*alphanumerics and special characters*>	Alphabetics, special characters, and numeric digits
<*unsigned integer*>	Numeric value
<*hexadecimal*>	Numeric digits and A-F (or a-f)
<*identifier*>[1]	Alphanumeric character string beginning with an alphabetic character
<*symbolic name*>	Alphanumeric character string beginning with an alphabetic character or a special character
<*character string*>	Any string of characters
<*path name*>[2]	Includes symbolic names, forward slashes (/), and periods (.).

Note

All alphabetic characters must be single-byte characters. The syntactical element symbols are case sensitive.

[1] An RDAREA name is an alphanumeric character string beginning with an alphabetic character or special character, and can include alphanumeric characters, underscores (_), and spaces. If an RDAREA name includes a space, the entire name must be enclosed in double quotation marks (").

A host name is a character string that can include alphabetic characters (A to Z, a to z), numeric characters, periods (.), hyphens (-), and underscores (_). A host name can begin with a numeric character.

[2] Path names depend on the OS being used. Do not use a backslash (\) in HiRDB file system area names.

Notations used in formulas

The following notations are used in computational expressions:

Symbol	Meaning
↑ ↑	Round up the result to the next integer. Example: The result of ↑34 ÷ 3↑ will be 12.

Symbol	Meaning
↓ ↓	Discard digits following the decimal point. Example: The result of ↓34 ÷ 3↓ will be 11.
MAX	Select the largest value as the result. Example: The result of MAX(3 × 6, 4 + 7) will be18.
MIN	Select the smallest value as the result. Example: The result of MIN(3 × 6, 4 + 7) will be 11.

Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

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

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

Notes on printed manuals

Please note that even though the printed manuals are separated into Part I and Part II, the chapters and page numbers sequentially continue from Part I to Part II. Also, please note that the index is only included in Part II.

Contents

List of figures

List of tables

Chapter

1. HiRDB Startup and Termination

This chapter describes the procedures for starting and terminating HiRDB.

In the case of a HiRDB/Parallel Server, HiRDB can be started and terminated at each unit and server.

This chapter contains the following sections:

1.1 Startup

Executor: HiRDB administrator

You use the `pdstart` command to start HiRDB. This section explains the following items related to the HiRDB startup modes and startup methods:

- Startup modes
- Server machine where the `pdstart` command is executed
- Automatic startup
- Reduced activation (applicable to HiRDB/Parallel Server only)
- Example (HiRDB normal startup)
- Checking for startup completion

1.1.1 Startup modes

HiRDB employs the concept of the startup modes listed in Table 1-1. The `pdstart` command's options depend on the startup mode.

Table 1-1: HiRDB startup modes

Startup mode	Execution command	Description of the startup mode	Previous termination mode
Normal startup	`pdstart`	This is the usual startup mode. In this mode, information that was in effect during the previous session is not inherited. However, the error shutdown status of RDAREAs is inherited.	Normal termination
Restart[1]		This is the mode that is used (automatically) when the previous termination mode was one of those listed to the right. In this mode, the information that was in effect during the previous session is inherited.	• Planned termination • Forced termination • Abnormal termination

Startup mode	Execution command	Description of the startup mode	Previous termination mode
Forced startup[2]	`pdstart dbdestroy`	This mode should not be used unless it is absolutely necessary. When HiRDB cannot be restarted, this mode is used to start HiRDB forcibly.	—
—[3]	`pdstart -i`	This mode should not be used unless it is absolutely necessary; it is used when a database is being reinitialized.	
	`pdstart -r`	This mode should not be used unless it is absolutely necessary; it is used in the event of an error in the master directory RDAREA.	
	`pdstart -a`	This mode should not be used unless it is absolutely necessary; it is used when the front-end server is in SUSPEND status.	

[1] Section *1.7.2 Notes on planned termination, forced termination, and abnormal termination* should be read before HiRDB is restarted.

[2] Section *1.6.2 Notes on forced startup of HiRDB (or a unit)* should be read before forced startup is executed.

[3] For details on using these startup methods, see *1.3 Special startup procedures*.

1.1.2 Server machine where the pdstart command is executed

To start a HiRDB/Single Server, the `pdstart` command must be executed at the server machine where the single server is defined. To start a utility special unit, the `pdstart` command must be executed at the server machine where the utility special unit is defined.

To start a HiRDB/Parallel Server, the `pdstart` command must be executed at the server machine where the system manager is defined.

1.1.3 Automatic startup

HiRDB can be started automatically by specifying the `pd_mode_conf` operand in the system common definition. Automatic startup means that HiRDB is started automatically when OS is started. The method of starting HiRDB by entering the `pdstart` command is called *manual startup*.

Notes

- If HiRDB (or the unit) terminates abnormally when HiRDB has been set up with the `pd_mode_conf` operand to start automatically, it will restart automatically. However, if it cannot restart in three consecutive attempts, it

will terminate abnormally.

- If HiRDB terminates abnormally during startup or termination processing, manual startup must be used to start it.
- In the case of automatic startup of a HiRDB/Parallel Server (`pd_mode_conf=AUTO` specified), all units must start within 20 minutes from the time the first unit starts. If all units do not start within 20 minutes, the HiRDB startup process will terminate. This 20-minute time limit can be changed with the `pd_reduced_check_time` operand.

  Note that the time limit does not apply when HiRDB is being restarted after the unit or the operating system has terminated abnormally.

1.1.4 Reduced activation (applicable to HiRDB/Parallel Server only)

Normally, a HiRDB/Parallel Server cannot be started unless all its units start. However, the reduced activation facility enables a HiRDB/Parallel Server to be started at the available units even if some units cannot be started due to an error. To use this facility, the following operands must be specified:

- `pd_start_level`
- `pd_start_skip_unit`

For details on the reduced activation facility, see *18.15 Handling of reduced activation (HiRDB/Parallel Server only)*.

1.1.5 Example (HiRDB normal startup)

In this example, HiRDB is started normally.

(1) Entering the pdstart command to start HiRDB normally:

```
pdstart
```

(2) The results of executing the pdstart command are displayed:

```
139 14:56:33 unt1 _rdm     KFPS05210-I HiRDB system initialization
 process complete
```

(3) Entering the pdls command to check HiRDB's operational status:

```
pdls
```

(4) The results of executing the pdls command are displayed:

```
HOSTNAME(145750)     UNITID  SVID      STATUS   STARTTIME
k95x620              unt1    ********  ACTIVE   145632
k95x620              unt1    sds01     ACTIVE   145632
```

Explanation

`ACTIVE` in the `STATUS` column indicates that HiRDB has been started successfully.

This example of execution results of the `pdls` command is for HiRDB/Single Server.

1.1.6 Checking for startup completion

This section explains how to check whether startup of HiRDB (or a unit) has been completed.

(1) Checking for HiRDB startup completion

(a) Checking the termination status of the pdls -d svr command

After you have executed the `pdstart` command, execute the `pdls -d svr` command from any unit and note its termination status. Table 1-2 shows the action that must be taken by the HiRDB administrator based on the termination status of the `pdls -d svr` command.

Table 1-2: Termination status of the pdls -d svr command and required actions

Termination status of the pdls -d svr command	Action to be taken by the HiRDB administrator
`0`	You can execute an application (SQL).
`4`	HiRDB startup may still be underway; execute the `pdls -d svr` command at approximately 5-second intervals until the termination status is no longer `4`. You should execute the `pdls -d svr` command over a period of time roughly equivalent to the value specified in the `pd_system_complete_wait_time` operand.
`8`	An error occurred during HiRDB startup. See the message that is output to `syslogfile` and remove the error cause; then restart HiRDB.

(b) Checking the termination status of the pdstart command

To start HiRDB from the unit in which the system manager is located, execute the `pdstart` command. If the termination status of the `pdstart` command is `0`, you can execute an application (SQL).

If the termination status of the pdstart command is not 0, an error occurred during HiRDB startup. See the message that is output to syslogfile and remove the error cause; then restart HiRDB.

(2) Checking for completion of startup of a unit

After you have executed the pdstart -q command, execute the pdls -d ust command on the unit and note its termination status. Table 1-3 shows the action that must be taken by the HiRDB administrator based on the termination status reported by the pdls -d ust command.

Table 1-3: Termination status of the pdls -d ust command and required actions

Termination status of the pdls -d ust command	Action to be taken by the HiRDB administrator
0	The unit is running. If the unit where the command was executed contains a front-end server, you can execute an application (SQL) for which this unit's front-end server is specified in the PDFESHOST and PDSERVICEGRP operands in the client environment definition.
4	HiRDB startup may still be underway; execute the pdls -d ust command at approximately 5-second intervals until the termination status is no longer 4. You should execute the pdls -d ust command over a period of time roughly equivalent to the value specified in the pd_system_complete_wait_time operand.
8 or 12	An error occurred during unit startup. See the message that is output to syslogfile and remove the error cause; then restart the unit.
16	The operation may be incorrect. First use the pdsetup command to register HiRDB in the OS, then restart the unit.

1.2 Termination

Executor: HiRDB administrator

You use the pdstop command to terminate HiRDB. This section explains the following items related to the HiRDB termination modes and termination methods:

- Termination modes
- Server machine where the pdstop command is executed
- Example (HiRDB normal termination)
- Terminating HiRDB during OS shutdown

1.2.1 Termination modes

HiRDB employs the concept of the termination modes listed in Table 1-4. The pdstop command's options depend on the termination mode.

Table 1-4: HiRDB termination modes

Termination mode	Input command	Description of the termination mode
Normal termination[1]	pdstop	This is the normal termination mode. This mode prohibits CONNECT requests and terminates HiRDB once all user processes have terminated. If HiRDB cannot be terminated even by executing the pdstop command because a utility is still running, the KFPS05074-E error message is output. The pdstop command terminates with return code 8.
Planned termination[1,2,3]	pdstop -P	This mode prohibits acceptance of any new transactions and terminates HiRDB after all transactions, including utilities, have terminated.
Forced termination	pdstop -f	This mode terminates HiRDB immediately without waiting for completion of current transactions. Current transactions become subject to rollback[4] during the subsequent HiRDB restart.
Abnormal termination	—	This is the mode in which HiRDB is terminated because of an error. HiRDB is terminated immediately without waiting for completion of current transactions. Current transactions become subject to rollback[4] during the subsequent HiRDB restart.

[1] If HiRDB is linked to an OLTP system, the OLTP system must be terminated before normal or planned termination of HiRDB is executed. Otherwise, an attempt to terminate HiRDB by normal or planned termination may fail, because transaction processing cannot be executed on the OLTP system.

[2] When planned termination of HiRDB is executed, the system resources held by the HiRDB server processes are released after all the active transactions are terminated in the single server or front-end server. It may take a minute or so for this release processing to be completed after all transactions at a unit have terminated.

[3] If planned termination of HiRDB has not been completed within 15 minutes of entering the `pdstop -P` command because a utility or transaction is still executing, the `KFPS05072-W` message is output and the `pdstop -P` command is terminated with return code 4. In such a case, planned termination processing is still in effect, and HiRDB will be terminated once the utility or transaction has terminated.

[4] Current transactions become subject to rollback during a HiRDB restart except in the following cases:

- The database load utility or database reorganization utility was executing in the no-log mode.
- The UAP was executing in the no-log mode.

In such a case, the HiRDB administrator must restore the RDAREAs from backup copies or re-execute the utility after HiRDB has been restarted. For details on how to restore RDAREAs, see *19.2 Recovering a database to the point at which a backup was made*.

1.2.2 Server machine where the pdstop command is executed

To terminate a HiRDB/Single Server, the `pdstop` command must be executed at the server machine where the single server is defined. To terminate a utility special unit, the `pdstop` command must be executed at the server machine where the utility special unit is defined.

To terminate a HiRDB/Parallel Server, the `pdstop` command must be executed at the server machine where the system manager is defined.

1.2.3 Example (HiRDB normal termination)

In this example, a HiRDB/Single Server that is currently running is terminated normally.

(1) Enter the pdls command to check HiRDB's operational status:

```
pdls
```

(2) The results of executing the pdls command are displayed:

```
HOSTNAME(140814)    UNITID  SVID      STATUS   STARTTIME
k95x620             unt1    ********  ACTIVE   140812
k95x620             unt1    sds01     ACTIVE   140812
```

Explanation

ACTIVE in the STATUS column indicates that HiRDB is running.

(3) Enter the pdls command to check whether or not there are any users connected to HiRDB:

```
pdls -d prc
```

(4) The results of executing the pdls command are displayed:

```
HOSTNAME : k95x620(141028)
STATUS   PID   UID   GID   SVID   TIME      PROGRAM    C-PID

  L      205   0     0     sds01  999999
  L      201   0     0     sds01  999999
  L      194   0     0     sds01  999999
  L      206   0     0     sds01  999999
  L      198   0     0     sds01  999999
  L      142   0     0     sds01  999999
  L      209   0     0     sds01  999999
  L      212   0     0     sds01  999999
```

Explanation

- If a user is connected to HiRDB, a UAP identifier is shown in the PROGRAM column. In this example, there are no users currently connected to HiRDB. HiRDB cannot be terminated if any user is still connected.
- For example, if HiRDB SQL Executer remains in use (i.e., HiRDB is still connected), HiRDB cannot be terminated. To terminate HiRDB, you must first terminate HiRDB SQL Executer.
- The identifier that is displayed in the PROGRAM column is the UAP identification name specified in the PDCLTAPNAME operand of the client environment definition. If the PDCLTAPNAME operand was omitted, Unknown is displayed.

(5) Enter the pdstop command to terminate HiRDB normally:

```
pdstop
```

(6) The results of executing the pdstop command are displayed:

```
129 14:15:20 unt1 _rdm     KFPS01841-I HiRDB unit unt1 terminated.
 mode = NORMAL
129 14:15:20 unt1 _rdm     KFPS01850-I HiRDB system terminated.
 mode = NORMAL
```

1.2.4 Terminating HiRDB during OS shutdown

If an OS shutdown occurs during HiRDB operation, databases may be damaged depending on the shutdown's timing. This subsection describes how to terminate HiRDB when the OS is shut down. In the case of the AIX 5L version of HiRDB, an OS shutdown does not damage databases.

There are two ways to terminate HiRDB during an OS shutdown; you use the `pdsetup` command to specify the appropriate settings. Normally, you use the settings for explicit forced termination of HiRDB during an OS shutdown.

(1) Explicit forcible termination of HiRDB during an OS shutdown

For explicit forcible termination of HiRDB during an OS shutdown, execute the following `pdsetup` command:

```
pdsetup -k on HiRDB-directory-name
```

(2) Terminating HiRDB normally during OS shutdown, or terminating HiRDB during OS shutdown depending on whether or not termination of the OS is forced

To terminate HiRDB normally during OS shutdown, or to terminate HiRDB during OS shutdown depending on whether or not the OS is terminated forcibly, execute the following pdsetup command:

```
pdsetup -k off HiRDB-directory-name
```

(a) Terminating HiRDB normally during OS shutdown

Specific conditions must be met in order to terminate HiRDB normally. You must also create and register an `rc` script.

Conditions for normal termination

To terminate HiRDB normally, the following conditions must be met:

- No users can be connected to HiRDB.
- There can be no uncompleted transactions.
- In the case of a HiRDB/Parallel Server, no unit that does not contain the system manager can have been terminated forcibly or abnormally.

Creating and registering an rc script

Create and register the following `rc` script:

Contents of the rc script

Create an `rc` script that executes the following procedure:

1. Terminates all clients connected to HiRDB.
2. Executes the `pdstop` command to effect normal termination of HiRDB.
3. If normal termination fails in step 2 (return code is not 0), executes the `pdstop -f` command to terminate HiRDB forcibly.

Location where the rc script is registered

Register the created `rc` script at the following location:

Platform	Location of registration
HP-UX	`/sbin/init.d/`*xxxx* `/sbin/rc1.d/`*Kyyyxxxx* (Symbolic link to `sbin/init.d/`*xxxx*)
Solaris	`/etc/init.d/`*xxxx* `/etc/rc0.d/`*Kyyyxxxx* `/etc/rc1.d/`*Kyyyxxxx* (Symbolic link to `etc/init.d/`*xxxx*)
Linux	`/etc/init.d/`*xxxx* `/etc/rc0.d/`*Kyyyxxxx* `/etc/rc1.d/`*Kyyyxxxx* `/etc/rc6.d/`*Kyyyxxxx* (Symbolic link to `etc/init.d/`*xxxx*)

Legend:

xxxx:

Any name

Kyyyxxxx:

K indicates the script that is to run during termination. *yyy* indicates a numeric value in the range from 000 to 999; the `rc` scripts are executed in ascending order of this value. *xxxx* indicates any name.

(b) Terminating HiRDB during OS shutdown depending on whether or not the OS termination is forced

If HiRDB termination depends on whether or not termination of the OS was performed forcibly, databases may be damaged depending on when the shutdown occurs. For this reason, you should make sure that you perform explicit forced termination of HiRDB during an OS shutdown.

1.3 Special startup procedures

In addition to normal startup, restart, and forced startup, the following startup procedures are also available:

- Startup procedure used to reinitialize a database
- Startup procedure used in the event of an error in the master directory RDAREA
- Startup procedure used in the event of an error in a data dictionary RDAREA

Note

You must exercise caution when you use the HA monitor's system switchover facility on an HP-UX, or AIX-5L version of a HiRDB/Parallel Server. You must use the following procedure to release the HA monitor's monitoring status, and then you must execute a special startup procedure:

Procedure

1. Check the status of each unit (whether it is the running system or a standby system).*
2. If the primary unit is in standby status, use the HA monitor's `monswap` command to switch the systems so that the primary unit becomes the running system. In other words, return the systems to their status when use of the system switchover facility began.
3. Enter the `pdstop` command to terminate HiRDB on the running system.
4. If a HiRDB is running as a server system having no HA monitor interface (in the monitor mode), use the `monend` command of the HA monitor to stop the HiRDB on the standby system.

* If IP addresses are not inherited, the `pdls -d ha` command can be used to check the status.

1.3.1 Startup procedure used to reinitialize a database (pdstart -i)

To reinitialize a database (re-execute the database initialization utility), the `pdstart -i` command is used to start HiRDB.

When plug-ins are used

If plug-ins are being used, the `pdplugin` operand must be deleted from the system common definition before the `pdstart -i` command is executed. Otherwise, the registry information will be lost. After the database initialization utility has been executed, the registry-related environment must be redefined.

1.3.2 Startup procedure used in the event of an error in the master directory RDAREA (pdstart -r)

If an error occurs in the master directory RDAREA, HiRDB cannot be started by any of the normal startup methods. In such a case, the `pdstart -r` command must be used to start HiRDB.

When HiRDB has been started with the `pdstart -r` command, only the following utilities can be executed; no other operations are possible:

- Database recovery utility
- Database copy utility

The HiRDB administrator must use one of these utilities to restore the master directory RDAREA. Once the master directory RDAREA has been restored, HiRDB can be restarted and application processing can be resumed.

For details on the actions to be taken when HiRDB cannot be restarted due to an error in the master directory RDAREA, see *18.4.2 When HiRDB does not restart*.

1.3.3 Startup procedure used when the front-end server is in SUSPEND status due to an error in a data dictionary RDAREA (pdstart -a)

If an error occurs in a data dictionary RDAREA during HiRDB startup, the front-end server is placed in `SUSPEND` status.* In such a case, the HiRDB administrator must restore the data dictionary RDAREA and release the front-end server from SUSPEND status.

The `pdstart -a` command is used to start HiRDB, which will release the front-end server from SUSPEND status.

* This is the status in which the front-end server is waiting for recovery of the data dictionary RDAREA.

1.4 Startup and termination of a unit (applicable to HiRDB/Parallel Server only)

Executor: HiRDB administrator

When a HiRDB/Parallel Server is being used, each unit can be started and terminated independently.

(1) Unit startup procedures

HiRDB can be started at a unit with the commands listed in Table 1-5.

Table 1-5: Startup procedures for a unit

Startup mode	Execution command	Description of the startup mode	Previous termination mode
Normal startup	`pdstart -u` `pdstart -x`	This startup mode is used to restart a unit after it terminated normally during HiRDB operation. Information that was in effect during the previous session is not inherited.	Normal termination
Restart[1]		The restart mode is used (automatically) when the previous termination mode was one of those listed to the right. In this mode, the information that was in effect during the previous session is inherited.	Forced termination Abnormal termination

Forced startup[2]	`pdstart -u`
	`dbdestroy`
	`pdstart -x`
	`dbdestroy`
		This mode should not be used unless it is absolutely necessary. This mode starts the unit forcibly without restoring the database.	—

[1] Section *1.7.2 Notes on planned termination, forced termination, and abnormal termination* should be read before restarting.

[2] Section *1.6.2 Notes on forced startup of HiRDB (or a unit)* should be read before forced startup is executed.

(2) Unit termination procedures

HiRDB can be terminated at a unit with the commands listed in Table 1-6.

Table 1-6: Termination procedures for a unit

Termination mode	Input command	Description of the termination mode
Normal termination	`pdstop -u` `pdstop -x`	This mode prohibits any more `CONNECT` requests to this unit, disconnects all UAPs currently connected to the unit, then terminates the unit. If the unit cannot be terminated, even by executing the `pdstop -u` or `pdstop -x` command, because a utility is still running, the `KFPS05070-E` error message is output. The `pdstop -u` or `pdstop -x` command terminates with return code `8`.
Forced termination[1]	`pdstop -f -u` `pdstop -f -x`	This mode terminates the unit immediately without waiting for completion of current transactions. Current transaction become subject to rollback[2] during the subsequent restart.
Abnormal termination	—	This is the mode in which the unit is terminated because of an error. The unit is terminated immediately without waiting for completion of current transactions. Current transactions become subject to rollback[2] during the subsequent restart.

[1] Section *1.7.2 Notes on planned termination, forced termination, and abnormal termination* should be read before restarting.

[2] Current transactions become subject to rollback during a HiRDB restart except in the following cases:

- The database load utility or database reorganization utility was executing in the no-log mode.
- The UAP was executing in the no-log mode.

In such a case, the HiRDB administrator must restore the RDAREAs from backup copies or re-execute the utility after HiRDB has been restarted. For details on how to restore RDAREAs, see *19.2 Recovering a database to the point at which a backup was made*.

(3) Unit startup and termination procedures

To start and terminate a unit:

1. Enter the `pdstart` command to start and run HiRDB:

   ```
   pdstart
   ```

2. Because an error occurred in the unit, enter the `pdstop -u` command to terminate the unit normally (if the unit did not terminate abnormally):

   `pdstop -u` *unit-identifier*

3. After the error has been corrected, enter the `pdstart -u` command to restart the unit:

   `pdstart -u` *unit-identifier*

1.5 Startup and termination of a server (applicable to HiRDB/Parallel Server only)

Executor: HiRDB administrator

When a HiRDB/Parallel Server is being used, each server can be started and terminated independently. This section explains the server startup and termination procedures, which are applicable to the following types of servers:

- Front-end servers
- Dictionary servers
- Back-end servers

(1) Server startup procedure

HiRDB can be started at a server with the command shown in Table 1-7.

Table 1-7: Startup procedure for a server

Startup mode	Execution command	Description of the startup mode	Previous termination mode
Normal startup	`pdstart -s`	This startup mode is used to restart a server after it terminated normally during HiRDB operation. Information that was in effect during the previous session is not inherited.	Normal termination

(2) Server termination procedure

HiRDB can be terminated at a server with the command shown in Table 1-8.

Table 1-8: Termination procedure for a server

Termination mode	Input command	Description of the termination mode
Normal termination	`pdstop -s`	This mode prohibits acceptance of any new transactions and terminates the server after all current transaction processing has been completed.

(3) Server startup and termination procedures

To start and terminate a server:

1. Enter the `pdstart` command to start and run HiRDB:

   ```
   pdstart
   ```

2. Because an error occurred in a server, enter the `pdstop -s` command to

terminate the server normally (if the server did not terminate abnormally):

```
pdstop -s server-name
```

3. After the error has been corrected, enter the `pdstart -s` command to restart the server:

```
pdstart -s server-name
```

1.6 Notes on startup

This section discusses important points to be kept in mind when HiRDB is started. The following topics are covered:

- Notes on HiRDB startup
- Notes on forced startup of HiRDB (or a unit)
- Notes on HiRDB startup processing errors (applicable to HiRDB/Parallel Server only)

1.6.1 Notes on HiRDB startup

(1) Do not stop the pdstart command forcibly

Event

Because there was no response to the entered `pdstart` command, the window from which the `pdstart` command was entered was closed, then HiRDB terminated abnormally.

Cause

If the `pdstart` command is stopped forcibly, HiRDB terminates abnormally because conformity among shared resources is lost. HiRDB also terminates abnormally if the window is closed while waiting for a response from the `pdstart` command.

Action

Do not close the window from which the `pdstart` command is entered until the `pdstart` command has terminated.

The same applies to the other operation commands and utilities. Do not close the window while waiting for a response from a command or utility or while executing a command or utility.

1.6.2 Notes on forced startup of HiRDB (or a unit)

When HiRDB (or a unit) is started forcibly, the information that was in effect during the previous session is not inherited. Therefore, HiRDB cannot restore the database to its status during the previous session. Instead, the HiRDB administrator must restore the database's status.

When HiRDB is started forcibly, all RDAREAs (including system RDAREAs) that were updated during the previous session are corrupted.

The corrupted RDAREAs must be recovered with the database recovery utility before forced startup. HiRDB operations cannot be guaranteed unless the RDAREAs are

recovered.

Procedure

The following is the procedure for recovering corrupted RDAREAs:

1. Check the KFPS01262-I message that was output when HiRDB restart failed.
2. Check in the KFPS01262-I message the group name of the system log's input starting file and check the system log that is created thereafter.*
3. Recover the RDAREAs using the system log checked in step 2 as the input to the database recovery utility (pdrstr command). RDAREAs can be recovered only from the system log.

* The pdlogls -d sys command is used to check the system log.

```
pdlogls -d sys

Group    Type Server   Gen No. Status  Run ID    Block No.
logfg01  sys  sds             1 cnu---u 35108db8        1        2
logfg02  sys  sds             4 cn---cu 3510ecba       10        0
logfg03  sys  sds             1 cn----- 3510ecba        1        3
logfg04  sys  sds             2 cn----- 3510ecba        4        8
logfg05  sys  sds             3 cn----- 3510ecba        9        f
logfg06  sys  sds             0 cn----- 00000000        0        0
```

Explanation

If logfg04 and 5 are displayed in the KFPS01262-I message as the fg name and block number, respectively, it is clear from Run ID, Gen No., and Block No. that the system log files were used in the following order:

logfg04 → logfg05 → logfg02

This means that the system log file to be used has the same Run ID as the file group displayed in the KFPS01262-I message and a generation number greater than displayed under Gen No..

This method is not applicable when the HiRDB/Parallel Server is started and terminated individually at each server.

1.6.3 Notes on HiRDB startup processing errors (applicable to HiRDB/Parallel Server only)

This section provides notes on errors during unit startup. You should consult this section when the following conditions are applicable.

(1) Conditions

All of conditions 1, 2, and 3 or all of conditions 1, 2, and 4 must be satisfied:

1. A HiRDB/Parallel Server is being used.
2. Multiple servers are defined at the unit whose startup processing failed.
3. The startup mode is normal startup or restart after planned termination.
4. The startup mode is restart after forced termination or abnormal termination, and `pd_log_rerun_swap=Y` is specified.

(2) Notes

When a unit is started, HiRDB starts all servers at the unit concurrently. If an invalid system definition operand is detected or an error occurs in one of the servers, startup of the entire unit fails. In such a case, the following events may occur:

- At a server whose startup processing is completed, allocation of the current system log file is completed (the `KFPS01221-I` message is output)
- At a server whose startup processing is not completed, the current system log file is not allocated (the `KFPS01221-I` message is not output)

In such a case, HiRDB allocates a new current file the next time unit startup is performed. The current file allocated during the previous unit startup is closed (it is not reused). The HiRDB administrator must take one of the actions described below in order to reallocate this closed system log file as the current file.

(a) Not reinitializing the system log file

After executing `pdlogunld` or `pdlogchg` for the corresponding system log file, one of the actions listed below can be taken, depending on HiRDB's activity status:

HiRDB's activity status	HiRDB administrator's action
During HiRDB operation	Execute the `pdlogopen` command on the corresponding system log file.
Before normal startup of unit	No action is required. This file is identified as the available current system log file during unit startup.
Before unit restart	After the unit is restarted, execute the `pdlogopen` command on the corresponding system log file.

(b) Reinitializing the system log file

Procedure

1. Execute the `pdlogunld` or `pdlogchg` command on the corresponding system log file:

```
pdlogunld -d sys -s b001 -g syslogfgp03 -o /unld/
```

```
unldlog01
```

2. Use the `pdlogrm` command to delete the corresponding system log file.

   ```
   pdlogrm -d sys -s b001 -f /unt1/sysfile01/log01
   ```

3. Use the `pdloginit` command to initialize the corresponding system log file:

   ```
   pdloginit -d sys -s b001 -f /unt1/sysfile01/log01 -n 5000
   ```

1.7 Notes on termination

This section discusses important points to be kept in mind when HiRDB is terminated. The following topics are covered:

- Notes on HiRDB termination
- Notes on planned termination, forced termination, and abnormal termination

1.7.1 Notes on HiRDB termination

(1) HiRDB cannot be terminated normally if there are uncompleted transactions or connected users

Event

Normal termination of HiRDB failed.

Cause

HiRDB cannot be terminated normally if there are transactions that have not been completed or if there are connected users.

Action

Before terminating HiRDB normally, check for any connected users or transactions that have not been completed:

- The `pdls -d prc` command is used to check for connected users
- The `pdls -d trn` command is used to check transaction status

If no user is connected but there is a transaction processing, the following commands can be entered to complete the transaction:

- Use the `pdcmt` command to commit the transaction
- Use the `pdrbk` command to roll back the transaction

For details on how to terminate HiRDB normally when transactions have not been completed or users are still connected, see *18.14 Actions when there is an undetermined transaction*.

(2) Do not stop the pdstop command forcibly

Event

The window from which the `pdstop` command was entered was closed because there was no response from the command, and HiRDB terminated abnormally.

Cause

If the `pdstop` command is stopped forcibly, HiRDB terminates abnormally

because conformity among shared resources is lost. HiRDB also terminates abnormally if the window is closed while waiting for a response from the `pdstop` command.

Action

Do not close the window from which the `pdstop` command is entered until the `pdstop` command has terminated.

This also applies to the other operation commands and utilities; i.e., do not close the window while waiting for a response from any command or utility or while a command or utility is executing.

(3) Note the shutdown command execution timing

Event

When the `pdstop` and `shutdown` commands were executed consecutively in this order as a shell script, the system server terminated abnormally.

Cause

When the `pdstop` command terminates, the system server has not yet terminated. If the `shutdown` command is executed while the system server is still engaged in termination processing, the system server terminates abnormally.

Action

Do not execute the `shutdown` command immediately after the `pdstop` command. To execute the `shutdown` command after the `pdstop` command, execute the following commands in the order shown:

1. `pdstop`
2. `sleep 60`
3. `shutdown`

If the system server terminates abnormally, the HiRDB startup mode after the next OS boot-up is restart. During a restart, HiRDB restores the system to its status at the time of the OS shutdown, which means that the restart operation may take a long time. For notes on restarting HiRDB, see *1.7.2 Notes on planned termination, forced termination, and abnormal termination*.

1.7.2 Notes on planned termination, forced termination, and abnormal termination

The points discussed below must be taken into consideration when HiRDB planned termination, forced termination, or abnormal termination has occurred.

(1) The HiRDB version must not be changed

The HiRDB version must not be changed when planned termination, forced

termination, or abnormal termination of HiRDB has occurred. If the version is changed, HiRDB restart will fail. HiRDB must be terminated normally before the HiRDB version can be changed.

(2) Only part of the HiRDB system definition can be modified

When planned termination, forced termination, or abnormal termination of HiRDB has occurred, some of the HiRDB system definition operands cannot be modified. For details on the operands that cannot be modified, see the manual *HiRDB Version 8 System Definition*.

HiRDB/Parallel Server

If any unit terminates abnormally during a normal or planned termination, the HiRDB system definitions should not be modified before the next startup. If they are modified, HiRDB startup will probably fail; even if HiRDB startup is successful, HiRDB will not operate normally.

(3) The server configuration cannot be modified (applicable to HiRDB/Parallel Server only)

The HiRDB server configuration should not be modified when planned termination, forced termination, or abnormal termination of HiRDB has occurred. The `pdstart` and `pdunit` operands cannot be modified in the system common definition before the restart. If they are modified, HiRDB restart will fail.

HiRDB must be terminated normally before the HiRDB server configuration can be modified. In such a case, the following files must be reinitialized (otherwise, HiRDB startup may fail):

- All system log files
- All synchronization point dump files
- All unit status files
- All server status files

(4) Addition, deletion, and modification of global buffers

When HiRDB is terminated forcibly or abnormally, the types of global buffer manipulation listed in Table 1-9 are not available.

Table 1-9: Unavailable global buffer manipulations when HiRDB is terminated forcibly or abnormally

Global buffer manipulation	Normal or planned termination	Forced or abnormal termination
Addition of global buffer (addition of `pdbuffer` operand)	Y	NA

Global buffer manipulation	Normal or planned termination	Forced or abnormal termination
Deletion of global buffer (deletion of `pdbuffer` operand)	Y	NA
Modification of global buffer (modification of `pdbuffer` operand)	Y	NA

Y: Available

[NA]: Not available

(5) Global buffer allocated to an RDAREA that was added during HiRDB operation

In the case of a global buffer allocated to an RDAREA that was added during HiRDB operation, whether or not it is inherited at the next startup is determined by the termination mode, as shown in Table 1-10.

Table 1-10: Inheritance of global buffer allocated to RDAREA that was added during HiRDB operation

Condition	Normal or planned termination	Forced or abnormal termination
Whether or not global buffer allocated to a RDAREA that was added during HiRDB operation is inherited.	Not inherited	Inherited

(6) Status files cannot be deleted, modified, or initialized

When planned termination, forced termination, or abnormal termination of HiRDB occurs, the types of status file manipulations listed in Table 1-11 are not available.

Table 1-11: Unavailable status file manipulations in the event of planned, forced, or abnormal termination

Status file manipulation	Normal termination	Planned, forced, or abnormal termination
Addition of status file	Y	Y
Deletion of status file	Y	NA
Modification of status file	Y	NA
Initialization of status file	Y	NA

Y: Available

[NA]: Not available (if executed, restart fails)

After planned, forced, or abnormal termination

The following operands can be added:

- `pd_syssts_file_name_1` to `pd_syssts_file_name_7`
- `pd_sts_file_name_1` to `pd_sts_file_name_7`

These operands cannot be deleted or modified. If deleted or modified, restart of the corresponding unit fails.

(7) Synchronization point dump files cannot be added, deleted, modified, or initialized

When planned termination, forced termination, or abnormal termination of HiRDB occurs, the types of synchronization point dump file manipulations listed in Table 1-12 are not available.

Table 1-12: Unavailable synchronization point dump file manipulations in the event of planned, forced, or abnormal termination

Synchronization point dump file manipulation	Normal termination	Planned, forced, or abnormal termination
Addition of synchronization point dump file	Y	NA
Deletion of synchronization point dump file	Y	NA
Modification of synchronization point dump file	Y	NA
Initialization of synchronization point dump file	Y	NA

Y: Available

NA: Not available (if executed, restart fails)

After planned, forced, or abnormal termination

The following operands cannot be added, deleted, or modified:

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

If these operands are added, deleted, or modified by mistake, restart of the corresponding unit fails.

A synchronization point dump file that was added during the previous HiRDB session will be inherited after a HiRDB restart.

(8) System log files cannot be deleted or modified

When planned termination, forced termination, or abnormal termination of HiRDB occurs, the types of system log file manipulations listed in Table 1-13 are not available.

Table 1-13: Unavailable system log file manipulations in the event of planned, forced, or abnormal termination

System log file manipulation	Normal termination	Planned, forced, or abnormal termination
Addition of system log file	Y	Y
Deletion of system log file	Y	NA
Modification of system log file	Y	NA
Initialization of system log file	Y	Y*

Y: Available.

NA: Not available (if executed, restart fails)

* The system log files listed below must not be initialized; if they are initialized by mistake, restart of the corresponding unit fails (or, in the event restart is successful, the database contents will be corrupted):

- The last system log file used during the previous HiRDB session
- Any system log file that is overwrite-disabled

After planned, forced, or abnormal termination

The following operands can be added:

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

These operands cannot be deleted or modified; if they are deleted or modified, restart of the corresponding unit fails.

(9) Swapping the current system log files

Whether or not swapping of the current system log file occurs during a HiRDB restart depends on the specification of the `pd_log_rerun_swap` operand, as shown in Table 1-14.

Table 1-14: Current system log file swapping conditions

Specification of pd_log_rerun_swap operand	Normal or planned termination	Forced or abnormal termination
`pd_log_rerun_swap=Y`	Y	Y
`pd_log_rerun_swap=N`	Y	NA
`pd_log_rerun_swap` operand omitted	Y	NA

Y: System log file swapping is performed when HiRDB is restarted. The current file in effect during the previous termination is swapped out, and a new current file is allocated.

NA: System log file swapping is not performed when HiRDB is restarted. The current file in effect during the previous termination is retained as the current file. However, the current system log file is swapped out at the HiRDB restart in the following cases:

- An error occurred in the current file in effect during the previous termination
- Unload was executed during HiRDB termination (`pdlogunld` or `pdlogchg` command was executed)

(10) Input error in the system log file during restart

At a HiRDB restart, HiRDB uses the last synchronization point validated during the previous session as the system log input start point. It recovers the database and transactions by reading sequentially all applicable system log files. If at least one system log file was lost due to an error (if dual system log files are used and both versions A and B were lost), restart fails; or, even if restart is successful, the database becomes invalid.

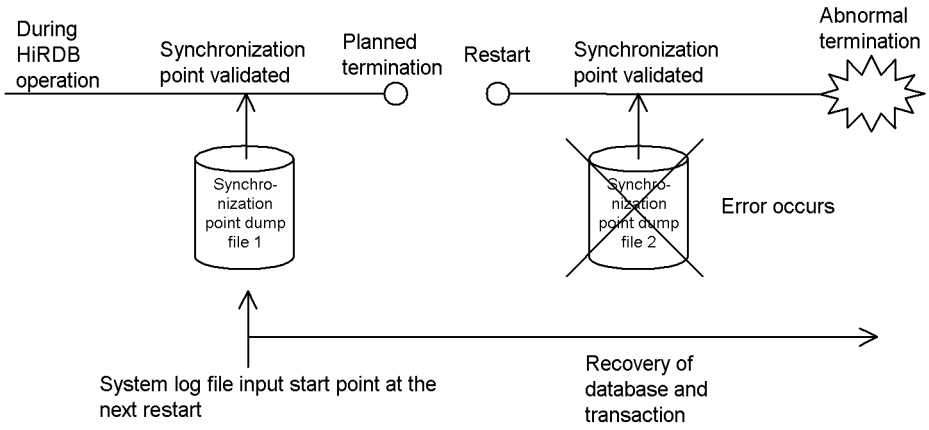
If unload log files have been acquired for those lost system log files, the database can be recovered by the database recovery utility using the backup copy and the unload log files.

If no unload log file is available, system log information cannot be used for database recovery. In such a case, either the database must be restored to the point where the last backup was made and then the same applications must be re-executed, or the database must be re-created.

(11) Notes on restart after planned termination

At a HiRDB restart, HiRDB uses the last synchronization point validated during the previous session as the system log input start point. If an error occurs in the last valid synchronization point dump file or status file, the system log input start point may be moved farther back in time to another valid synchronization point. Figure 1-1 shows an example of moving back the system log input start point.

Figure 1-1: Example of moving back the system log input point



Explanation

If planned termination is executed between the system log input start point determined in this manner and the latest system log file, the following must be noted.

- *If a HiRDB system definition is modified after a planned termination and prior to restart, the restart will fail. In such a case, restart will fail even if the HiRDB system definitions are restored later to their status before the planned termination.*

In such a case, normal startup of the unit must be executed forcibly, and then the database recovery utility (`pdrstr`) must be executed to recover the database from its backup copy and unload log files.

(12) Forced termination may result in abnormal termination of unit

If HiRDB is terminated forcibly (by the `pdstop -f command`), a HiRDB unit may terminate abnormally with abort code `Polkcrt`. This is because a process in critical status was terminated during forced termination processing. However, this event can be ignored, because it does not cause any operational problems. Even a process in critical status terminates immediately during forced termination processing, resulting in this event. However, such a process is restarted the next time the `pdstart` command is entered (the database is recovered from the system log).

1.8 Reducing the HiRDB startup processing time

Executor: HiRDB administrator

Normally, startup processing of resident processes is executed during HiRDB startup processing. As the number of resident processes increases, the HiRDB startup processing time also increases. As a guideline, it takes approximately one second to start one process in a server machine with a performance rating of about 100 MIPS.

When the system switching facility is used, it may be desirable to consider reducing the HiRDB startup processing time.

The number of resident processes is determined by the value of the first parameter in the `pd_process_count` operand.

(1) Procedure for reducing the startup processing time

The second parameter in the `pd_process_count` operand is specified. The following describes the processing depending on whether or not the second parameter is specified.

(a) Second parameter not specified (pd_process_count = 500 specified)

All resident processes are started during HiRDB startup processing. In this case, HiRDB cannot start unless all the resident processes (500 processes in this example) start successfully. If a server machine with a performance rating of approximately 100 MIPS is used in this example, it takes some 500 seconds to start all the resident processes during HiRDB startup processing.

(b) Second parameter is specified (pd_process_count = 500,50 specified)

Some resident processes are started during HiRDB startup processing, and the rest are started after HiRDB startup processing is completed. In this case, HiRDB starts if some of the resident processes (50 processes in this example) start successfully. If a 100-MIPS server machine is used in this example, it takes some 50 seconds to start the resident processes during HiRDB startup processing. The remaining resident processes (450 processes in this example) are started after HiRDB startup processing is completed.

(2) Example

Assume that the following system is used:

- One that requires 200 resident processes
- For OpenTP1's SPP (50 processes), one for which processing is executed concurrently with HiRDB startup processing

System definition

```
pd_process_count=200,50
```

Explanation

- `200` is specified in the first parameter as the total number of resident processes.
- `50` is specified in the second parameter in order to obtain the resident processes for OpenTP1's SPP immediately after HiRDB startup.
- In the case of a HiRDB/Parallel Server, the `pd_process_count` operand is specified in the front-end server definition.

(3) Notes

When this facility is used, the `PDCWAITTIME` operand's value in the client environment definition should be evaluated.

Reason

If more UAPs need to execute immediately after HiRDB startup than the value of the second parameter in `pd_process_count`, their transaction processing will not be executed until after all the remaining resident processes have been started. If the value of the `PDCWAITTIME` operand in the client environment definition is too small, some of the UAPs may result in timeouts.

For details on the `PDCWAITTIME` operand, see the manual *HiRDB Version 8 UAP Development Guide*.

Chapter

2. Security Definition

HiRDB provides security features in order to protect databases from damage and unauthorized use. This chapter describes the security definition procedures.

This chapter contains the following sections:

2.1 About security

HiRDB provides security features in order to protect databases from unauthorized access. The security features are based on the concept of user privileges that prohibit access to a database by a user who does not have the required privilege.

(1) Types of user privileges

Table 2-1 lists the user privileges.

Table 2-1: User privileges

Type of user privilege	Description	What users who have this privilege can do	Who can grant this privilege		
			H	D	S
DBA privilege	This privilege is required in order to grant or revoke the DBA, CONNECT, and schema definition privileges.	• Grant the DBA, CONNECT, and schema definition privileges to other users. • Revoke the DBA, CONNECT, and schema definition privileges of other users. • Define schemas for other users. When a schema is defined, the schema's owner can define base tables, view tables, indexes, abstract data types, foreign tables,[2] foreign indexes,[2] stored procedures, stored functions, and triggers. • Drop other users' schemas, base tables, view tables, indexes, abstract data types, foreign tables,[2] foreign indexes,[2] stored procedures, stored functions, and triggers. • Define user mapping.[2] • Define and change foreign servers.[2] • Define items related to the connection security facility. • Connect to HiRDB (has the CONNECT privilege[1]).	Y	Y	N

Type of user privilege	Description	What users who have this privilege can do	Who can grant this privilege		
			H	D	S
Audit privilege	This privilege must be granted to auditors. Users with this privilege set audit privileges when the security audit facility is being used. For details about the security audit facility, see *22. Using the Security Audit Facility*. Users with the audit privilege have the following privileges: • CONNECT privilege[1] • Schema definition privilege	• Access audit trail tables.[3] • Load data into audit trail tables. • Grant and revoke SELECT privileges for audit trail tables. • Delete audit trail tables. • Modify the passwords of auditors. • Define and delete audit events.	Y	N	N
CONNECT privilege	This privilege is required to use HiRDB. An error results when a user who does not have the CONNECT privilege attempts to use HiRDB.	Connect to databases.	N	Y	N
Schema definition privilege	This privilege is required to define a schema.	• Define your own schema. When a schema is defined, the schema's owner can define base tables, view tables, indexes, abstract data types, foreign tables,[2] foreign indexes,[2] stored procedures, stored functions, and triggers. • Drop the schema owner's own schemas, base tables, view tables, indexes, abstract data types, foreign tables,[2] foreign indexes,[2] stored procedures, stored functions, and triggers.	N	Y	N
RDAREA usage privilege	This privilege is required to use a private RDAREA, but is not needed for creating tables and indexes in public RDAREAs.	Create tables and indexes in a private RDAREA.	Y	Y	N

Type of user privilege	Description	What users who have this privilege can do	Who can grant this privilege		
			H	D	S
Access privilege	This privilege is required to access tables (base tables, view tables, and foreign tables). There are four access privilege types; the types are set at the table level:	Access the tables of other users.	N	N	Y
	SELECT privilege	Search for (SELECT) a table.	N	N	Y
	INSERT privilege	Add (INSERT) row data into a table.	N	N	Y
	DELETE privilege	Delete (DELETE) row data from a table.	N	N	Y
	UPDATE privilege	Update (UPDATE) row data in a table.	N	N	Y

Legend:

H: HiRDB administrator

D: User with DBA privilege

S: Schema owner

Y: Capable of granting privileges

N: Not capable of granting privileges

[1] The CONNECT privilege is not required for use of the directory server linkage facility. For details about the directory server linkage facility, see *24. Using the Directory Server Linkage Facility*.

[2] These operations are applicable when you are using the HiRDB External Data Access facility. For details about the HiRDB External Data Access facility, see the manual *HiRDB External Data Access Version 7 Description and User's Guide*.

[3] You cannot add data to or delete data from an audit trail table (INSERT or UPDATE).

(2) Relationship to a falsification prevented table

The falsification prevention facility is a security function that is provided in addition to the table access privileges. When the falsification prevention option (INSERT ONLY) is specified for a table that is being defined, the defined table becomes a falsification prevented table.

The objectives and features of falsification prevented tables are as follows:

Objectives

- Prevent accidental deletion and updating of data.
- Prevent data from illegal updating and deletion.

Features

- Users with the UPDATE privilege cannot update these tables; even the owners of these tables cannot update them.
- Users with the DELETE privilege cannot delete from these tables data that has not reached the deletion prevention time limit; even the owners of these tables cannot delete such data.
- Users with the INSERT privilege can insert rows into these tables.
- Users with the SELECT privilege can search these tables.

For details about the falsification prevention facility, see the manual *HiRDB Version 8 Installation and Design Guide*.

(3) Relationship to an audit trail table

HiRDB supports a facility that registers the results of security-related checking into an audit trail table as an audit trail when an event that accesses a HiRDB resource occurs. This facility is called the *security audit facility*. An audit trail table records who accessed which resource when, and whether or not the security check was successful. An audit trail table can be used for auditing illegal accesses.

To prevent illegal modification of audit trail tables, the users who are permitted to operate an audit trail table are limited to those shown in Table 2-2.

Table 2-2: Users who are permitted to operate an audit trail table

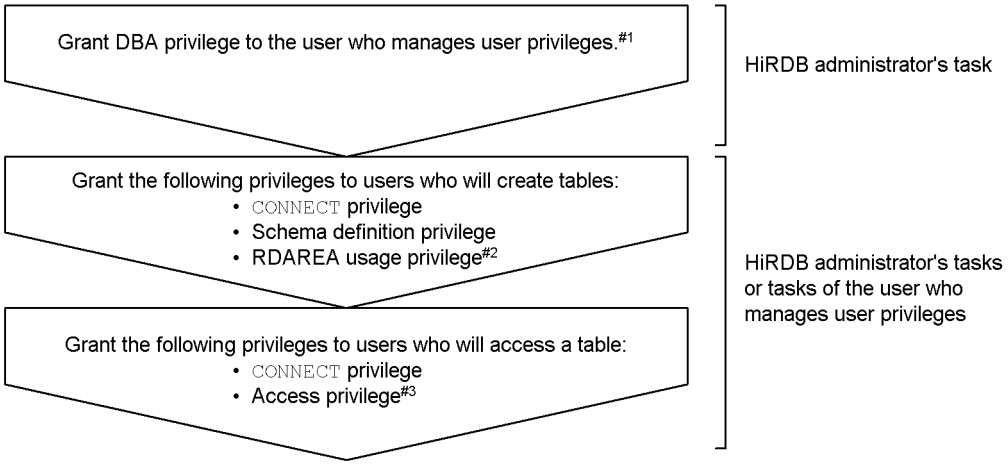
Operation on the audit trail table	Permitted users
Referencing of data (SELECT)	• Auditor • Users who have the SELECT privilege for the audit trail table
Deletion of data (DELETE and PURGE)	Auditor
Addition or modification of data (INSERT or UPDATE)	None

For details about the security audit facility, see *22. Using the Security Audit Facility*.

2.2 Setting user privileges

Figure 2-1 shows the procedure for setting user privileges.

Figure 2-1: Procedure for setting user privileges



#1: This task is required when a user other than a HiRDB administrator manages user privileges.
#2: This privilege is required in order to create tables in a private RDAREA.
#3: Access privilege is required for the table that is to be accessed.

Directory server linkage facility

When the directory server linkage facility is being used, the CONNECT privilege need be granted. For details about the directory server linkage facility, see *24. Using the Directory Server Linkage Facility*.

Setting the audit privilege

For details about setting the audit privilege, see *22.4.3(1) Register the auditor*.

2.2.1 Granting the DBA privilege to users who manage user privileges

Executor: HiRDB administrator

The GRANT statement, which is a definition SQL statement, is used to grant the DBA privilege to a user who will manage user privileges. This task is required when the user who will manage user privileges is not a HiRDB administrator.

Example

Grant the DBA privilege to the user who will manage user privileges (authorization identifier: USER001, password: HIRDB001):

```
GRANT DBA TO USER001 IDENTIFIED BY HIRDB001
```

2.2.2 Granting the CONNECT privilege, schema definition privilege, and RDAREA usage privilege to users who create tables

Executor: HiRDB administrator or user who manages user privileges (user with DBA privilege)

The GRANT statement, which is a definition SQL statement, is used to grant the following privileges to users who will creates tables:

- CONNECT privilege
- Schema definition privilege
- RDAREA usage privilege*

* This privilege is required in order to create tables in private RDAREAs; it is not required in order to create tables in public RDAREAs.

Example

Grant the CONNECT, schema definition, and RDAREA usage privileges (RDAREA name: RDAREA01) to a table creator (authorization identifier: USER002, password: HIRDB002):

```
GRANT CONNECT TO USER002 IDENTIFIED BY HIRDB002
GRANT SCHEMA TO USER002
GRANT RDAREA RDAREA01 TO USER002
```

(1) Granting a schema definition privilege without using the GRANT statement

A schema can be defined for a user to whom the schema definition privilege is to be granted. The schema definition privilege is then granted to the user that defined the schema.

(2) Granting an RDAREA usage privilege without using the GRANT statement

When defining a private RDAREA,* the HiRDB administrator can grant the RDAREA usage privilege to that private RDAREA.

* The following utilities and control statements are used:

- Database initialization utility's CREATE RDAREA statement with the USER USED BY operand specified
- Database structure modification utility's CREATE RDAREA statement with the USER USED BY operand specified

(3) Changing a private RDAREA to a public RDAREA

After revoking RDAREA usage privileges to a private RDAREA, use the GRANT statement, which is a definition SQL statement, to change the RDAREA. For details about revoking RDAREA usage privileges, see *2.3(4) Revoking RDAREA usage privileges*.

Example

Redefine the private RDAREA named RDAREA02 as a public RDAREA:

```
GRANT RDAREA RDAREA02 TO PUBLIC
```

2.2.3 Granting CONNECT and access privileges to users who access tables (database)

(1) Granting CONNECT privileges

Executor: HiRDB administrator or user who manages user privileges (user with DBA privilege)

The GRANT statement, which is a definition SQL statement, is used to grant the CONNECT privilege to users who will access database tables.

Example

Grant the CONNECT privilege to a user (authorization identifier USER003, password HIRDB003) who will access tables:

```
GRANT CONNECT TO USER003 IDENTIFIED BY HIRDB003
```

(2) Granting access privileges

Executor: Table owner

The GRANT statement, which is a definition SQL statement, is used by a table's owner to grant access privileges to users who will access the table.

Example 1

Grant only the SELECT privilege for a table (authorization-identifier.table-identifier USER002.T001) to a user (authorization identifier USER004) who is to be permitted only to make retrievals from the table:

```
GRANT SELECT ON USER002.T001 TO USER004
```

Example 2

Grant the SELECT and UPDATE privileges for a table (authorization-identifier.table-identifier USER002.T001) to a user (authorization identifier USER005) who is to be permitted to retrieve and update the table:

```
GRANT SELECT,UPDATE ON USER002.T001 TO USER005
```

Example 3

Grant the SELECT, UPDATE, and INSERT privileges for a table (authorization-identifier.table-identifier USER002.T001) to a user (authorization identifier USER006) who is to be permitted to retrieve, update, add to, and delete the table:

```
GRANT ALL ON USER002.T001 TO USER006
```

Notes

1. A table owner can grant to another user only the access privileges to that table that the owner has been granted. Table 2-3 lists the access privileges granted to a table owner.
2. A table owner cannot grant to another user access privileges to a view table that was defined from a table that belongs to a different user.

Table 2-3: Table owner's access privileges

Type of table	Table owner's access privileges	Can provide access privileges?
Base table	All access privileges	Yes
Foreign table	All access privileges	Yes
Read-only view table defined from a base table or foreign table that belongs to the user[1]	SELECT privileges	Yes
Updatable view table defined from a base table or foreign table that belongs to the user[2]	All access privileges	Yes
Read-only view table defined from a base table or foreign table that belongs to another user[1,3]	SELECT privileges	No
Updatable view table defined from a base table or foreign table that belongs to another user[2,3]	All access privileges the user has for base tables or foreign tables	No

Legend:

Yes: The user can grant and revoke the access privileges of other users.

No: The user cannot grant or revoke the access privileges of other users.

Note

You can grant to other users only those access privileges to a foreign table that you have for the base table of that foreign table (table at a foreign server). If you attempt to grant access privileges you do not have, an error results.

[1] A read-only view table is a view table for which any of the following information is

specified in the view definition:

- `DISTINCT`, set function, literal, or arithmetic operation is specified in the `SELECT` clause.
- The same base table column or foreign table column is specified in the `SELECT` clause more than once.
- Tables are joined.
- `GROUP BY` clause is specified.
- `HAVING` clause is specified.
- `READ ONLY` clause is specified.

[2] An updatable view table is a view table to which the read-only attribute is not assigned.

[3] To define a view table from a table that belongs to another user, you must have `SELECT` privileges for that table.

2.3 Revoking user privileges

(1) Revoking DBA privileges

Executor: HiRDB administrator

The REVOKE statement, which is a definition SQL statement, is used to revoke the DBA privilege.

Example

Revoke the DBA privilege of the user who manages user privileges (authorization identifier: USER001):

```
REVOKE DBA FROM USER001
```

(2) Revoking CONNECT privileges

Executor: HiRDB administrator or user who manages user privileges (user with DBA privilege)

The REVOKE statement, which is a definition SQL statement, is used to revoke the CONNECT privilege.

Example

Revoke the CONNECT privilege of the user whose authorization identifier is USER003:

```
REVOKE CONNECT FROM USER003
```

(3) Revoking schema definition privileges

Executor: HiRDB administrator or user who manages user privileges (user with DBA privilege)

The REVOKE statement, which is a definition SQL statement, is used to revoke the schema definition privilege.

Example

Revoke the schema definition privilege of the user whose authorization identifier is USER002:

```
REVOKE SCHEMA FROM USER002
```

Note

The schema definition privilege of a user who has defined a schema cannot be revoked. When a schema definition privilege is to be revoked, a check should be made to ensure that no schema has been defined by that user.

(4) Revoking RDAREA usage privileges

Executor: HiRDB administrator or user who manages user privileges (user with DBA privilege)

The REVOKE statement, which is a definition SQL statement, is used to revoke the RDAREA usage privilege.

Example

Revoke the RDAREA usage privilege for the RDAREA named RDAREA01 of the user whose authorization identifier is USER002:

```
REVOKE RDAREA RDAREA01 FROM USER002
```

Note

The RDAREA usage privilege of a user who has defined a table or index in the specified RDAREA cannot be revoked. When an RDAREA usage privilege is to be revoked, a check should be made to ensure that no table or index has been defined in the RDAREA by that user.

(5) Revoking access privileges

Executor: Table owner

The REVOKE statement, which is a definition SQL statement, is used to revoke access privileges.

Example

Revoke the DELETE privilege for a table (authorization-identifier.table-identifier USER002.T001) that was granted to the user with authorization identifier USER004:

```
REVOKE UPDATE ON USER002.T001 TO USER004
```

Note

When the SELECT table access privilege is revoked, all the user's view tables defined from that table are deleted.

2.4 Setting a referencing privilege for data dictionary tables

Executor: HiRDB administrator

To enhance system security, you can set a reference privilege for data dictionary tables in order to restrict access to the data dictionary tables. You set the reference privilege for data dictionary tables by specifying `limited` in the following utility control statements:

- Database initialization utility: `dicinf` operand of define system statement
- Database structure modification utility: `dicinf` operand of alter system statement

Table 2-4 shows the relationship between the `dicinf` operand value and the data dictionary tables that can be referenced.

Table 2-4: Relationship between the dicinf operand value and the data dictionary tables that can be referenced

Data dictionary table	dicinf operand value			
	limited			unlimited
	DBA privilege holders	Auditor	General users	
SQL_PHYSICAL_FILES	All	All	Table information about HiRDB files that comprise RDAREAs for which the general user has usage privileges.	All
SQL_RDAREAS	All	All	Table information about RDAREAs for which the general user has usage privileges.	All
SQL_TABLES	All[1]	All[1]	Table information about tables for which the general user has access privileges.	All
SQL_COLUMNS	All[1]	All[1]	Table information about columns in tables for which the general user has access privileges.	All
SQL_INDEXES	All[1]	All[1]	Table information about indexes for tables for which the general user has access privileges.	All
SQL_USERS	All	All	None	None

Data dictionary table	dicinf operand value			
	limited			unlimited
	DBA privilege holders	Auditor	General users	
SQL_RDAREA_PRIVILEGES	All	All	Table information about RDAREAs for which the general user has access privileges.	All
SQL_TABLE_PRIVILEGES	All[1]	All[1]	Table information about tables for which the general user has access privileges.	All
SQL_DIV_TABLE	All[1]	All[1]		All
SQL_INDEX_COLINF	All[1]	All[1]	Table information about indexes for tables for which the general user has access privileges.	All
SQL_TABLE_STATISTICS	All	All	Table statistical information about tables for which the general user has access privileges.	All
SQL_COLUMN_STATISTICS	All	All	Column statistical information about tables for which the general user has access privileges.	All
SQL_INDEX_STATISTICS	All	All	Index statistical information about tables for which the general user has access privileges.	All
SQL_VIEW_TABLE_USAGE	All	All	Table information about view tables for which the general user has access privileges.	All
SQL_VIEWS	All	All		All
SQL_DIV_INDEX	All[1]	All[1]	Table information about indexes for tables for which the general user has access privileges.	All
SQL_DIV_COLUMN	All[2]	All[2]	Table information about tables for which the general user has access privileges.	All
SQL_REFERENTIAL_CONSTRAINTS	All	All	Constraint information about tables for which the general user has access privileges.	All
SQL_ALIASES	All	All	All	All
SQL_ROUTINES	All	All	All	All
SQL_ROUTINE_RESOURCES	All	All	All	All

Data dictionary table	dicinf operand value			
	limited			unlimited
	DBA privilege holders	Auditor	General users	
SQL_ROUTINE_PARAMS	All	All	All	All
SQL_DATATYPES	All	All	All	All
SQL_DATATYPE_DESCRIPTORS	All	All	All	All
SQL_TABLE_RESOURCES	All	All	All	All
SQL_PLUGINS	All	All	All	All
SQL_PLUGIN_ROUTINES	All	All	All	All
SQL_PLUGIN_ROUTINE_PARAMS	All	All	All	All
SQL_REGISTRY	All	All	All	All
SQL_INDEX_TYPES	All	All	All	All
SQL_INDEX_DATATYPE	All	All	All	All
SQL_INDEX_FUNCTION	All	All	All	All
SQL_TYPE_RESOURCES	All	All	All	All
SQL_INDEX_RESOURCES	All	All	All	All
SQL_INDEX_TYPE_FUNCTION	All	All	All	All
SQL_EXCEPT	All[1]	All[1]	Table information about indexes for tables for which the general user has access privileges.	All
SQL_FOREIGN_SERVERS	All	All	Table information about foreign servers that can be accessed.	All
SQL_USER_MAPPINGS	All	All	Table mapping information given to the general user.	All
SQL_IOS_GENERATIONS	All	All	All	All
SQL_PARTKEY	All	All	Table information about tables for which the general user has access privileges.	All

Data dictionary table	dicinf operand value			
	limited			unlimited
	DBA privilege holders	Auditor	General users	
SQL_PARTKEY_DIVISION	All	All	Table information about tables for which the general user has access privileges.	All
SQL_TRIGGERS	All	All	Table information about triggers defined by the general user.	All
SQL_TRIGGER_COLUMNS	All	All		All
SQL_TRIGGER_DEF_SOURCE	All	All		All
SQL_TRIGGER_USAGE	All	All		All
SQL_AUDITS	None	All	None	None
SQL_KEYCOLUMN_USAGE	All	All	Table information about tables for which the general user has access privileges.	All
SQL_TABLE_CONSTRAINTS	All	All		All
SQL_CHECKS	All	All		All
SQL_CHECK_COLUMNS	All	All		All
SQL_DIV_TYPE	All	All		All
SQL_SYSPARAMS	All	All	None	None

All: All columns can be referenced.

None: No columns can be referenced.

[1] The base table of the data dictionary table cannot be referenced.

[2] The base table of the data dictionary table can also be referenced.

Chapter

3. Handling System Log Files

This chapter describes the procedures for handling the system log files.

This chapter contains the following sections:

3.1 Basics

This section provides an overview of the system log files. The user must understand system log files before actually handling them.

(1) System log is used for error recovery and tuning information

Information about the updates made to the database (system log information) is stored in a system log file. This system log information is used for the following purposes:

- If HiRDB or a UAP terminates abnormally, HiRDB uses the system log to recover the database and transactions.
- If an error occurs in the database, the HiRDB administrator uses the system log in conjunction with the database recovery utility to recover the database. The system log provides the input to the database recovery utility.
- When the HiRDB administrator performs system tuning, the system log is used as the input information for this tuning (input to the statistics analysis utility).

(2) HiRDB manages the system log files by status

HiRDB manages system log files according to the statuses listed in Table 3-1.

Table 3-1: System log file statuses

Status	Description	
Current	File to which the system log is being output. There is always only one file in this status.	
Standby (swappable)	File that is not currently the target file for output of the system log, but is available to be swapped in as the current file when any of the following occurs: ● The existing current file reaches its capacity ● An error occurs in the existing current file ● The `pdlogswap` command is entered All of the following conditions must be applicable to a file in this status.	
	Overwrite enabled	Status of a file that does not include the system log required for full system recovery (system log corresponding to the synchronization point dumps for the number of guaranteed-valid generations).
	Unload completed	Status of a file whose acquired system log has been unloaded to an unload log file.
	Extraction completed state (HiRDB Datareplicator)	Status of a file for which the HiRDB Datareplicator on the source machine has completely read the system log. This status is applicable only when the HiRDB Datareplicator is being used. With HiRDB, the extraction status is checked when a synchronization point dump is acquired. In other words, the extraction completed status results when the HiRDB Datareplicator on the source machine has completely read the system log and synchronization points have been generated.
	Overwriting permitted status for online reorganization (HiRDB Staticizer Option)	Status of a file that does not include the system log required for update processing. This status is applicable when updatable online reorganization is being performed.
Standby (unswappable)	File to which the system log will not be output (file that will not become the current file). The current file is in post-swapping status. One of the following conditions must be satisfied for this status to result.	
	Overwrite disabled state	Status of a file that includes the system log required for full system recovery (system log corresponding to the synchronization point dumps for the number of guaranteed-valid generations).
	Unload wait state	Status of a file whose acquired system log has not been unloaded to the unload log file.
	Extracting status (HiRDB Datareplicator)	Status of a file for which the HiRDB Datareplicator on the source machine has not completely read the system log (unread data remains). This status is applicable only when the HiRDB Datareplicator is being used.

Status	Description	
	Overwriting denied status for online reorganization (HiRDB Staticizer Option)	Status of a file that includes the system log required for update processing. This status is applicable when updatable online reorganization is being performed.
Reserved	File to which the system log will not be output.	

(3) HiRDB administrator's task

System log information is output during HiRDB operation. When one system log file is filled, the output destination is changed to another file whose status makes it a swappable target. When this happens, the current file is placed in unswappable target status and the selected swappable target file becomes the current file. This is called system log file swapping. Thus, the status of system log files changes during HiRDB operation.

The HiRDB administrator must operate the system log files so that a swappable file always exists. If file swapping is needed, but there are no swappable files available, HiRDB (the unit for a HiRDB/Parallel Server) terminates abnormally.

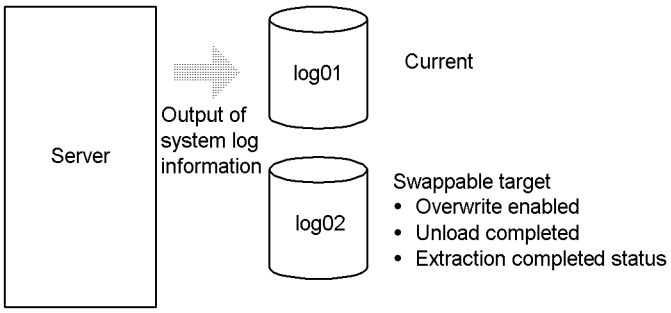
HiRDB provides a function for monitoring the free area for system log files. For details about this function, see *3.9 Monitoring the free area for system log files*.

(4) Status changes of system log files

When HiRDB is operating, the status of the system log files changes as described below. It is assumed here that updatable online reorganization of the HiRDB Staticizer Option is not being used, which means that system log files will not be in overwriting permitted status for online organization or overwriting denied status for online reorganization. For details about the statuses of system log files when updatable online reorganization is being used, see the *HiRDB Staticizer Option Version 7 Description and User's Guide*.

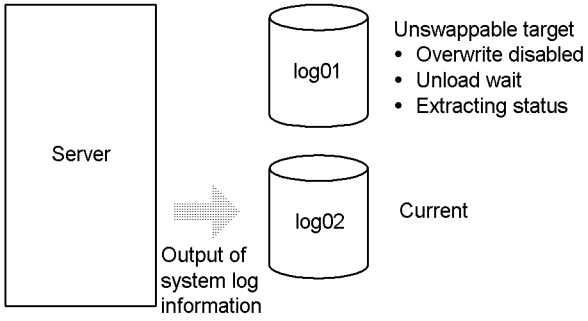
(a) HiRDB is started normally

When HiRDB is started normally, all system log files for which `ONL` is specified with the `pdlogadfg -d sys` operand are opened. Of the opened files, the first file that was specified in the `pdlogadfg` operand becomes the current file, and the other opened files are placed in swappable target status. The files that did not open because of an opening error and files for which `ONL` was not specified are placed in reserved status. When HiRDB is restarted, the current file that was in effect during the previous session is inherited.



(b) File status changes when system log files are swapped

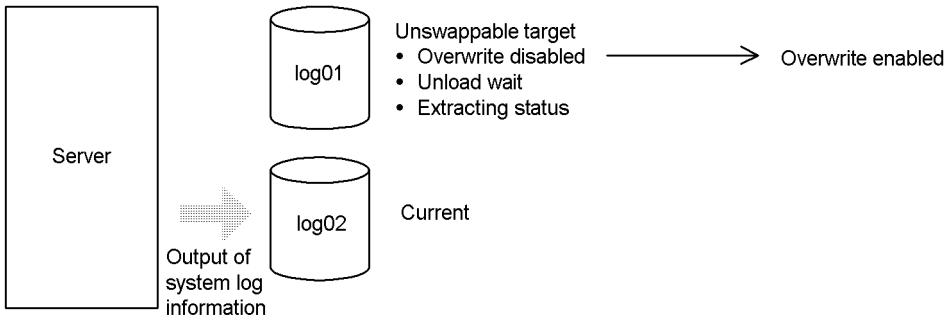
When the current file becomes full, the output destination changes to a file in swappable target status; i.e., system log files are swapped. When this happens, file status changes as follows:



(c) File status changes when a synchronization point dump is validated

When system log files are swapped, HiRDB executes synchronization point dump validation processing. Once the synchronization point dumps have been validated, the system log information collected prior to this validation processing is no longer needed for a HiRDB restart. When none of the system log information stored in the file is needed, the file status changes from overwrite disabled to overwrite enabled.

If a transaction is executing during synchronization point dump validation processing, the synchronization point dump will not be validated until the transaction has terminated.

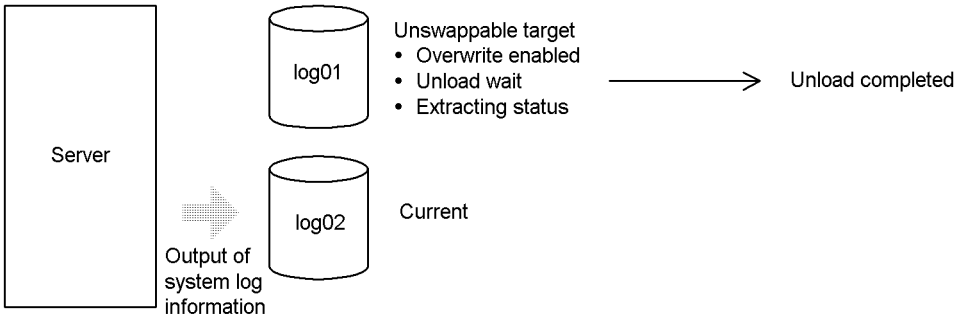


(d) File status changes when system log information is unloaded or a system log file is released

The HiRDB administrator executes the operations described here.

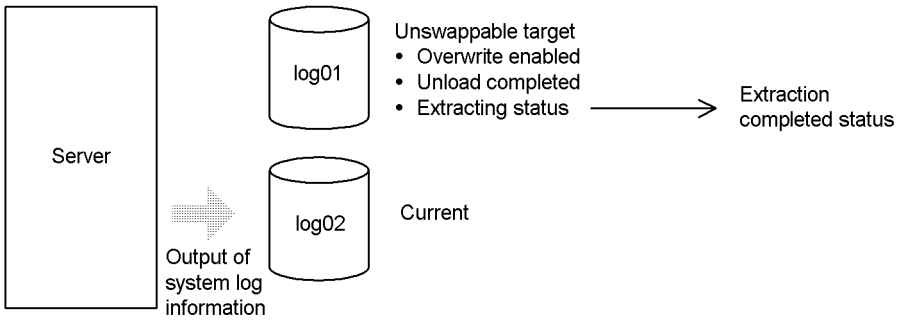
When the `pdlogunld` command is executed to unload the system log information contained in a file that is in unload wait status, the file status changes from unload wait to unload completed. The system log information that is unloaded here can be used if it is necessary to recover the database.

If the system is being operated without unloading system log information and the `pdlogchg -z` command is executed to release a system log file, the file's status changes from unload wait to unload completed.



(e) File status changes when HiRDB Datareplicator at the extracted side completes extraction of system log information

When HiRDB Datareplicator at the extracted side completes extraction of system log information, the file's status changes from extracting status to extraction completed status.



Note

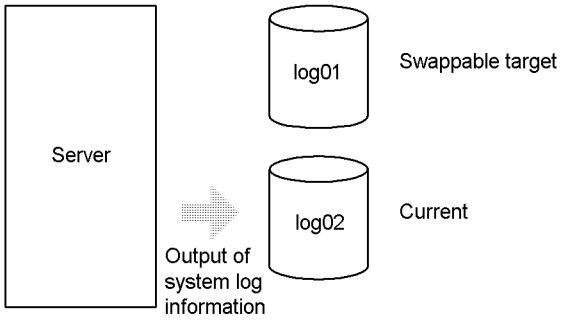
HiRDB checks the extraction status each time it acquires a synchronization point dump. A system log file is placed in extraction completed status at the next synchronization point after the system log has been read in its entirety by HiRDB Datareplicator at the extracted side.

(f) File is then placed in swappable target status

Because the file is now in the following statuses, it changes from being an unswappable target to a swappable target:

- Overwrite enabled
- Unload completed
- Extraction completed status

The HiRDB administrator must handle the system log files in such a manner that there is always an available swappable target.



(5) Selecting a system log file handling method

There are different methods for handling the system log files, as shown in Table 3-2. The HiRDB administrator must select one of these methods.

Table 3-2: System log file handling methods

Operation	Handling method	Database recovery procedure
System log unloading*	Any file in unload wait status must be unloaded. For details, see *3.2 Unloading the system log*.	The database is recovered from a backup and unload log files. The database can be restored to the point at which the backup was made or to any synchronization point since the backup was made.
Without unloading the system log	Any file in unload wait status is released (there is no need to unload it). Backups must be made at each server. For details, see *3.3 Operating without unloading the system log*.	The database is recovered from a backup and the system log information acquired since the backup was made. The database can be restored to the point at which the backup was made or to any synchronization point since the backup was made.
Releasing check of unload status	Because there are no files in unload wait status, there is no need to unload system log files. For details, see *3.4 Releasing checking of unload status*.	The database is recovered from a backup. The database can be restored only to the point at which the backup was made.

* If the system switchover facility is being used, create an unload log file on a shared disk (character special file).

When an unload log file is created as a regular file, it must be shared by the primary and secondary systems. Therefore, in the case of HP-UX, JFS must be installed.

(6) When an error occurs in a system log file

For details on how to handle errors in system log files, see *18.6 Handling of system log file errors*.

Frequently asked questions concerning system log file error handling procedures are answered in Q&A format in *A.1 System log files*.

(7) Commands used to manipulate system log files

Table 3-3 lists the commands that are provided for manipulating system log files. These commands are used by the HiRDB administrator.

Table 3-3: Commands used to manipulate system log files

Command	Description
`pdloginit`	Initializes a system log file.
`pdlogls`	Displays information about a system log file

Command	Description
pdlogunld	• Unloads the contents of a system log file into an unload log file • Changes the system log file's status from unload wait to unload completed
pdlogchg	Forcibly places a system log file in the following statuses: • Unload completed • Extraction completed status
pdlogswap	Swaps system log files; places the current file in unswappable target status.
pdlogopen	Opens a system log file. Places a reserved file in overwrite enabled status.
pdlogcls	Closes a system log file. Places an overwrite enabled file in reserved status.
pdlogrm	Deletes a system log file.

3.2 Unloading the system log

Executor: HiRDB administrator

The `pdlogunld` command is used to unload into an unload log file the information needed for database recovery. In the event of an error in the database, the database recovery utility uses this unload log file to recover the database.

The HiRDB administrator must ensure than any file in unload wait status is unloaded.

Automatic log unloading facility

Normally, an unload log file is created by using the `pdlogunld` command to unload the system log files. This process can be automated by using the automatic log unloading facility. For details on the automatic log unloading facility, see *3.8 Using the automatic log unloading facility*.

Note

If the system switchover facility is being used, create an unload log file on a shared disk (character special file).

When an unload log file is created as a regular file, it must be shared by the primary and secondary systems. Therefore, in the case of HP-UX, JFS must be installed.

3.2.1 HiRDB/Single Server

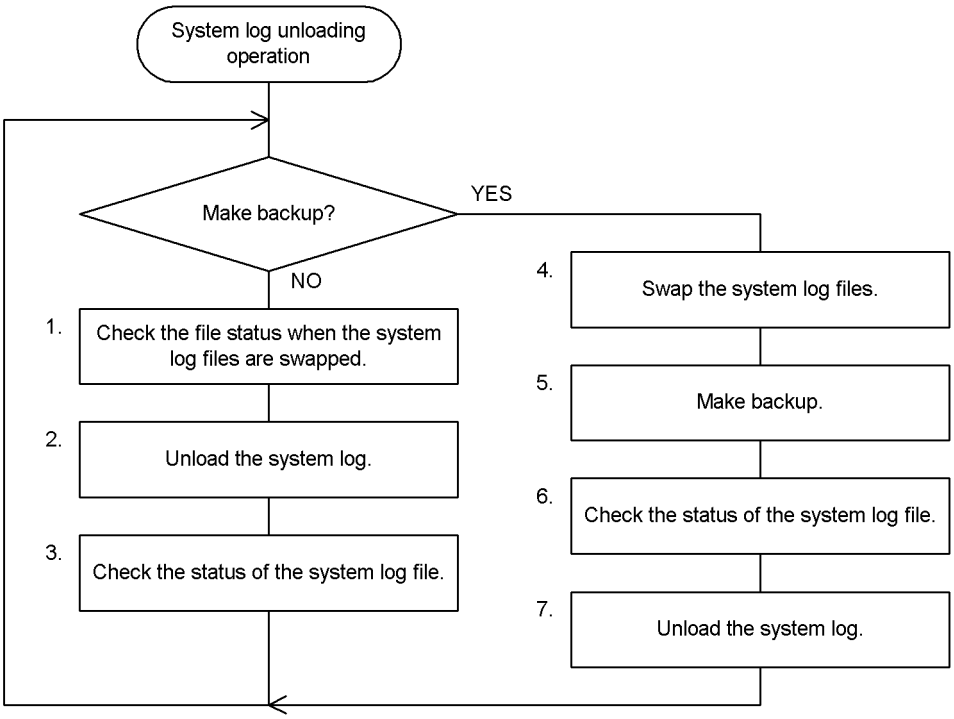
Database recovery in the event of an error

If an error occurs in the database in the operating environment in which system logs are unloaded, the database is recovered from a backup copy and the unload log files that contain system log information collected since the backup was made. For details on the database recovery procedure, see *19. Database Recovery Procedures*.

Example

Figure 3-1 shows the procedure for unloading a system log at a HiRDB/Single Server.

Figure 3-1: Procedure for unloading a system log (HiRDB/Single Server)



Notes

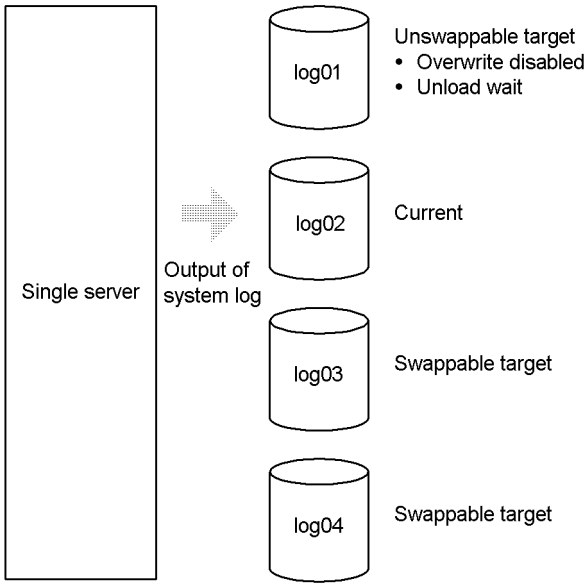
[1] The numbers to the left of the process boxes correspond to the paragraph numbers of the explanations on the following pages. For example, step 5 is explained in paragraph (5) below.

[2] Perform Steps 1 to 3 each time system log file swapping occurs.

(1) Check the file status when the system log files are swapped

When a system log file becomes full, it is swapped with another file. When system log file swapping occurs, the `KFPS01221-I` and `KFPS01222-I` messages are output to the message log file and the syslogfile. The `pdlogls` command can be used to check the status of the system log file:

```
pdlogls -d sys
```



Explanation

Because the `log01` file has been filled with the system log, the system log output destination changes from `log01` to `log02` (the system log files are swapped). As a result, `log01` is placed in the following statuses:

- Overwrite disabled
- Unload wait

(2) Unload the system log

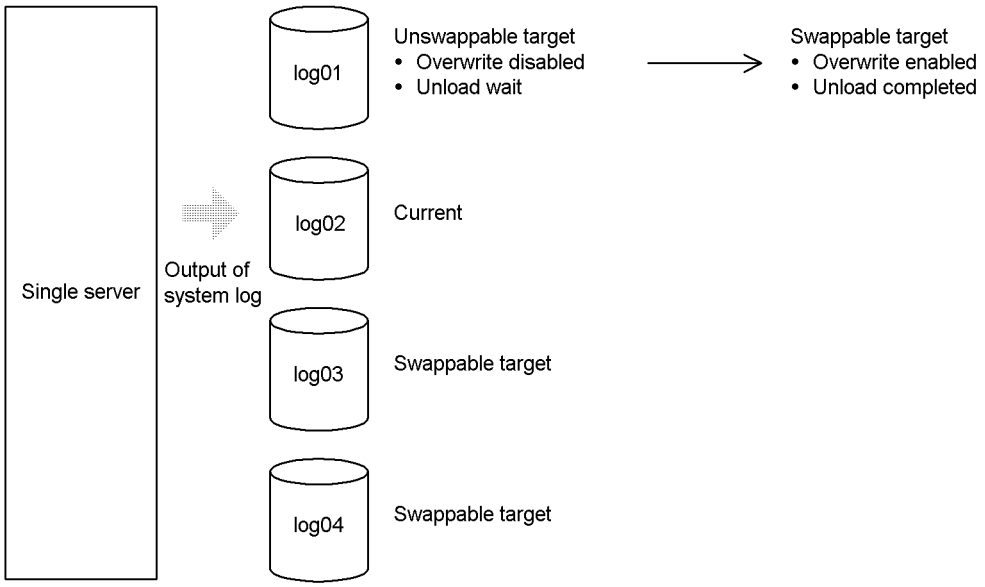
Use the `pdlogunld` command to unload the file (`log01`) that is in unload wait status.

```
pdlogunld -d sys -g log01 -o /unld/unldlog01
```

(3) Check the status of the system log file

The `pdlogls` command is used to check the status of the system log file (`log01`):

```
pdlogls -d sys
```



Explanation

- Once the system log has been unloaded, the file status changes from unload wait to unload completed.
- When the system log files are swapped, synchronization point dump validation processing is executed. When the synchronization point dumps have been validated, the file status changes from overwrite disabled to overwrite enabled.
- As a result, the file status changes from unswappable target to swappable target.

Important

If system log file swapping occurs when there is no file in swappable target status, the HiRDB/Single Server terminates abnormally. For this reason, the HiRDB administrator must ensure that there is always available a file in swappable target status. When there are no files in swappable target status, HiRDB outputs the `KFPS01224-I` message to the message log file and to the syslogfile.

(4) Swap the system log files

Before a backup copy is made, the `pdlogswap` command is used to swap the system log files. System log files are swapped in order to physically separate the system logs needed for database restoration. The system log files storing the system log information needed for database restoration are those that become primary from this point on.

```
pdlogswap -d sys -w
```

(5) Make a backup

The pdcopy command (database copy utility) is used to back up all RDAREAs:

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode (r or s).

-a: Specifies that all RDAREAs are to be backed up.

-b: Specifies the name of the backup file.

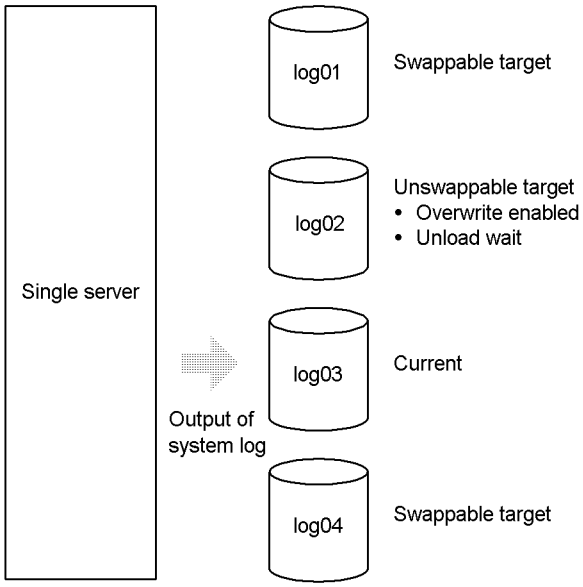
Important

If the backup copy made in this step is used subsequently to recover the RDAREAs, the unload log files containing system log information subsequent to this current file are used as input to the database recovery utility.

(6) Check the status of the system log file

The pdlogls command is used to check the status of the system log file:

```
pdlogls -d sys
```



(7) Unload the system log

Use the `pdlogunld` command to unload the file (`log02`) that is in unload wait status.

```
pdlogunld -d sys -g log02 -o /unld/unldlog02
```

3.2.2 HiRDB/Parallel Server

Database recovery in the event of an error

If a database error occurs in the operating environment in which system logs are unloaded, the database is recovered from a backup copy and the unload log files that contain system log information collected since the backup was made. For details on the database recovery procedure, see *19. Database Recovery Procedures*.

System log for a front-end server

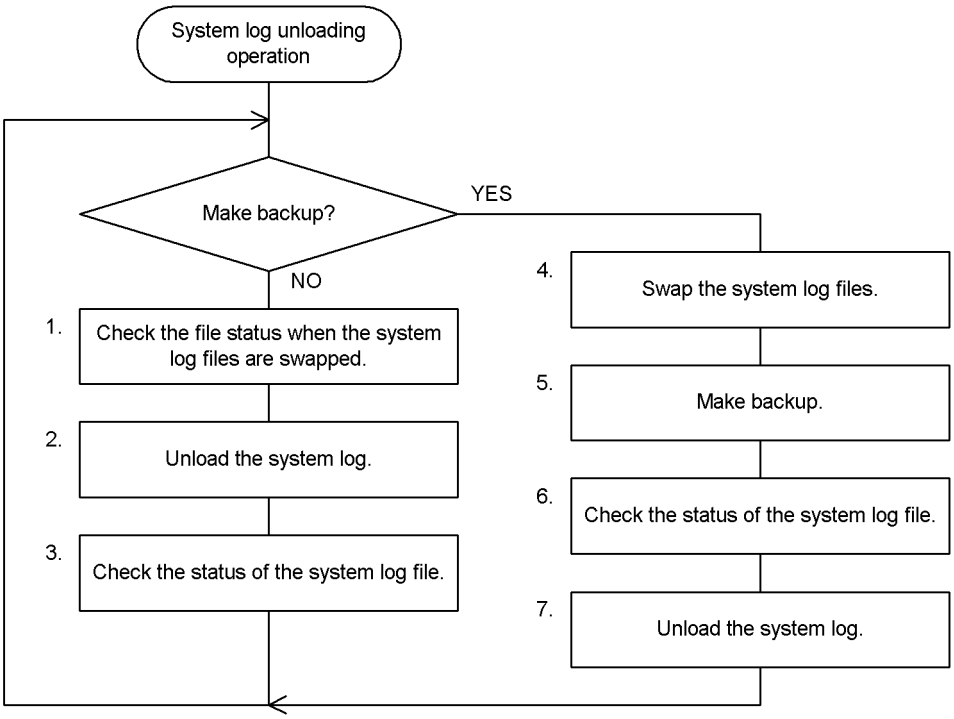
System log information for a front-end server need not be unloaded because it is not required for database recovery. The specification `pd_log_unload_check=N` in the front-end server definition releases checking of system log file unload status for the front-end server. This eliminates the need to unload the front-end server's system log file. When `pd_log_unload_check=N` is not specified, one of the following actions must be taken for a file in unload wait status at the front-end server:

- Execute the `pdlogunld` command to unload the system log information and place the file in unload completed status
- Execute the `pdlogchg` command to forcibly change the file's status to unload completed

Example

Figure 3-2 shows the procedure for unloading a system log at a HiRDB/Parallel Server.

Figure 3-2: Procedure for unloading a system log (HiRDB/Parallel Server)



Notes

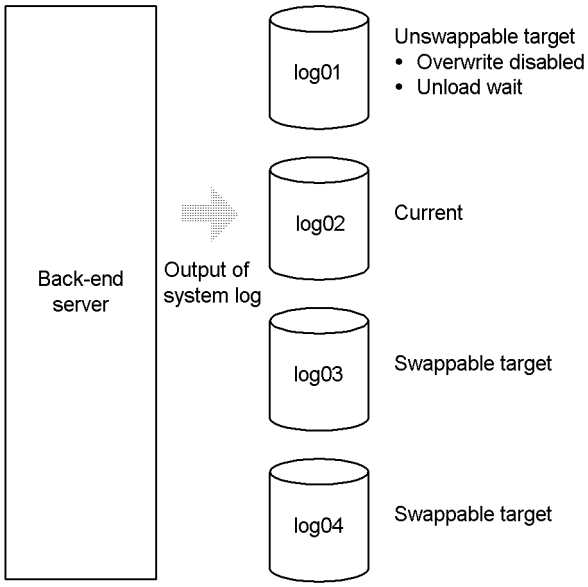
[1] The numbers to the left of the process boxes correspond to the paragraph numbers of the explanations on the following pages. For example, step 5 is explained in paragraph (5) below.

[2] Perform Steps 1 to 3 each time system log file swapping occurs.

(1) Check the file status when the system log files are swapped

When a system log file becomes full, it is swapped with another file. When system log file swapping occurs, the `KFPS01221-I` and `KFPS01222-I` messages are output to the message log file and the syslogfile. The `pdlogls` command can be used to check the status of the system log file:

```
pdlogls -d sys -s bes1
```



Explanation

Because the `log01` file has been filled with the system log, the system log output destination changes from `log01` to `log02` (the system log files are swapped). As a result, `log01` is placed in the following statuses:

- Overwrite disabled
- Unload wait

(2) Unload the system log

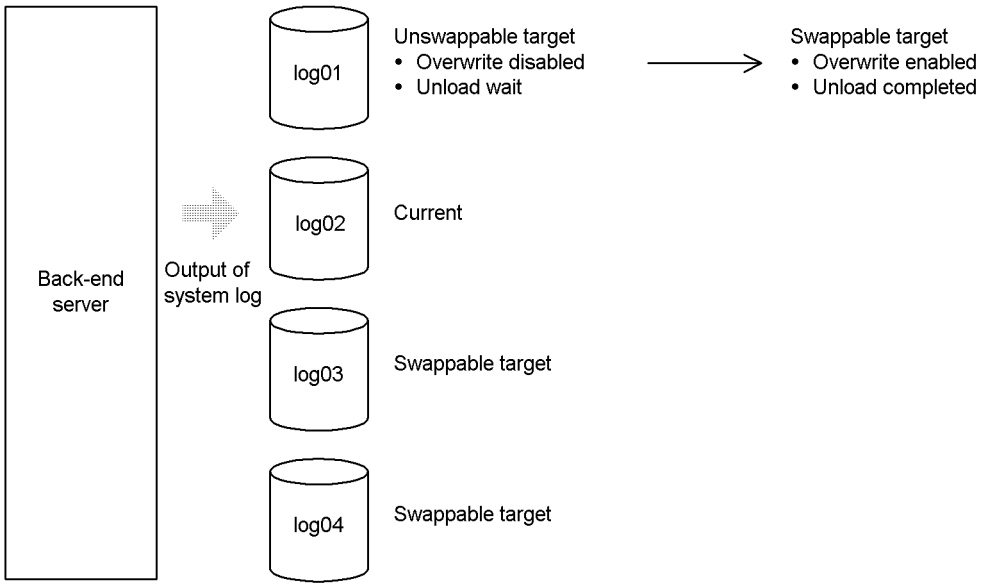
Use the `pdlogunld` command to unload the file (`log01`) that is in unload wait status.

```
pdlogunld -d sys -s bes1 -g log01 -o /unld/unldlog01
```

(3) Check the status of the system log file

The `pdlogls` command is used to check the status of the system log file (`log01`):

```
pdlogls -d sys -s bes1
```



Explanation

- Once the system log has been unloaded, the file status changes from unload wait to unload completed.
- When the system log files are swapped, synchronization point dump validation processing is executed. When the synchronization point dumps have been validated, the file status changes from overwrite disabled to overwrite enabled.
- As a result, the file status changes from unswappable target to swappable target.

Important

If system log file swapping occurs when there is no file in swappable target status, the unit terminates abnormally. For this reason, the HiRDB administrator must ensure that there is always available a file in swappable target status. When there are no files in swappable target status, HiRDB outputs the `KFPS01224-I` message to the message log file and to the syslogfile.

(4) Swap the system log files

Before a backup copy is made, the `pdlogswap` command is used to swap the system log files; in this example, the system log file for a back-end server (`bes1`) is swapped because the RDAREAs in `bes1` are to be backed up.

System log files are swapped in order to physically separate the system logs needed for

database restoration. The system log files storing the system log information needed for database restoration are those that become primary from this point on.

```
pdlogswap -d sys -s bes1 -w
```

(5) Make a backup

The pdcopy command (database copy utility) is used to back up in units of servers:

```
pdcopy -m /rdarea/mast/mast01 -M r -s bes1 -b /pdcopy/backup01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode (r or s).

-s: Specifies that all RDAREAs on back-end server bes1 are to be backed up.

-b: Specifies the name of the backup file.

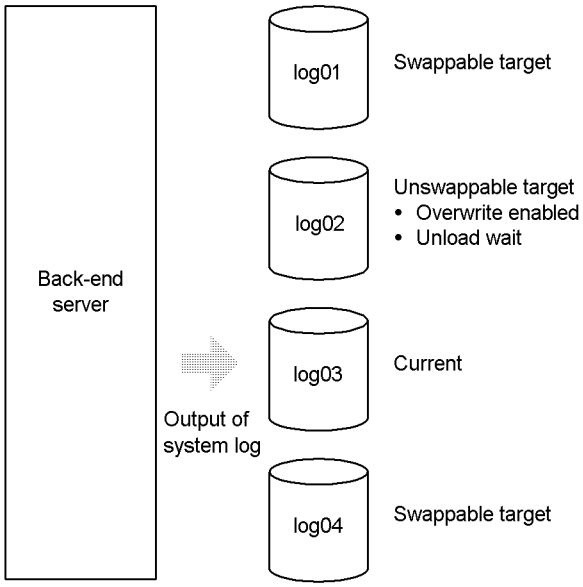
Important

If the backup copy made in this step is used subsequently to recover the RDAREAs, the unload log files containing system log information subsequent to this current file are used as input to the database recovery utility.

(6) Check the status of the system log file

The pdlogls command is used to check the status of the system log file at the back-end server (bes1):

```
pdlogls -d sys -s bes1
```



(7) Unload the system log

Use the `pdlogunld` command to unload the file (`log02`) that is in unload wait status.

```
pdlogunld -d sys -s bes1 -g log02 -o /unld/unldlog02
```

3.3 Operating without unloading the system log

Executor: HiRDB administrator

When the system operating environment requires unloading of system log files, a filled system log file must be unloaded in timely fashion in order to be prepared for the possibility of a database error. This operating environment requires high CPU and input/output workload and complex actions by the HiRDB administrator. Alternatively, the database can be recovered from an error by using system log information as input to the database recovery utility without having to unload the system log. This operating environment is called operation without unloading the system log. Operation without unloading the system log has the following advantages.

Advantages

- Because it is not necessary to unload system log information, the CPU and input/output workload are reduced and the HiRDB administrator's responsibilities are simplified.
- There is no need to provide disk space for storing unload log files.
- The HiRDB administrator does not need to unload files that are in unload wait status. However, the database must be backed up periodically, and the obtained log point information file must be used to release the system log files.

Log point concept

Operating without unloading the system log employs the log point concept.

When a database is to be recovered from an error, system log information obtained before a backup of the database was made is not needed. The point separating the needed system log information from the unneeded system log information is called a log point. A log point is set when a backup is made by the database copy utility.

When a log point is set during the backup operation by the database copy utility, the log point information shown in Figure 3-3 is output to the log point information file. This information is used to release the system log files.

Figure 3-3: Log point information

```
 # The Log Point Information
system_id                = aaaa      ••• 1.
unit_id                  = bbbb      ••• 2.
type                     = ccc       ••• 3.
server_name              = dddddddd  ••• 4.
server_runit             = Oxeeeeeeee ••• 5.
log server_runit         = Oxffffffff ••• 6.
filegroup                = gggggggg  ••• 7.
generation               = Oxhhhhhhhh ••• 8.
block                    = Oxiiiiiiii ••• 9.
header_write_counter     = Oxjjjjjjj ••• 10.
header_write_time        = Oxkkkkkkkk ••• 11.
```

Explanation:

1. HiRDB identifier
2. Unit identifier
3. Type of log
4. Name of server at which log was acquired
5. LAN-ID of server at which log was acquired
6. LAN-ID of log server that set the log point information
7. Name of first file group that contains log information following the log point
8. Generation number of first file group that contains log information following the log point
9. Number of first block that contains log information following the log point
10. Number of times header was updated during allocation of the current file
11. Usage begin time

3.3.1 HiRDB/Single Server

Backup acquisition units

A backup is made of the entire system (all RDAREAs). The obtained log point information file is used to release the system log files.

Backup acquisition interval

A backup should be made daily at a specified time.

The backup acquisition interval depends on the system log file size; the larger the

system log file size, the longer the backup acquisition interval can be. The backup acquisition interval should be set so that the system will never run out of files in swappable target status.

Number of system log files

The total number of system log files should be more than double the number of system log files needed for one day.

For example, if two system log files are used per day, no fewer than four system log files should be provided.

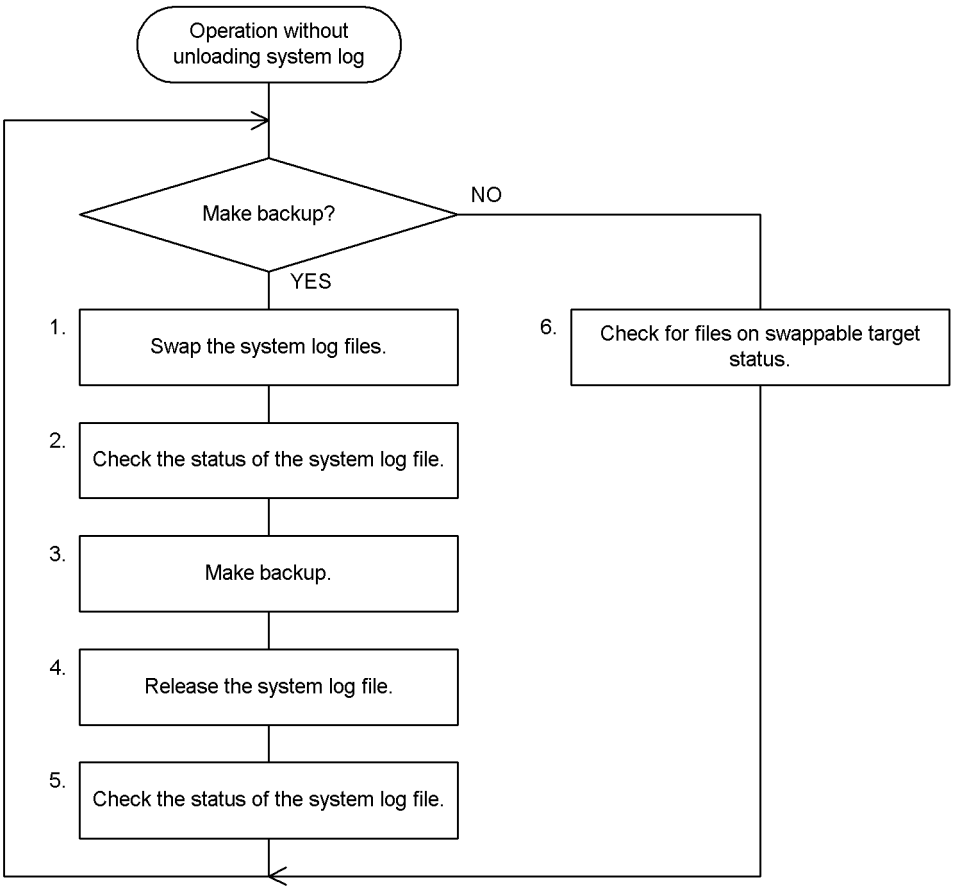
Database recovery in the event of an error

If a database error occurs during operation without unloading the system log, the database must be recovered from its backup copy and the unload log files that contain system log information obtained since the backup was made. For details on the database recovery procedure, see *19. Database Recovery Procedures*.

Example

Figure 3-4 shows the procedure for operation without unloading the system log at a HiRDB/Single Server.

Figure 3-4: Procedure for operation without unloading the system log (HiRDB/Single Server)



Note

The numbers to the left of the process boxes correspond to the paragraph numbers of the explanations on the following pages. For example, step 5 is explained in paragraph (5) below.

(1) Swap the system log files

Before a backup copy is made, the `pdlogswap` command is used to swap the system log files.

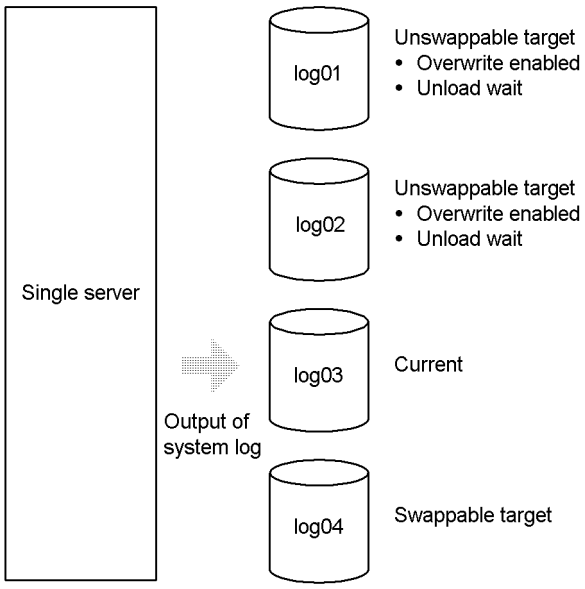
System log files are swapped in order to physically separate the system logs needed for database restoration. The system log files storing the system log information needed for database restoration are those that become primary from this point on:

```
pdlogswap -d sys -w
```

(2) Check the status of the system log file

The `pdlogls` command is used to check the status of the system log file:

```
pdlogls -d sys
```



Explanation

- Files `log01` and `log02` were used today, so these two files are in unload wait status.
- Synchronization point dumps have been validated because the system log files were swapped by the `pdlogswap -w` command. In this manner, files `log01` and `log02` are placed in overwrite enabled status.

(3) Make a backup

The `pdcopy` command (database copy utility) is used to back up all RDAREAs; log point information is also obtained by specifying the `-z` option:

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01
-z /pdcopy/logpoint01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies r as the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up.

-b: Specifies the name of the backup file.

-z: Specifies the name of the log point information file.

In the event of an error in the log point information file

If an error occurs in the log point information file, it must be re-created using the backup file obtained in this step; the pdrstr -z command (database recovery utility) is used to re-create a log point information file:

```
pdrstr -b /pdcopy/backup01 -z /pdcopy/logpoint01
```

(4) Release the system log file

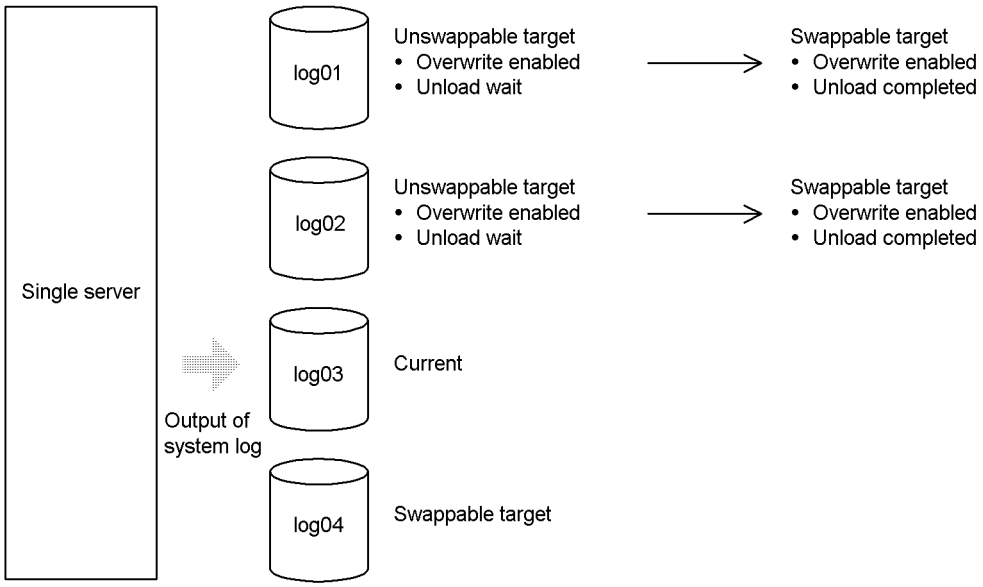
The pdlogchg command is used to release the system log files that predate the log point (log01 and log 02); the log point information file obtained in step (3) is specified in the -z option:

```
pdlogchg -z /pdcopy/logpoint01
```

(5) Check the status of the system log file

The pdlogls command is used to check the status of the system log file:

```
pdlogls -d sys
```



Explanation

- Once the system log file has been released, the file status changes from unload wait to unload completed.
- The file is now in overwrite enabled and unload completed status; therefore, it changes from unswappable target to swappable target.

(6) Check for files in swappable target status

The `pdlogls` command is used to check for files in swappable target status:

```
pdlogls -d sys
```

Important

If system log file swapping occurs when there is no file in swappable target status, the HiRDB/Single Server terminates abnormally. For this reason, the HiRDB administrator must ensure that there is always available a file in swappable target status.

When there is no file in swappable target status, HiRDB outputs the `KFPS01224-I` message to the message log file and the syslogfile.

3.3.2 HiRDB/Parallel Server

Backup acquisition units

A backup must be made at each back-end server and dictionary server. The log point information file obtained for each server is used to release the respective server's system log files.

Backup acquisition interval

Backups should be made daily at a specified time.

The backup acquisition interval depends on the system log file size; the larger the system log file size, the longer the backup acquisition interval can be. The backup acquisition interval should be set so that the system will never run out of files in swappable target status.

Number of system log files

The total number of system log files for each server should be more than double the number of system log files needed for one day by the server.

For example, if two system log files are used per day at a back-end server, no fewer than four system log files should be provided for that back-end server.

System log files for a front-end server

System log information for a front-end server is not needed for database recovery; therefore, `pd_log_unload_check=N` should be specified in the front-end server definition. When this specification is made, checking of system log file unload status is released for the front-end server. This eliminates the need to release the front-end server's system log files. When `pd_log_unload_check=N` is not specified, the `pdlogchg` command must be used at the front-end server to change the file status from unload wait to unload completed.

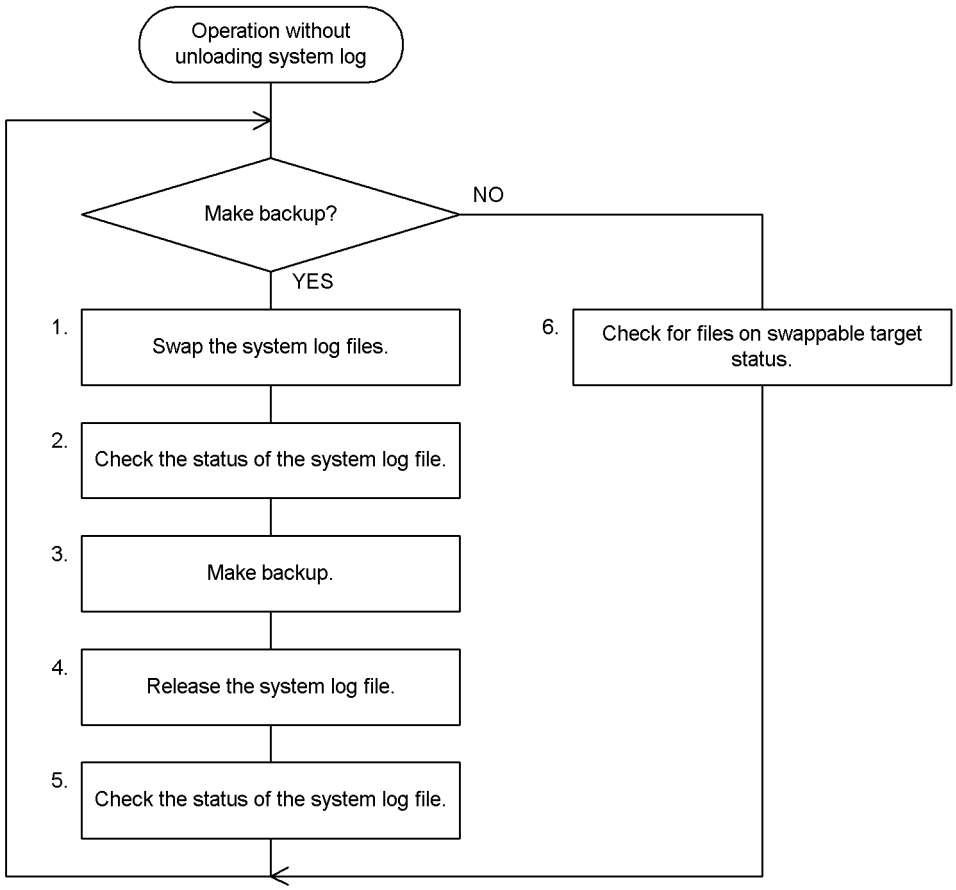
Database recovery in the event of an error

If a database error occurs during operation without unloading the system log, the database must be recovered from its backup copy and the unload log files that contain system log information obtained since the backup was made. For details on the database recovery procedure, see *19. Database Recovery Procedures*.

Example

Figure 3-5 shows the procedure for operation without unloading the system log at a HiRDB/Parallel Server.

Figure 3-5: Procedure for operation without unloading the system log (HiRDB/ Parallel Server)



Note

The numbers to the left of the process boxes correspond to the paragraph numbers of the explanations on the following pages. For example, step 5 is explained in paragraph (5) below.

(1) Swap the system log files

Before a backup copy is made, the `pdlogswap` command is used to swap the system log files at the server that is to be backed up.

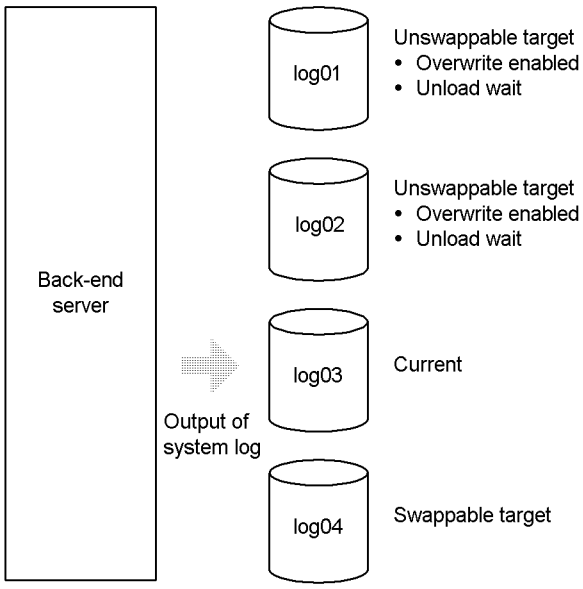
System log files are swapped in order to physically separate the system logs needed for database restoration. The system log files storing the system log information needed for database restoration are those that become primary from this point on:

```
pdlogswap -d sys -s bes1 -w
```

(2) Check the status of the system log file

The `pdlogls` command is used to check the status of the system log file:

```
pdlogls -d sys -s bes1
```



Explanation

- Files `log01` and `log02` were used today, so these two files are in unload wait status.
- Synchronization point dumps have been validated because the system log files were swapped by the `pdlogswap -w` command. In this manner, files `log01` and `log02` are placed in overwrite enabled status.

(3) Make a backup

The `pdcopy` command (database copy utility) is used to back up all RDAREAs in the back-end server (`bes1`); log point information is also obtained by specifying the `-z` option:

```
pdcopy -m /rdarea/mast/mast01 -M r -s bes1 -b /pdcopy/bes1bkup01
-z /pdcopy/bes1logp01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies r as the backup acquisition mode.

-s: Specifies that all RDAREAs in the back-end server (bes1) are to be backed up.

-b: Specifies the name of the backup file.

-z: Specifies the name of the log point information file.

In the event of an error in the log point information file

If an error occurs in the log point information file, it must be re-created using the backup file obtained in this step; the pdrstr -z command (database recovery utility) is used to re-create a log point information file:

```
pdrstr -b /pdcopy/bes1bkup01 -z /pdcopy/bes1logp01
```

(4) Release the system log file

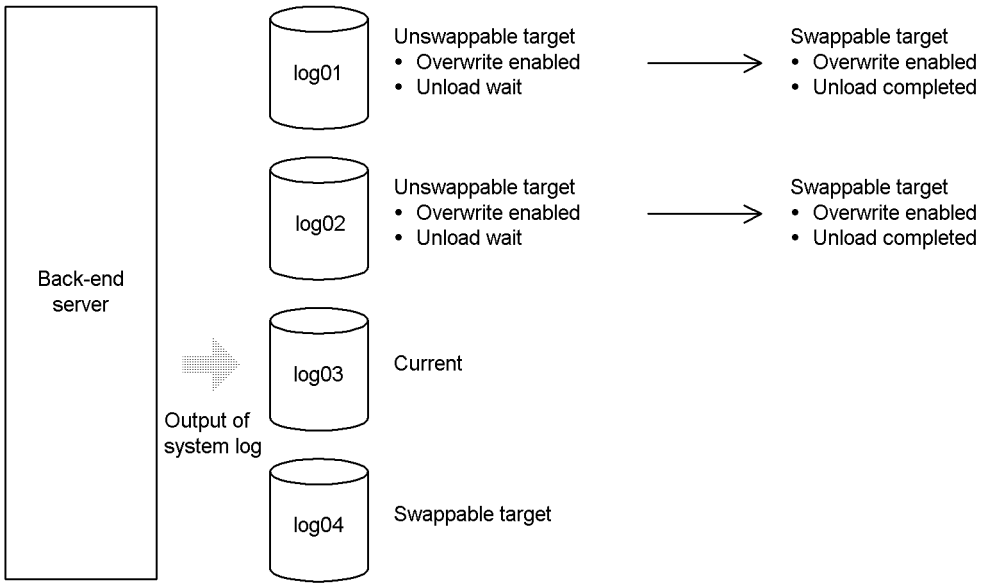
The pdlogchg command is used to release the system log files that predate the log point (log01 and log02); the log point information file obtained in step (3) is specified in the -z option:

```
pdlogchg -z /pdcopy/bes1logp01 -x host01
```

(5) Check the status of the system log file

The pdlogls command is used to check the status of the system log file at the back-end server (bes1).

```
pdlogls -d sys -s bes1
```



Explanation

- Once the system log file has been released, the file status changes from unload wait to unload completed.
- The file is now in overwrite enabled and unload completed status; therefore, it changes from unswappable target to swappable target.

(6) Check for files in swappable target status

The `pdlogls` command is used to check for files in swappable target status:

```
pdlogls -d sys -s bes1
pdlogls -d sys -s bes2
pdlogls -d sys -s bes3
pdlogls -d sys -s bes4
pdlogls -d sys -s dic
```

Important

If system log file swapping occurs when there is no file in swappable target status, the unit terminates abnormally. For this reason, the HiRDB administrator must ensure that there is always available a file in swappable target status. When there is no file in swappable target status, HiRDB outputs the `KFPS01224-I` message to the message log file and the syslogfile.

3.4 Releasing checking of unload status

Executor: HiRDB administrator

HiRDB keeps track of the unload status of system log files and protects files in unload wait status from being overwritten. The HiRDB administrator is responsible for changing the status of such files to unload completed either by unloading them or by releasing them.

When system log information (unload logs) is not used for database recovery, neither the system log unload operation (execution of `pdlogunld` command) nor the system log release operation (execution of `pdlogchg` command) is necessary. In such a case, checking by HiRDB of the unload status of system logs should be released. This eliminates the system log file unload and release operations from the HiRDB administrator's responsibilities.

(1) Swappable conditions

When checking of unload status has been released, there are only three swappable conditions:

- Overwrite enabled
- Extraction completed status (HiRDB Datareplicator)
- Overwriting permitted status for online reorganization (HiRDB Staticizer Option)

The unload status is no longer used to determine whether a system log file is a swappable target.

(2) Advantages

- Operations are simpler because the system log file unload and release operations are eliminated.
- There is no need to provide space for storing unload log files.

(3) Criteria

This operating environment is used principally for referencing-based databases.

For example, this operating environment should be employed if a database can be recovered as described below without having to use system log information:

- Database can be recovered simply by reloading the data.
- Database can be recovered from a backup copy.
- Data updated since the backup was made can be restored by reexecuting the UAPs or utilities.

HiRDB/Parallel Server

- This operating environment can be applied to each server individually. This means that it can be used for any back-end server that satisfies the criteria.
- There is never a need to unload system log information from a front-end server because such information is not required for database recovery. Therefore, this operating environment should always be used for a front-end server.

(4) Notes

(a) Database recovery

If this operating environment is used when system log information is needed for database recovery, it will not be possible to recover the database.

(b) System log file size

If one transaction updates a large amount of data, all system log files may be used by this transaction alone. In such a case, HiRDB prohibits overwriting of all system log files (places them in overwrite disabled status). As a result, there will be no system log file that can be allocated as the current file.

(c) Restrictions on the pdlogunld and pdlogchg commands

1. The following commands cannot be used while HiRDB is operating:
   - `pdlogunld` (except when the `-f` option is specified)
   - `pdlogchg` (except when the `-R` option is specified)

   While HiRDB is inactive, these commands can be used normally (all options are available).
2. HiRDB must not be started while the `pdlogunld` or `pdlogchg` command is executing; otherwise, the `pdlogunld` or `pdlogchg` command may terminate with an error. An unload log file may be created even though the `pdlogunld` command has terminated with an error; however, such an unload log file cannot be used for database recovery.

(5) Environment setup

To use this operating environment, the following specification must be made in each server's definition in the HiRDB system definitions:

- `pd_log_unload_check=N`

In the case of a HiRDB/Single Server, this operand is specified in the single server definition; in the case of a HiRDB/Parallel Server, this option is specified in the server common definition or in the following individual server definitions:

- Front-end server definition
- Back-end server definition

- Dictionary server definition

(6) Procedure

The pdlogls command is used to check for files in swappable target status. There is no need to unload or release system log files.

Procedure for executing a database-updating UAP

Use the following procedure for executing a database-updating UAP.

1. Use the pdlogswap command to swap the system log files:

   ```
   pdlogswap -d sys -s b001
   ```

2. Use the pdcopy command (database copy utility) to back up the RDAREAs:

   ```
   pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01
   ```

3. Execute the UAP.

This procedure is not necessary if the data can be stored again in the table and the UAP and all other subsequent processing can be reexecuted.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

(7) In the event of an error in a system log file

When an error occurs in a system log file, it is placed in standby status and will not be allocated as the current file during subsequent HiRDB operation. Erroneous system log files include not only files containing a physical error but also the following files:

- When forced normal startup (pdstart dbdestroy or pdstart -i) is executed after an HiRDB abnormal termination, the system log file that was used as the current file during the previous HiRDB session.
- When unit startup failed due to an error during normal startup processing, the system log file that was allocated as the current file during the previous startup processing (applicable to the server for which the current file was allocated before completion of startup).

An erroneous system log file cannot be allocated as the current file.

A system log file allocated as the current file during the previous HiRDB session can be reallocated as the current file by executing the pdlogchg or pdlogunld command.

It is possible for a system log file that contains a physical error to be allocated as the current file during the next HiRDB session by executing the pdlogchg or pdlogunld command. However, if an erroneous system log file is actually used as the current file, HiRDB will terminate abnormally. In such a case, the error must be corrected before

the `pdlogchg` or `pdlogunld` command is executed.

3.5 Procedures for manipulating system log files

Executor: HiRDB administrator

This section describes the procedures for manipulating system log files. The following topics are covered:

1. Checking for files in swappable target status
2. When there is no file in swappable target status
3. Unloading the current file
4. Unloading a file in unload completed status
5. When the system log in a file in unload wait status is not needed
6. Changing a file's status
7. Increasing (or reducing) the system log file size during HiRDB operation
8. Adding a new system log file
9. Deleting a system log file

3.5.1 Checking for files in swappable target status

The `pdlogls` command is used to check for files in swappable target status.

If there is no file in swappable target status, one must be created; see *3.5.2 When there is no file in swappable target status* for details.

If swapping of system log files occurs when there are no files in swappable target status, HiRDB (the unit for a HiRDB/Parallel Server) terminates abnormally. When there is no file in swappable target status, HiRDB outputs a message to that effect.

3.5.2 When there is no file in swappable target status

The `pdlogls` command is used to check for files in overwrite enabled status.

(1) When there is a file in overwrite enabled status

If there is a file in overwrite enabled status, it should be placed in unload completed status, thus creating a file in swappable target status. There are two ways to change a file's status from unload wait to unload completed:

- Use the `pdlogunld` command to unload the file
- Use the `pdlogchg` command to discard the system log information in the file

Care must be exercised in executing the `pdlogchg` command on a file that contains system log information that is required for system or database recovery. If the file is overwritten, system log information required for system and database recovery will be

lost, in which case it will not be possible to recover the database in the event of an error.

If you are using the HiRDB Datareplicator, you must release the extracting status. When you are using the updatable online reorganization function of the HiRDB Staticizer Option, you must release the overwriting denied status for online reorganization. If you do not release these statuses, it will not be possible for the file to be placed in swappable target status.

(2) When there is no file in overwrite enabled status

Add a system log file. For details, see *3.5.8 Adding a new system log file*.

3.5.3 Unloading the current file

To unload the current file during database recovery, for example, execute the `pdlogunld -f` command. However, the status of the file may not change even though the `pdlogunld -f` command is executed.

3.5.4 Unloading a file in unload completed status

Use the procedure explained below to unload a file in unload completed status.

Procedure

1. Use the `pdlogcls` command to close the file to be unloaded (place it in reserved status).

   ```
   pdlogcls -d sys -s b001 -g syslog01
   ```

2. Use the `pdlogunld -f` command to execute the unload function. Note that the status of the file will not change.

   ```
   pdlogunld -d sys -s b001 -g syslog01 -f
   ```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

3.5.5 When the system log in a file in unload wait status is not needed

If the system log in a file in unload wait status is not needed, the `pdlogchg` command can be used to change the file's status from unload wait to unload completed.

3.5.6 Changing a file's status

(1) Changing the file used as the current file

To use a different file as the current file, the `pdlogswap` command is executed to swap files. The current file is placed in unswappable target status, and a file in swappable target status becomes the current file. This command should be used when the system log information contained in the current file is needed.

The pdlogswap command cannot be executed if there are no files that can be placed in swappable target status.

(2) Placing a reserved file in overwrite enabled status

The pdlogopen command can be used to place a reserved file in overwrite enabled status. When normal startup of HiRDB is executed, such a file will become overwrite disabled.

(3) Reserving a file in overwrite enabled status

The pdlogcls command can be used to reserve a file that is in overwrite enabled status. If there is only one file in overwrite enabled status, it cannot be reserved. A file in overwrite disabled status must not be reserved if it contains system log information needed for system recovery.

(4) Changing a file from extracting status to extraction completed status

The pdlogchg -R command can be used to forcibly change a file's status from extracting to extraction completed. When the file status is changed forcibly in this manner, the system log required by the HiRDB Datareplicator cannot be extracted. Also, the file may not be synchronized to the target database, or data linkage may not be performed. You should execute the pdlogchg -R command only in cases such as when an error occurs and it is necessary to re-create the system log file. You should not execute the pdlogchg -R command in any other cases.

(5) Changing a file from overwriting denied status for online reorganization to operating permitted status for online reorganization

The pdlogchg -G command can be used to forcibly change a file from overwriting denied status for online reorganization to overwriting permitted status for online reorganization. When the file status is changed forcibly in this manner, the system log required for update processing will be lost, and it may not be possible to perform updatable online reorganization. You should execute the pdlogchg -G command only in cases such as when an error occurs and it is necessary to re-create the system log file. You should not execute the pdlogchg -R command in any other cases.

3.5.7 Increasing (or reducing) the system log file size during HiRDB operation

Use the procedure explained below to increase the system log file size.

Procedure

1. Use the pdlogs command to check the status of the system log files.

   ```
   pdlogls -d sys -s b001
   ```

2. Use the pdlogcls command to reserve files that can be placed in swappable target status.

```
pdlogcls -d sys -s b001 -g syslog01
```

3. Use the pdlogrm command to delete reserved files.

```
pdlogrm -d sys -s b001 -f /sysfile01/syslog1a
pdlogrm -d sys -s b001 -f /sysfile01/syslog1b
```

4. Use the pdloginit command to re-create the system log files deleted in step 3.

   In such a case, increase the record count to a value greater than that for the previous system log files. When reducing the file size, also reduce the record count.

```
pdloginit -d sys -s b001 -f /sysfile01/syslog1a -n 5000
pdloginit -d sys -s b001 -f /sysfile01/syslog1b -n 5000
```

5. Use the pdlogopen command to place the files created in step 4 in swappable target status.

```
pdlogopen -d sys -s b001 -g syslog01
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

Note

When you change files in swappable target status to reserved status, be sure not to change all of them. If swapping of system log files occurs when there are no files in swappable target status, HiRDB (or the unit for a HiRDB/Parallel Server) terminates abnormally. You should make sure that there is still at least one swappable target file when you change the system log file size.

3.5.8 Adding a new system log file

This section explains the procedure for adding a new system log file.

Before a system log file is added, it is recommended that the system log file design guidelines be reviewed; for details, see the manual *HiRDB Version 8 Installation and Design Guide*.

(1) When HiRDB can be terminated normally

Procedure

1. Use the pdfstatfs command to check for free space in the HiRDB file system area for creating system log files.

```
pdfstatfs /sysfile01
```

   If there is no free space, create a new HiRDB file system area. For details

about creating a HiRDB file system area, see *10.2 Creating (initializing) a HiRDB file system area*.

2. Use the `pdstop` command to terminate HiRDB normally.
3. Add one of the operands listed below to the HiRDB system definition. Specify in the operand the system log files added in step 4.
   - `pdlogadfg -d sys` operand
   - `pdlogadpf -d sys` operand
4. Use the `pdloginit` command to add (initialize) system log files.

   ```
   pdloginit -d sys -s b001 -f /sysfile01/syslog1a -n 5000
   pdloginit -d sys -s b001 -f /sysfile01/syslog1b -n 5000
   ```
5. Use the `pdconfchk` command to check the HiRDB system definition. If an error is detected, correct the HiRDB system definition.
6. Use the `pdstart` command to start HiRDB normally.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

(2) When HiRDB cannot be terminated normally

You can use the `pdchgconf` system reconfiguration command to change the HiRDB system definition. However, HiRDB Advanced High Availability must have been installed to use this command.

Procedure

1. Use the `pdfstatfs` command to check for free space in the HiRDB file system area for creating system log files.

   ```
   pdfstatfs /sysfile01
   ```

   If there is no free space, create a new HiRDB file system area. For details about creating a HiRDB file system area, see *10.2 Creating (initializing) a HiRDB file system area*.
2. Create the `$PDDIR/conf/chgconf` directory.
3. Copy the HiRDB system definition files being used to the directory created in step 2.
4. Add one of the following operands to the HiRDB system definition in the `$PDDIR/conf/chgconf` directory. Specify in this operand the system log files that will be added in step 5.
   - `pdlogadfg -d sys` operand

- `pdlogadpf -d sys` operand

5. Use the `pdloginit` command to add (initialize) system log files.

```
pdloginit -d sys -s b001 -f /sysfile01/syslog1a -n 5000
pdloginit -d sys -s b001 -f /sysfile01/syslog1b -n 5000
```

6. Use the `pdconfchk` command to check the HiRDB system definition in the `$PDDIR/conf/chgconf` directory. If an error is detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
7. Use the `pdchgconf` command to replace the HiRDB system definition with the modified HiRDB system definition.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

Remarks

When you use the procedure described here to add system log files, the system log files are swapped while the `pdchgconf` command is being executed. If there are no files in swappable target status, an added system log file is swapped in.

3.5.9 Deleting a system log file

The `pdlogrm` command is used to delete a reserved system log file. Only a reserved system log file can be deleted.

Delete the operands listed below as necessary. Delete the operands corresponding to the system log files that were deleted using the `pdlogrm` command.

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

If these operands are not deleted, the deleted system log files will become virtual files.

3.6 Status changes of system log files

Tables 3-4 to 3-7 list the status changes for system log files while HiRDB is operating.

Remarks

- The status transitions listed in these tables assume that the relevant events were processed normally.
- HiRDB manages the status of system log files only while it is operating.
- The status of system log files displayed by the `pdlogls` command does not change even if the `pdlogunld` or `pdlogchg` command is executed on reserved files. To view the status of the system log files, execute the `pdlogopen` command then execute the `pdlogls` command.

Table 3-4: Status changes of system log files while HiRDB is operating (part 1)

Event			Status of system log files							
			Current	Standby (unload wait)						
				Overwrite enabled				Overwrite disabled		
				Enable		Disable		Enable		Disable	
				C	I	C	I	C	I	C	I
			1	2	3	4	5	6	7	8	9
Swapping occurred when system log file area was full			→ 9	—	—	—	—	—	—	—	—
Error in current file caused file swapping to occur	Error occurred during nonduplexing, or error occurred in A and B systems during duplexing		→ 25	—	—	—	—	—	—	—	—
	Error occurred in one system during duplexing	Single system can operate	→ 9	—	—	—	—	—	—	—	—
		Single system cannot operate	→ 25	—	—	—	—	—	—	—	—
`pdlogswap` command was executed			→ 9	—	—	—	—	—	—	—	—
Validation of synchronization point dump			—	—	—	—	—	→ 2	→ 3	→ 4	→ 5
Unloading by the automatic log unloading facility was completed			—	→ 10	→ 11	→ 12	→ 13	→ 14	→ 15	→ 16	→ 17

Event	Status of system log files								
	Current	Standby (unload wait)							
		Overwrite enabled				Overwrite disabled			
		Enable		Disable		Enable		Disable	
		C	I	C	I	C	I	C	I
	1	2	3	4	5	6	7	8	9
All data extraction by HiRDB Datareplicator was completed	—	—	→ 2	—	→ 4	—	→ 6	—	→ 8
Reading of all data in system log files by update processing of updatable online reorganization was completed (`pd_log_org_reflected_logpoint=release` is specified)	—	—	—	→ 2	→ 3	—	—	→ 6	→ 7
Update processing of updatable online reorganization was completed	—	—	—	→ 2	→ 3	—	—	→ 6	→ 7
Overwriting denied status for online reorganization was released by full detection of a system log file (`pd_log_org_no_standby_file_opr=continue is specified`)	—	—	—	→ 2	→ 3	—	—	→ 6	→ 7
`pdlogunld` command was executed	—	→ 10	→ 11	→ 12	→ 13	→ 14	→ 15	→ 16	→ 17
`pdlogchg` command was executed	—	→ 10	→ 11	→ 12	→ 13	→ 14	→ 15	→ 16	→ 17
`pdlogchg -R` was executed	—	—	→ 2	—	→ 4	—	→ 6	—	→ 8
`pdlogchg -G` was executed	—	—	—	→ 2	→ 3	—	—	→ 6	→ 7
`pdlogopen` command was executed*	—	—	—	—	—	—	—	—	—
`pdlogcls` command was executed*	—	→ 18	→ 19	→ 20	→ 21	→ 22	→ 23	→ 24	→ 25

Legend:

Enable: Overwriting permitted status for online reorganization

Disable: Overwriting denied status for online reorganization

C: Extraction complete

I: Extraction incomplete

—: Does not apply, or status does not change.

→*n*: Status to which system log file changes after occurrence of the event.

For example, →1 indicates that after the event occurs, the status of the system log file changes to current (status No. 1).

Note: See Tables 3-5 to 3-7 for system log file status Nos. 10 and higher.

* Applies when the `-a` or `-b` option is not specified.

Table 3-5: Status changes of system log files while HiRDB is operating (part 2)

Event			Status of system log files							
			Standby (unload wait)							
			Overwrite enabled				Overwrite disabled			
			Enable		Disable		Enable		Disable	
			C	I	C	I	C	I	C	I
			10	11	12	13	14	15	16	17
Swapping occurred when system log file area was full			→1	—	—	—	—	—	—	—
Error in current file caused file swapping to occur	Error occurred during nonduplexing, or error occurred in A and B systems during duplexing		→1	—	—	—	—	—	—	—
	Error occurred in one system during duplexing	Single system can operate	→1	—	—	—	—	—	—	—
		Single system cannot operate	→1	—	—	—	—	—	—	—
`pdlogswap` command was executed			→1	—	—	—	—	—	—	—
Validation of synchronization point dump			—	—	—	—	→10	→11	→12	→13
Unloading by the automatic log unloading facility was completed			—	—	—	—	—	—	—	—
All data extraction by HiRDB Datareplicator was completed			—	→10	—	→12	—	→14	—	→16

Event	Status of system log files							
	Standby (unload wait)							
	Overwrite enabled				Overwrite disabled			
	Enable		Disable		Enable		Disable	
	C	I	C	I	C	I	C	I
	10	11	12	13	14	15	16	17
Reading of all data in system log files by update processing of updatable online reorganization was completed (`pd_log_org_reflected_logpoint=release` is specified)	—	—	→ 10	→ 11	—	—	→ 14	→ 15
Update processing of updatable online reorganization was completed	—	—	→ 10	→ 11	—	—	→ 14	→ 15
Overwriting denied status for online reorganization was released by full detection of a system log file (`pd_log_org_no_standby_file_opr=continue` is specified)	—	—	→ 10	→ 11	—	—	→ 14	→ 15
`pdlogunld` command was executed	—	—	—	—	—	—	—	—
`pdlogchg` command was executed	—	—	—	—	—	—	—	—
`pdlogchg -R` was executed	—	→ 10	—	→ 12	—	→ 14	—	→ 16
`pdlogchg -G` was executed	—	—	→ 10	→ 11	—	—	→ 14	→ 15
`pdlogopen` command was executed*	—	—	—	—	—	—	—	—
`pdlogcls` command was executed*	→ 26	→ 27	→ 28	→ 29	→ 30	→ 31	→ 32	→ 33

Legend:

Enable: Overwriting permitted status for online reorganization

Disable: Overwriting denied status for online reorganization

C: Extraction complete

I: Extraction incomplete

—: Does not apply, or status does not change.

→*n*: Status to which system log file changes after occurrence of the event.

For example, ➔1 indicates that after the event occurs, the status of the system log file changes to current (status No. 1).

Note: See Table 3-4 for system log file status Nos. 1-9 and Tables 3-6 and 3-7 for system log file status Nos. 18 and higher.

* Applies when the `-a` or `-b` option is not specified.

Table 3-6: Status changes of system log files while HiRDB is operating (part 3)

Event			Status of system log files							
			Reserved and unload wait state							
			Overwrite enabled				Overwrite disabled			
			Enable		Disable		Enable		Disable	
			C	I	C	I	C	I	C	I
			18	19	20	21	22	23	24	25
Swapping occurred when system log file area was full			—	—	—	—	—	—	—	—
Error in current file caused file swapping to occur	Error occurred during nonduplexing, or error occurred in A and B systems during duplexing		—	—	—	—	—	—	—	—
	Error occurred in one system during duplexing	Single system can operate	—	—	—	—	—	—	—	—
		Single system cannot operate	—	—	—	—	—	—	—	—
`pdlogswap` command was executed			—	—	—	—	—	—	—	—
Validation of synchronization point dump			—	—	—	—	➔18	➔19	➔20	➔21
Unloading by the automatic log unloading facility was completed			➔26	➔27	➔28	➔29	➔30	➔31	➔32	➔33
All data extraction by HiRDB Datareplicator was completed			—	➔18	—	➔20	—	➔22	—	➔24
Reading of all data in system log files by update processing of updatable online reorganization was completed (`pd_log_org_reflected_logpoint=release` is specified)			—	—	➔18	➔19	—	—	➔22	➔23
Update processing of updatable online reorganization was completed			—	—	➔18	➔19	—	—	➔22	➔23

Event	Status of system log files							
	Reserved and unload wait state							
	Overwrite enabled				Overwrite disabled			
	Enable		Disable		Enable		Disable	
	C	I	C	I	C	I	C	I
	18	19	20	21	22	23	24	25
Overwriting denied status for online reorganization was released by full detection of a system log file (`pd_log_org_no_standby_file_opr=continue` is specified)	—	—	→ 18	→ 19	—	—	→ 22	→ 23
`pdlogunld` command was executed	→ 26	→ 27	→ 28	→ 29	→ 30	→ 31	→ 32	→ 33
`pdlogchg` command was executed	→ 26	→ 27	→ 28	→ 29	→ 30	→ 31	→ 32	→ 33
`pdlogchg -R` was executed	—	→ 18	—	→ 20	—	→ 22	—	→ 24
`pdlogchg -G` was executed	—	—	→ 18	→ 19	—	—	→ 22	→ 23
`pdlogopen` command was executed*	→ 2	→ 3	→ 4	→ 5	→ 6	→ 7	→ 8	→ 9
`pdlogcls` command was executed*	—	—	—	—	—	—	—	—

Legend:

Enable: Overwriting permitted status for online reorganization

Disable: Overwriting denied status for online reorganization

C: Extraction complete

I: Extraction incomplete

—: Does not apply, or status does not change.

→*n*: Status to which system log file changes after occurrence of the event.

For example, → 1 indicates that after the event occurs, the status of the system log file changes to current (status No. 1).

Note: See Tables 3-4 and 3-5 for system log file status Nos. 1-17 and Table 3-7 for system log file status Nos. 26 and higher.

* Applies when the `-a` or `-b` option is not specified.

Table 3-7: Status changes of system log files while HiRDB is operating (part 4)

Event			Status of system log files							
			Reserved and unload wait state							
			Overwrite enabled				Overwrite disabled			
			Enable		Disable		Enable		Disable	
			C	I	C	I	C	I	C	I
			26	27	28	29	30	31	32	33
Swapping occurred when system log file area was full			—	—	—	—	—	—	—	—
Error in current file caused file swapping to occur	Error occurred during nonduplexing, or error occurred in A and B systems during duplexing		—	—	—	—	—	—	—	—
	Error occurred in one system during duplexing	Single system can operate	—	—	—	—	—	—	—	—
		Single system cannot operate	—	—	—	—	—	—	—	—
`pdlogswap` command was executed			—	—	—	—	—	—	—	—
Validation of synchronization point dump			—	—	—	—	→ 26	→ 27	→ 28	→ 29
Unloading by the automatic log unloading facility was completed			—	—	—	—	—	—	—	—
All data extraction by HiRDB Datareplicator was completed			—	→ 26	—	→ 28	—	→ 30	—	→ 32
Reading of all data in system log files by update processing of updatable online reorganization was completed (`pd_log_org_reflected_logpoint=release` is specified)			—	—	→ 26	→ 27	—	—	→ 30	→ 31
Update processing of updatable online reorganization was completed			—	—	→ 26	→ 27	—	—	→ 30	→ 31
Overwriting denied status for online reorganization was released by full detection of a system log file (`pd_log_org_no_standby_file_opr=continue` is specified)			—	—	→ 26	→ 27	—	—	→ 30	→ 31
`pdlogunld` command was executed			—	—	—	—	—	—	—	—

Event	Status of system log files							
	Reserved and unload wait state							
	Overwrite enabled				Overwrite disabled			
	Enable		Disable		Enable		Disable	
	C	I	C	I	C	I	C	I
	26	27	28	29	30	31	32	33
`pdlogchg` command was executed	—	—	—	—	—	—	—	—
`pdlogchg -R` was executed	—	→ 26	—	→ 28	—	→ 30	—	→ 32
`pdlogchg -G` was executed	—	—	→ 26	→ 27	—	—	→ 30	→ 31
`pdlogopen` command was executed*	→ 10	→ 11	→ 12	→ 13	→ 14	→ 15	→ 16	→ 17
`pdlogcls` command was executed*	—	—	—	—	—	—	—	—

Legend:

Enable: Overwriting permitted status for online reorganization

Disable: Overwriting denied status for online reorganization

C: Extraction complete

I: Extraction incomplete

— : Does not apply, or status does not change.

→*n*: Status to which system log file changes after occurrence of the event.

For example, → 1 indicates that after the event occurs, the status of the system log file changes to current (status No. 1).

Note: See Tables 3-4 to 3-6 for system log file status Nos. 1-25.

* Applies when the `-a` or `-b` option is not specified.

3.7 Changing the system log file record length

Executor: HiRDB administrator

The system log file record length can be changed. The record length must be 1024, 2048, or 4096 bytes. The system log size can be reduced by selecting a short record length.

The current record length can be determined by executing the `pdlogls` command.

Notes

- The same record length must be set for all system log files.
- When 1024 or 2048 is specified as the system log file record length, that record length must be specified in the `pd_log_rec_leng` operand. A system log file cannot be opened if its record length does not match the length specified in the `pd_log_rec_leng` operand.

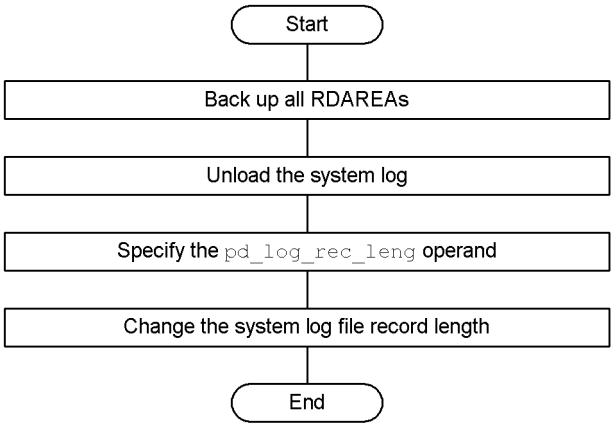
3.7.1 Example 1 (system log unloading operation)

This example changes the system log file record length from 4096 bytes to 1024 bytes.

Conditions

- HiRDB is in normally terminated status.
- If data is linked to HiRDB Datareplicator, HiRDB Datareplicator has been terminated after all data has been extracted from the most recent system log file.

Procedure



(1) Start HiRDB with the pdstart -r command

```
pdstart -r
```

(2) Back up all RDAREAs with the pdcopy command

```
pdcopy -m /rdarea/mast/mast01 -M x -a -b /pdcopy/backup01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode. Because `pdstart -r` was used to start HiRDB, there is no need to place the RDAREAs in shutdown and closed status even if `x` is specified.

`-a`: Specifies that all RDAREAs are to be backed up.

`-b`: Specifies the name of the backup file.

(3) Terminate HiRDB normally with the pdstop command

```
pdstop
```

(4) Unload the system log file in unload wait status with the pdlogunld command

```
pdlogunld -d sys -g log1 -o /unld/unldlog1
```

(5) Add the pd_log_rec_leng operand

The pd_log_rec_leng operand must be added as follows in the server definition:

```
        :
set pd_log_rec_leng = 1024
          :
```

Explanation

Specifies the system log file record length (1024).

(6) Delete system log files with the pdlogrm command

```
pdlogrm -d sys -f /sysfile_a/log1a
pdlogrm -d sys -f /sysfile_b/log1b
pdlogrm -d sys -f /sysfile_a/log2a
pdlogrm -d sys -f /sysfile_b/log2b
pdlogrm -d sys -f /sysfile_a/log3a
pdlogrm -d sys -f /sysfile_b/log3b
pdlogrm -d sys -f /sysfile_a/log4a
pdlogrm -d sys -f /sysfile_b/log4b
```

Explanation

When dual system log files are being used, the B versions of the system log files must also be deleted.

(7) Re-create system log files with the pdloginit command

```
pdloginit -d sys -f /sysfile_a/log1a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log1b -n 2000 -l 1024
pdloginit -d sys -f /sysfile_a/log2a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log2b -n 2000 -l 1024
pdloginit -d sys -f /sysfile_a/log3a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log3b -n 2000 -l 1024
pdloginit -d sys -f /sysfile_a/log4a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log4b -n 2000 -l 1024
```

Explanation

Specifies the new record length (1024) in the `-l` option.

(8) Start HiRDB with the pdstart command

```
pdstart
```

If data is linked to HiRDB Datareplicator, HiRDB Datareplicator must also be started.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

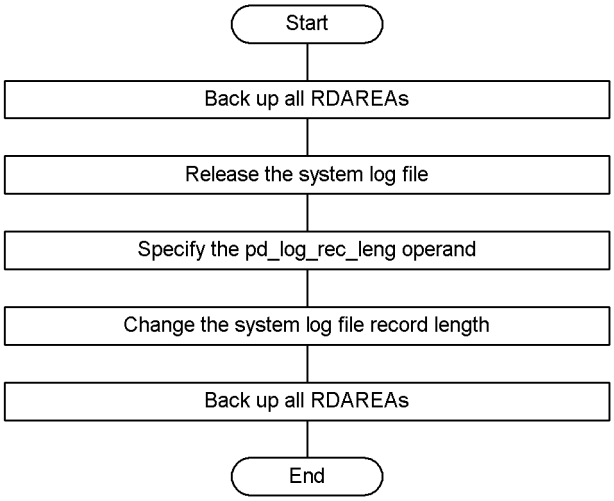
3.7.2 Example 2 (operation without unloading the system log)

This example changes the system log file record length from 4096 bytes to 1024 bytes.

Conditions

- HiRDB is in normally terminated status.
- If data is linked to HiRDB Datareplicator, HiRDB Datareplicator has been terminated after all data has been extracted from the most recent system log file.

Procedure



(1) Start HiRDB with the pdstart command

```
pdstart
```

(2) Back up all RDAREAs with the pdcopy command

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01
-z /pdcopy/logpoint01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. In the case of a HiRDB/Parallel Server, data cannot be backed up by specifying -a; instead, -s must be specified to back up RDAREAs in units of servers.

-b: Specifies the name of the backup file.

-z: Specifies the name of the log point information file.

(3) Terminate HiRDB normally with the pdstop command

```
pdstop
```

(4) Release the system log in unload wait status with the pdlogchg command

```
pdlogchg -d sys -g log1 -z /pdcopy/logpoint01
```

(5) Add the pd_log_rec_leng operand

The pd_log_rec_leng operand must be added as follows in the server definition:

```
          :
set pd_log_rec_leng = 1024
          :
```

Explanation

Specifies the system log file record length (1024).

(6) Delete system log files with the pdlogrm command

```
pdlogrm -d sys -f /sysfile_a/log1a -u
pdlogrm -d sys -f /sysfile_b/log1b -u
pdlogrm -d sys -f /sysfile_a/log2a -u
pdlogrm -d sys -f /sysfile_b/log2b -u
pdlogrm -d sys -f /sysfile_a/log3a -u
pdlogrm -d sys -f /sysfile_b/log3b -u
pdlogrm -d sys -f /sysfile_a/log4a -u
pdlogrm -d sys -f /sysfile_b/log4b -u
```

Explanation

- Specify -u for the forced deletion option.
- When dual system log files are being used, the B versions of the system log files must also be deleted.

(7) Re-create system log files with the pdloginit command

```
pdloginit -d sys -f /sysfile_a/log1a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log1b -n 2000 -l 1024
pdloginit -d sys -f /sysfile_a/log2a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log2b -n 2000 -l 1024
pdloginit -d sys -f /sysfile_a/log3a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log3b -n 2000 -l 1024
pdloginit -d sys -f /sysfile_a/log4a -n 2000 -l 1024
pdloginit -d sys -f /sysfile_b/log4b -n 2000 -l 1024
```

Explanation

Specifies the new record length (1024) in the `-l` option.

(8) Start HiRDB with the pdstart command

```
pdstart
```

If data is linked to HiRDB Datareplicator, HiRDB Datareplicator must also be started.

(9) Back up all RDAREAs with the pdcopy command

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02
-z /pdcopy/logpoint02
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up. In the case of a HiRDB/Parallel Server, data cannot be backed up by specifying `-a`; instead, `-s` must be specified to back up RDAREAs in units of servers.

`-b`: Specifies the name of the backup file.

`-z`: Specifies the name of the log point information file.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

3.8 Using the automatic log unloading facility

Executor: HiRDB administrator

Unloading system log files means that the `pdlogunld` command is entered to unload system log files that are waiting to be unloaded. If this step is skipped, there will be no system log files that are eligible to be swapped in (swappable target status files) and HiRDB will terminate abnormally.

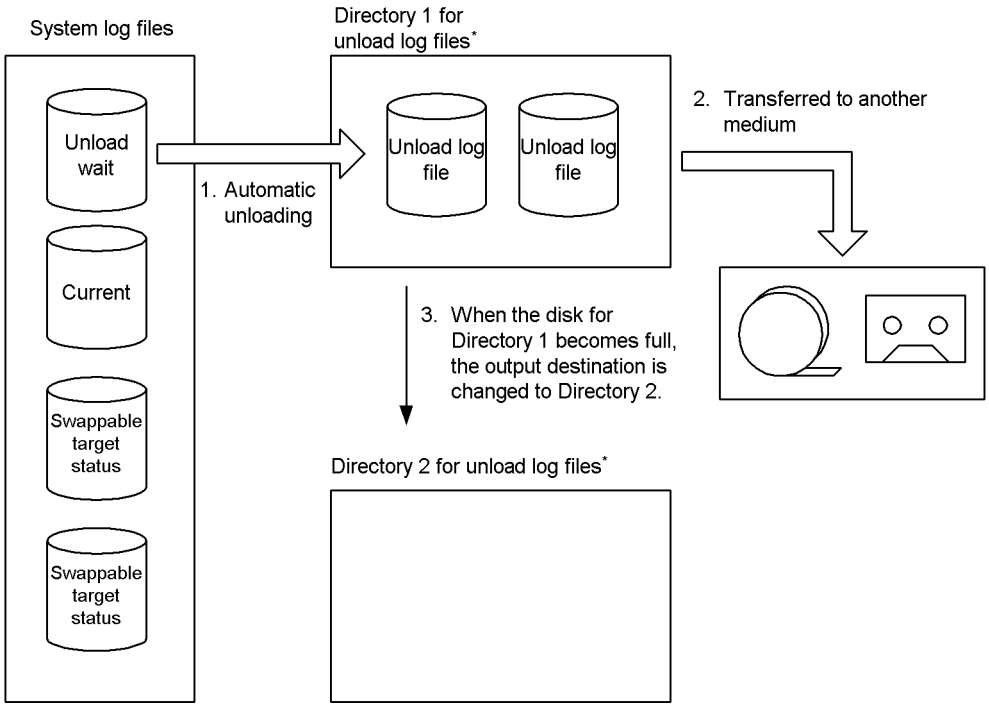
HiRDB provides a facility, called the automatic log unloading facility, for unloading system log files automatically. If frequent unloading of system log files is anticipated, use of the automatic log unloading facility should be considered.

This section assumes that the reader is familiar with the system log unloading operations.

3.8.1 Overview of automatic log unloading facility

When the *automatic log unloading facility* is used, HiRDB unloads automatically the system log files that are waiting to be unloaded. Unload log files are created under a directory created in advance by the HiRDB administrator (this directory is called the *directory for unload log files*). Figure 3-6 shows the operation of the automatic log unloading facility.

Figure 3-6: Automatic log unloading facility



* Unload log files can also be created in a HiRDB file system area.

Explanation

1. When a system log file in unload wait status is generated, it is unloaded automatically. Unload log files are created under the directory for unload log files.
2. The HiRDB administrator should transfer the unload log files to a separate medium at regular intervals.
3. When multiple directories for unload log files are created, the output destination of unload log files is switched to another directory whenever one directory's disk becomes full.

(1) HiRDB administrator's tasks

The automatic log unloading facility stops operating when unload log files cannot be created in the directory for unload log files because of a full disk, disk error, etc. To prevent this from happening, *the HiRDB administrator should transfer the unload log files in the directory for unload log files to a separate medium on a regular basis. At the same time, unneeded unload log files should be deleted.*

When multiple directories for unload log files are created, the automatic log unloading facility stops only when unload log files cannot be created in any of the directories.

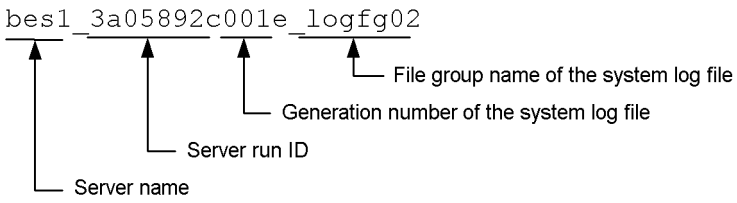
(2) Relationship to other facilities

The automatic log unloading facility cannot be used when HiRDB is not set to check the unload status of system log files (when `pd_log_unload_check = N` is specified).

(3) Names of created unload log files

The format of the names of the unload log files that are created by the automatic log unloading facility is shown below (the names of created unload log files are displayed in the `KFPS01212-I` message):
server-name_server-run-ID-and-generation-number_file-group-name

Example



The names of unload log files created by the automatic log unloading facility can be checked by entering the `pdlogatul` command.

(4) When multiple directories for unload log files are created

When there are multiple unload log file directories, unload log files that cannot be created in one directory because of a full disk, disk error, etc., are created in another directory. HiRDB selects the directories to be used in the order of their specification in the `pd_log_auto_unload_path` operand.

For example, if `pd_log_auto_unload_path = "Directory 1","Directory 2","Directory 3",...` is specified, HiRDB uses Directory 1 first, then Directory 2, then Directory 3, etc. However, if Directory 2 is not available because of a disk error, etc., Directory 3 is used after Directory 1.

When HiRDB is started normally, Directory 1 is used first, assuming that there are no unload log files in Directory 1. If Directory 1 contains unload log files, Directory 2 is used.

When HiRDB is restarted, the directory that was being used when HiRDB was terminated is used.

Note:

You should note the following point when you create multiple directories for unload log files:

- If no empty directory is available when HiRDB is started normally, the automatic log unloading facility stops operating.

3.8.2 Environment setup

To use the automatic log unloading facility, the following environment settings are required:

- Creation of directories for unload log files
- Specification of `pd_log_auto_unload_path` operand

(1) Creation of directories for unload log files

The HiRDB administrator creates directories for unload log files. It is wise to check the availability of disk space before creating a directory. If the amount of available disk space is small, creation of the directory may result in a disk space shortage.

A HiRDB file system area for utilities must be created in order to create a directory for unload log files in an HiRDB file system area.

(a) Required disk space

The required disk space can be determined from the following formula:

Required disk space (bytes) =
total-number-of-records-in-system-log-files-to-be-unloaded ×
system-log-file-record-length ×
number-of-unload-log-files-to-be-created-in-directories × *1.2*

Notes

- Note that the disk space requirement computed here is not for a single directory; it is the total disk space requirement for all directories if multiple directories are created.
- When the disk space available for a single directory is not adequate, multiple directories should be created.

(b) Number of directories for unload log files

Although 128 directories for unload log files can be created, creating too many is not advisable. Four directories at the most are recommended so that their management does not become cumbersome. While a single directory is easy to manage, the automatic log unloading facility will stop if the disk becomes full or an error occurs.

When multiple directories are available, the automatic log unloading facility does not

stop as long as unload log files can be created in at least one of the directories. For this reason, the directories should be created in different partitions.

(c) Important

- In a multi-HiRDB configuration, different sets of directories must be created for the different HiRDBs. If the same directory is specified for more than one HiRDB, it may not be possible to determine the HiRDB to which each unload log file belongs.
- Files other than unload log files must not be created in a directory for unload log files. If other types of files are created in such a directory, HiRDB may delete those files.

(d) Creating unload log files in a HiRDB file system area

To create unload log files in a HiRDB file system area, a HiRDB file system area must be created with the `pdfmkfs` command. Table 3-8 provides guidelines for the options that are specified in the `pdfmkfs` command.

Table 3-8: Guidelines for options to be specified in the pdfmkfs command (when unload log files are to be created in a HiRDB file system area)

pdfmkfs command option	Specification guideline
`-k`	Specify `UTL` as the usage objective.
`-n`	For the HiRDB file system area length, specify the value determined by the following formula: *total-number-of-records-in-system-log-files-to-be-unloaded* × *system-log-file-record-length* × *number-of-unload-log-files-to-be-created* × 1.2 ÷ 1048576
`-l`	For the maximum files count, specify the number of unload log files to be created.
`-e`	For the maximum increments count, specify the number of unload log files to be created × 10.

(2) Specification of the pd_log_auto_unload_path operand

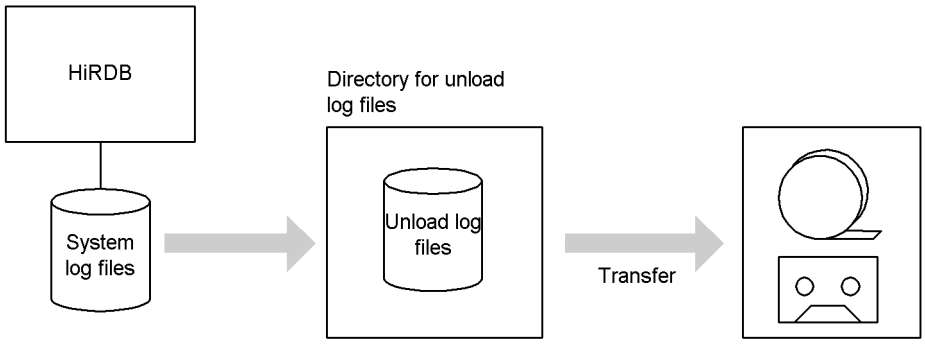
Specify in the `pd_log_auto_unload_path` operand the directories that have been created for unload log files.

When unload log files are to be created in an HiRDB file system area, specify the name of the HiRDB file system area.

3.8.3 Application example 1 (using a single directory for unload log files)

In this application example, a single directory for unload log files is provided.

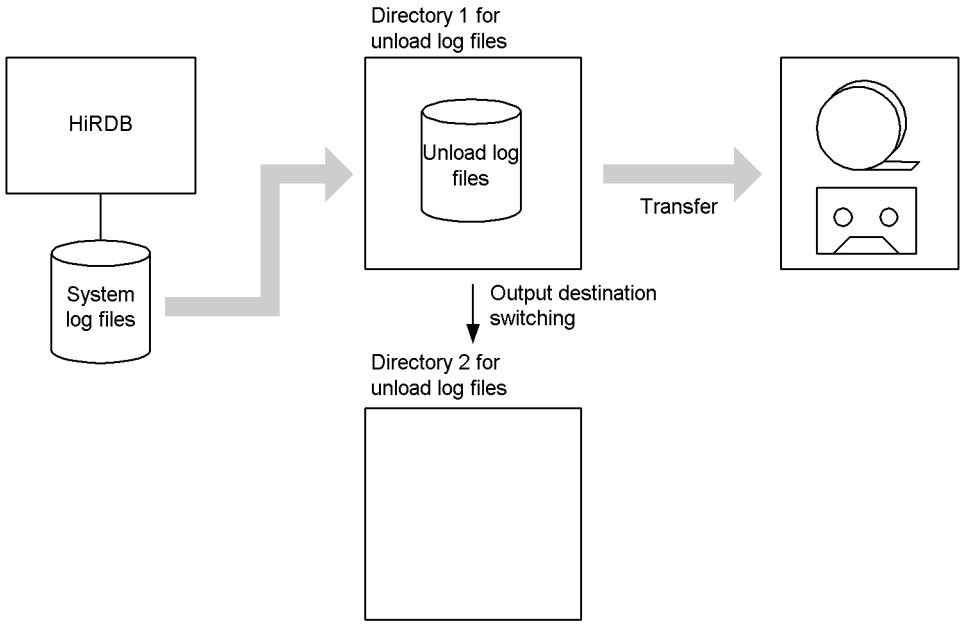
This directory for unload log files stores one week's worth of unload log files, so the unload log files are transferred to another medium once a week. Thus, unload log files are transferred to another medium on a regular basis.



3.8.4 Application example 2 (using two directories for unload log files)

In this application example, the automatic log unloading facility operates under the following conditions:

- Two directories for unload log files have been created.
- When the directories are switched because one of them has become full, the unload log files in the full directory are transferred to another medium.



(1) When a disk becomes full, the directory for unload log files is changed

When a disk becomes full, the destination directory is switched to Directory 2 for unload log files. When this occurs, the `KFPS01151-I` message is output:

```
KFPS01151-I bes1 changed auto log unload directory from /unlddir1/bes1
to /unlddir2/bes1.reason=1665
```

Explanation

The directory for unload log files has been switched from /`unlddir1/bes1` to /`unlddir2/bes1`.

(2) Transferring all unload log files in a directory to another medium

Move all the unload log files in `/unlddir1/bes1` onto another medium.

Repetitions of steps (1) and (2) are used to transfer unload log files to other media.

3.8.5 Application example 3 (making a backup)

In this application example, a backup is made. The procedure is the same regardless of the number of directories for unload log files that are used.

(1) Verifying that the automatic log unloading facility is operating

Enter the `pdlogatul` command to verify that the automatic log unloading facility is operating (in the case of a HiRDB/Parallel Server, you must perform this check at all servers for which the backup is to be made):

```
pdlogatul -d sys

HOSTNAME:host01(105321)
SERVER_NAME:sds1
AUTO_LOG_UNLOAD  NOW_UNLOAD_LOG_GROUP  CREATE_DIR
         ACTIVE               logfg02  /unlddir1/bes1
CURRENT LOG GENERATION INFO.
LOG_GROUP  GEN_NO. SERVER_RUN_ID RUN_ID    UNLOAD_FILE_NAME
logfg03        15   3a765d82      3a765d6d  sds1_3a765d820015_logfg03
```

Explanation

`AUTO_LOG_UNLOAD`

Indicates whether or not the automatic log unloading facility is operating. **`ACTIVE`** indicates that the facility is operating.

`NOW_UNLOAD_LOG_GROUP`

Shows the file group name of system log files that are being unloaded. **** is displayed if no files are being unloaded.

`CREATE_DIR`

Shows the name of the directory for unload log files that is currently being used. In this example, `/unlddir1/bes1` is being used.

`LOG_GROUP`

Shows the file group name of the current system log files.

`GEN_NO`

Shows the generation number of the current system log files.

`SERVER_RUN_ID`

Shows the server run ID of the current system log files.

`RUN_ID`

Shows the run ID of the current system log files.

`UNLOAD_FILE_NAME`

Shows the unload log file name when the current system log files are unloaded by the automatic log unloading facility.

(2) Swapping system log files

Before a backup is made, use the pdlogswap command to swap the system log files. In the case of a HiRDB/Parallel Server, swap the system log files on all servers for which the backup is to be made.

System log files are swapped in order to physically separate the system log information needed for database restoration. The system log files storing the system log information needed for database restoration are those that become primary from this point on.

```
pdlogswap -d sys -w
```

(3) Verifying that unloading of the system log has been completed

Enter the pdlogls command to check the status of the system log files (in the case of a HiRDB/Parallel Server, check the status of the system log files at all servers for which the backup is to be made):

```
pdlogls -d sys

HOSTNAME : host01(153027)
Group    Type Server   Gen No. Status  Run ID    Block No.
logfg01  sys  sds1           13 os----u 3a765d6d      eff      fc8
logfg02  sys  sds1           14 os----u 3a765d6d      fc9     1092
logfg03  sys  sds1           15 osu-b-u 3a765d6d     1093     115b
logfg04  sys  sds1           16 oc-d--u 3a765d6d     115c     1226
```

Explanation

- A file for which u is displayed in the third column under Status is a system log file waiting to be unloaded. When a file's unloading is completed, - is displayed. In this example, file logfg03 is waiting to be unloaded.
- Wait until there are no more system log files waiting to be unloaded (i.e., execute the pdlogls command until all system log files have been unloaded).

(4) Making a backup of all RDAREAs with the pdcopy command

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01
-z /pdcopy/logpoint01 -p /pdcopylist/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the referencing-permitted mode as the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up.

-b: Specifies the name of the backup file.

-z: Specifies the name of the log point information file.

-p: Specifies the file name for the pdcopy command's processing results listing.

Remarks

Specifying the -z option is recommended when the automatic log unloading facility is being used. When the -z option is specified, the following information is output in LOG FILE INFORMATION of the pdcopy command's processing results listing:

- Name of the server for which the backup was made (SERVER NAME)
- Server run ID when the backup was made (SERVER RUN ID)
- Generation number of the current system log files when the backup was made (GENERATION NO)
- File group name of the current system log files when the backup was made (FILE NAME)

This information makes it possible to identify the unload log file name for the current system log files when the backup was made. Based on this unload log file name, the unload log files required for database restoration are identified. For details on how to identify the unload log files required for database restoration, see *3.8.6 Creating a time series list of unload log files (identifying the unload log files required for database restoration)*.

Note that because the -z option cannot be specified in the cases listed below, you must in these cases perform the operation explained in (5) below to determine the unload log file name for the current system log files when a backup was made:

- HiRDB was started with the pdstart -r command
- Backup is not made on a server-by-server basis

(5) Checking the name of the unload log file for the current system log files

Use the pdlogatul command to check the name of the unload log file for the current system log files (in the case of a HiRDB/Parallel Server, perform this check for all servers for which backup is made):

```
pdlogatul -d sys

HOSTNAME:host01(105528)
SERVER_NAME:sds1
AUTO_LOG_UNLOAD  NOW_UNLOAD_LOG_GROUP  CREATE_DIR
         ACTIVE                  ****  /unlddir1/bes1
CURRENT LOG GENERATION INFO.
LOG_GROUP  GEN_NO.  SERVER_RUN_ID  RUN_ID    UNLOAD_FILE_NAME
logfg04        16   3a765d82       3a765d6d  sds1_3a765d820016_logfg04
```

Explanation

UNLOAD_FILE_NAME

Shows the unload log file name when the current system log files are unloaded by the automatic log unloading facility.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

3.8.6 Creating a time series list of unload log files (identifying the unload log files required for database restoration)

To restore a database, it is necessary to specify in the `pdrstr` command the unload log files in time series (according to generations). Therefore, before transferring the unload log files to another medium, create a time series list of unload log files. When the database is restored, refer to this list to identify the required unload log files.

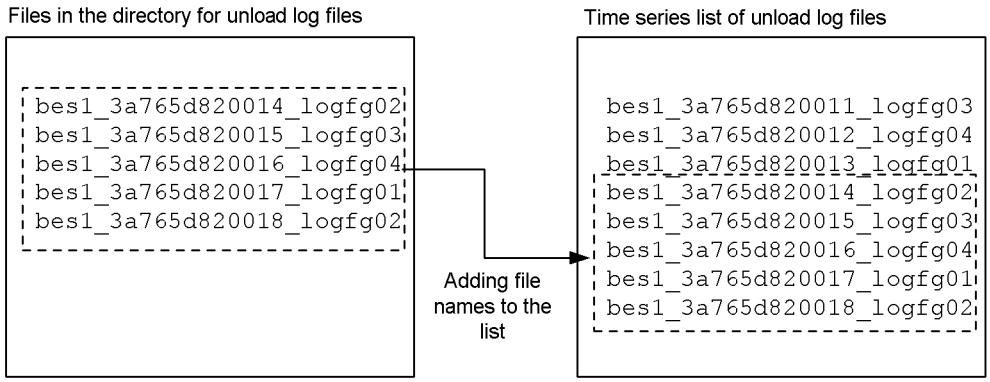
This section explains how to create a time series list of unload log files and how to identify the unload log files required for database restoration.

(1) When a single directory for unload log files is used

Because the names of the unload log files created by the automatic log unloading facility are in the format shown below, the unload log files in the directory for unload log files are arranged in time series order:
server-name_server-run-ID-and-generation-number_file-group-name

Figure 3-7 shows how to create a time series list of unload log files.

Figure 3-7: Creating a time series list of unload log files (using a single directory for unload log files)



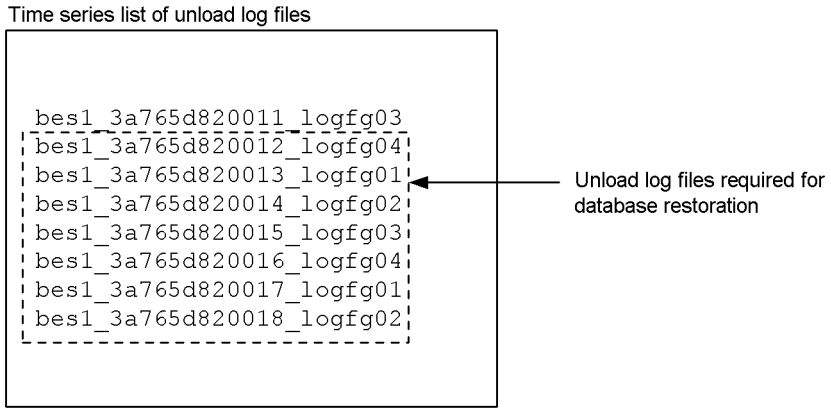
Explanation

The files in the directory for unload log files is added to the time series list of unload log files without modification.

Example (identifying the unload log files required for database restoration)

The following is assumed for this example: the server run ID of the system log files when the backup was made was `3a765d82`, the generation number was `12`, and the file group name was `logfg04`. The unload log files required for database restoration under these assumptions are identified from the time series list.

The time series list of unload log files is shown below:



Explanation

Because the server run ID of the system log files at the time the backup was made

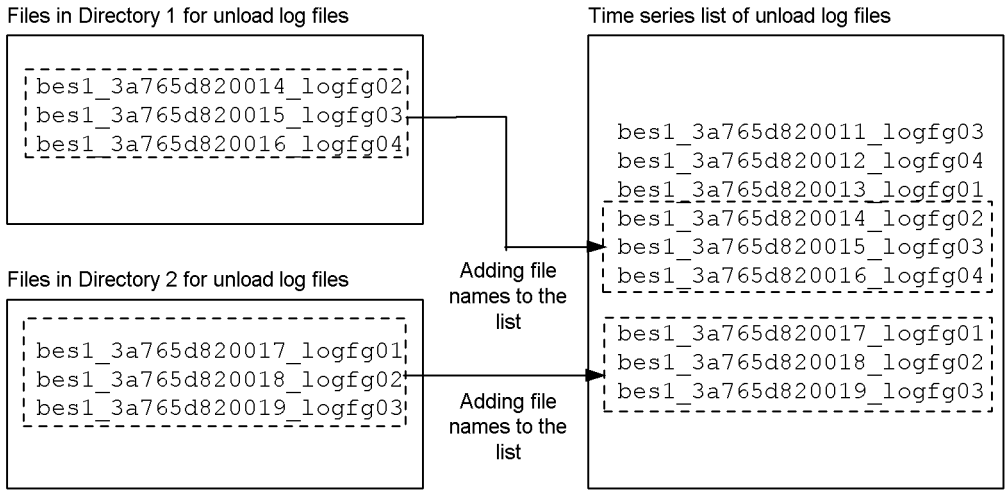
was 3a765d82, the generation number was 12, and the file group name was logfg04, the unload log file name is bes1_3a765d820012_logfg04. For database restoration, all unload log files beginning with this file will be required.

(2) When multiple directories for unload log files are used

Because the names of the unload log files created by the automatic log unloading facility are in the format shown below, the unload log files in the directory for unload log files are arranged in time series order:
server-name_server-run-ID-and-generation-number_file-group-name

Figure 3-8 shows how to create a time series list of unload log files.

Figure 3-8: Creating a time series list of unload log files (using multiple directories for unload log files)



Explanation

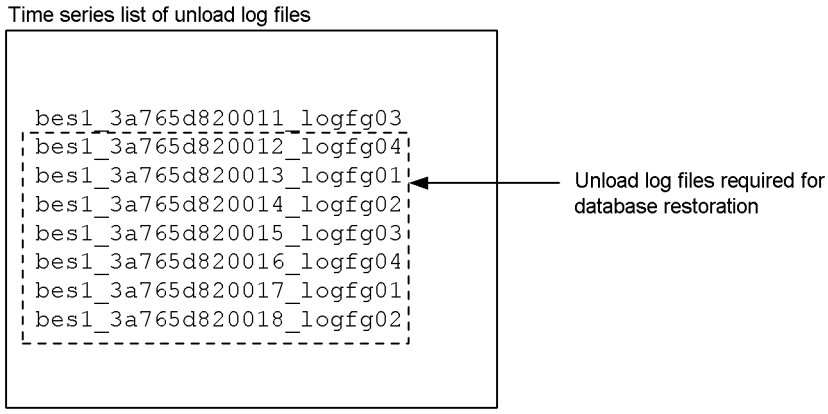
The files in Directories 1 and 2 for unload log files are added to the time series list of unload log files without modification.

Although this example shows Directory 1 followed by Directory 2 in the time series of unload log files, the reverse order (Directory 2 first, followed by Directory 1) is also possible. Which files are older in terms of generation can be determined by checking the names of the unload log files in the directories.

Example (identifying the unload log files required for database restoration)

The following is assumed for this example: the server run ID of the system log files when the backup was made was 3a765d82, the generation number was 12, and the file group name was logfg04. The unload log files required for database restoration under these assumptions are identified from the time series list.

The time series list of unload log files is shown below:



Explanation

Because the server run ID of the system log files at the time the backup was made was `3a765d82`, the generation number was `12`, and the file group name was `logfg04`, the unload log file name is `bes1_3a765d820012_logfg04`. For database restoration, all unload log files beginning with this file will be required.

(3) When unload log files are not arranged in time series order

Care must be taken when `0000` is shown as the generation number for a system log file. When generation numbers are in the range from ffff to 0000 under the same server run ID, the unload log files are not arranged in time series order. An example follows in which unload log files are not arranged in time series order.

Files in a directory for unload log files

```
bes1_3a765d820000_logfg04    "0000" is shown.
bes1_3a765d820001_logfg01
bes1_3a765d820002_logfg02
bes1_3a765d82fffe_logfg02
bes1_3a765d82ffff_logfg03
```

In this case, the unload log files were created in the sequence shown below, so they must be used in this sequence:

1. `bes1_3a765d82fffe_logfg02`
2. `bes1_3a765d82ffff_logfg03`
3. `bes1_3a765d820000_logfg04`
4. `bes1_3a765d820001_logfg01`

5. `bes1_3a765d820002_logfg02`

Remarks

The situation explained above may occur because HiRDB is running 24 hours a day and the server run ID does not change.

(4) Notes

You must check that there are no missing files in the time series list of unload log files. When the server run ID is the same, either the generation numbers increment by 1 or different file group names are used under the same generation number.

If this is not the case, an unload log file may be missing. If a required unload log file is missing, it is necessary to unload again the system log files for the missing information. If the system log files have already been overwritten and unload log files cannot be created, the database cannot be restored to the most recent synchronization point. In such a case, the database can be restored only to the point where the most recent backup was made.

3.8.7 Error handling

Table 3-9 shows the actions that can be taken when an error occurs while the automatic log unloading facility is being used.

Table 3-9: Handling of errors when the automatic log unloading facility is used

Error detail	Corrective action
An error has occurred on the disk containing a directory for unload log files.	Because the unload log files may become unavailable, stop operations and make backups. Make backups for the RDAREAs under all servers that are using the directory for unload log files in which the error occurred. Once the disk on which the error occurred has been restored, the automatic log unloading facility can be used again.
The disk containing a directory for unload log files has become full.	If a single directory for unload log files is being used, stop the automatic log unloading facility, transfer the unload log files in the directory to another medium, then use the `pdlogatul` command to start the automatic log unloading facility. If multiple directories for unload log files are being used, switch directories. If the disks for all directories become full, stop the automatic log unloading facility, transfer the unload log files in the directories to another medium, then use the `pdlogatul` command to start the automatic log unloading facility.

Error detail	Corrective action
The automatic log unloading facility has stopped (KFPS01150-E message is output)	First correct the error, then use the pdlogatul command to restart the automatic log unloading facility. If HiRDB is restarted, the automatic log unloading facility will be restarted automatically even if the pdlogatul command is not executed. It is possible that the error occurred because the unload log files cannot be used due to the fact that all directories for unload log files have reached one of the following statuses: • Disk full* • Disk error • Wrong permission • Nonexistent directory • Unload log files created during the previous operation of HiRDB are still present. If none of the above is applicable, notify maintenance personnel after collecting the following information: • Syslogfile • All files under $PDDIR/spool • pdlogls command's execution results • Backup of the HiRDB system definition file

* If the unloading process fails because of a disk space shortage, HiRDB deletes an incomplete unload log file. Because this file is deleted after the disk became full during unload log file creation, the disk may now appear to have available space.

3.8.8 Notes on HiRDB termination

If the pdstop command is executed while unload processing is underway, the normal or planned termination of HiRDB is carried out after the unload processing is completed. To terminate HiRDB normally immediately without waiting for completion of unload processing, enter the pdlogatul -t command to stop the automatic log unloading facility, then execute the pdstop command. In such a case, the system log files waiting to be unloaded will be unloaded automatically when HiRDB is started subsequently.

If HiRDB is terminated forcibly, unload processing is cancelled immediately.

3.9 Monitoring the free area for system log files

Executor: HiRDB administrator

This section describes how to monitor the free area for system log files. The following is described:

- What is monitoring of the free area for system log files?
- Environment setting
- HiRDB processing when the percentage of free area falls below the warning value
- Tasks the HiRDB administrator must perform when the percentage of free area falls below the warning value
- Notes
- Output of status information file for system log files

3.9.1 What is monitoring the free area for system log files?

(1) Function overview

While HiRDB operation is underway, database update logs are accumulated in system log files. When all system log files become full, no more database update logs can be output and the operation of HiRDB cannot continue; as a result, HiRDB terminates abnormally. To avoid a service interrupt due to abnormal termination of HiRDB, the HiRDB administrator should constantly monitor the amount of free space available for system log files.

HiRDB provides a facility for this purpose (monitoring the free area for system log files). When this facility is used, HiRDB monitors the percentage of free area in the system log files and issues a warning message or limits database usage according to a usage level specified by the HiRDB administrator. Either of the following levels can be selected for use by this facility:

Level 1

When the percentage of free area available for system log files falls below the warning value, the `KFPS01162-W` warning message is output.

Level 2

When the percentage of free area available for system log files falls below the warning value, all transactions at the server are forcibly terminated and scheduling of new transactions is suppressed, which reduces the output to the system log files; the `KFPS01160-E` message is output.

(2) Functional differences between levels 1 and 2

Table 3-10 shows the functional differences between levels 1 and 2.

Table 3-10: Functional differences between levels 1 and 2

Functional item	Level 1	Level 2
Monitors percentage of free area for system log files	Yes	Yes
Outputs error or warning message	Yes	Yes
Suppresses scheduling of new transactions	No	Yes
Forcibly terminates all transactions at the server	No	Yes
Acquires synchronization point	No	Yes

(3) Percentage of free area for system log files

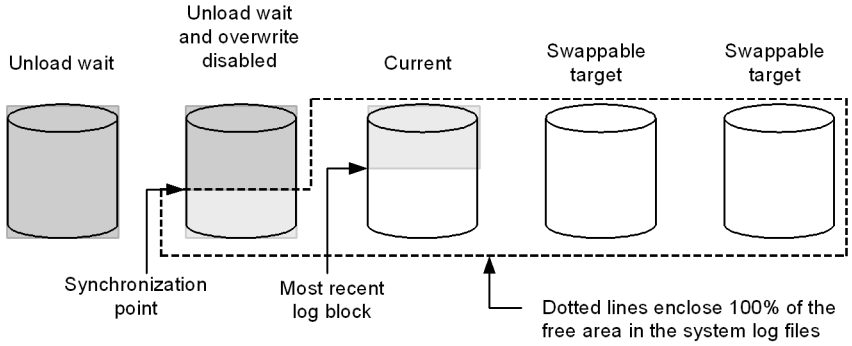
The following formula is used by HiRDB to determine the percentage of free area that is available for system log files:

Percentage (%) of free area for system log files =

Free area in system log files ÷ (free area in system log files + update log amount) × 100%

The free area in system log files is the sum of the available system log output area in the current file and all areas of log files that are in swappable target status. The update log amount is the system log information from a synchronization point to the most recent log block. Figure 3-9 illustrates the concept of free area in system log files.

Figure 3-9: Concept of free area in system log files



Explanation

- The lightly shaded area in files is the update log amount.

- The white area in files is the free area.
- The darkly shaded area in files is system logs that cannot be overwritten, and thus are not subject to rollback. Therefore, this area is not included in either the update log volume or the free area in the system log files.

(4) Percentage of free area for system log files at which the monitoring function activates (warning value)

The percentage of free area for system log files at which the function for monitoring the free area for system log files activates (warning value) is the value at which the database can be relied on to be safely recovered from a system failure (value at which no system log file shortage will occur during database recovery). The warning value depends on the type of server. Table 3-11 shows the percentage of free area for system log files at which the function for monitoring the free area for system log files activates (warning value).

Table 3-11: Percentage of free area for system log files at which the function for monitoring the free area activates (warning value)

Type of server		Percentage of free area for system log files at which the function for monitoring the free area activates (warning value)
HiRDB/Single Server		67%
HiRDB/Parallel Server	Front-end server	30%
	Dictionary server	67%
	Back-end server	

3.9.2 Environment setting

(1) Selecting a level

To reduce the possibility of insufficient free area for system log files causing HiRDB to terminate abnormally, Hitachi recommends that you set level 2. However, when you set level 2 and an inadequacy of free area for system log files develops, all transactions at the server will be forcibly terminated. For this reason, it is important to design the system log files carefully. For details about designing system log files, see the manual *HiRDB Version 8 Installation and Design Guide*.

(2) Setting the level

Use the `pd_log_remain_space_check` operand to specify the level you have selected. Level 1 is set when `warn` (the default value) is specified in this operand; level 2 is set when `safe` is specified.

3.9.3 HiRDB processing when the percentage of free area falls below the warning value

(1) Level 1

When the percentage of free area for system log files becomes less than the warning value, the `KFPS01162-W` warning message is output.

(2) Level 2

(a) Suppressing scheduling of new transactions

The `KFPS01160-E` message is output when scheduling of new transactions at the server is suppressed. Transactions occurring within an operation command or a utility are also suppressed. The `KFPA19703-E` message (with `reason=TRNPAUSE`) is output subsequently to the UAPs engaged in transaction processing at the server and an error is returned for each UAP. An operation command or utility terminates abnormally. Transactions can continue to be processed by other servers, but if a transaction branches from another server to the affected server, the `KFPA19703-E` message (with `reason=TRNPAUSE`) is output to the UAP and an error is returned.

(b) Forcibly terminating all transactions at the server

Transactions currently being processed at the affected server are forcibly terminated. The `KFPA11722-E` message (with `reason=SERVER PROCESS DOWN`) is output to the forcibly-terminated transactions and an error is returned. However, the transactions listed below are not subject to forced termination; in the case of these transactions, the `KFPS01163-W` message is output and HiRDB waits for the transactions' processing to terminate:

- Transactions engaged in commit or rollback processing
- Transactions waiting for determination of the second commit phase

- Transactions in the no-log mode

Even when a system log file error occurs during processing by a transaction other than those listed above, the transaction may not be forcibly terminated.

(c) Acquiring a synchronization point

Synchronization points are acquired for the number of times equal to the number of guaranteed-valid generations + 1. When a synchronization point cannot be immediately validated, HiRDB waits for validation of the synchronization point to be completed without skipping its acquisition. When a synchronization point is acquired, the percentage of free area for system log files increases. If an increasing percentage of free area reaches the warning value, scheduling of new transactions resumes and the `KFPS01161-I` message is output.

3.9.4 Tasks performed by the HiRDB administrator when the percentage of free area falls below the warning value

The HiRDB administrator must perform the tasks explained in this section.

(1) Identify the server affected by the free area insufficiency

Check the `KFPS01160-E` or `KFPS01162-W` message to identify the server that has insufficient free area. If an error occurred with the `pdlogcls` command, the server specified by this command is the affected server.

Handling level 2 limits

This server is in a status in which new transactions can no longer be scheduled. The HiRDB administrator can execute the `pdls -d svr` command to determine the server that is in this status. The `STATUS` entry in the command's execution results shows `TRNPAUSE` for the server that is in scheduling suppression status.

(2) Confirm the status of the system log files

Execute the `pdlogls` command on the system log files at the applicable server to determine their status.

(3) Place system log files in swappable target status

Perform the tasks listed below to increase the number of system log files in swappable target status:

- If there are reserved files, place them in swappable target status.
- If there are files in unload wait status, place them in unload completed status.
- If there are files in extracting status, place them in extraction completed status.
- If there are files in overwriting denied status for online reorganization, place them in overwriting permitted status for online reorganization.

(4) Determine the forcibly terminated transactions (only for level 2)

Refer to the KFPS00993-I message to check the transactions that were forcibly terminated. The REQUEST entry in this message shows log_remain_check for transactions that were forcibly terminated. Once the KFPS01161-I message is output, scheduling of new transactions has resumed, and the forcibly terminated transactions must be re-executed.

(5) When the KFPS01163-W message is output (only for level 2)

The KFPS01163-W message is output if there are transactions that are not subject to forced termination. HiRDB waits for the processing by these transactions to terminate. Do not use a command such as pdcancel to forcibly terminate these transactions, because this may cause error shutdowns in RDAREAs.

You can use the following procedure to identify the UAPs, operation commands, and utilities that issued transactions that continue to operate:

Procedure

1. Refer to the KFPS01160-E or KFPS01163-W message to identify the name of the affected server.
2. Use the pdls -d trn -a -s *server-name* command to acquire the required transaction information. In -s *server-name*, specify the server name identified in 1.
3. Refer to PROGRAM or C-PID in the command execution results to identify the UAP, operation command, or utility that issued a transaction.

(6) Identify the cause of insufficient area for system log files

Use the status information file for system log files to identify the cause of the area insufficiency for system log files. For details, see *3.9.6 Output of status information file for system log files*.

(7) Revise the design of the system log files

If the function for monitoring the free area for system log files activates more than once, there may be a defect in the design of the system log files (such as the number of files or the file size). In this case, you should revise the design of the system log files. For details about designing system log files, see the manual *HiRDB Version 8 Installation and Design Guide*.

(8) Ensure that the pdlogswap command is not executed

If the pdlogswap command is used to swap system log files while the percentage of free area is less than the warning value, HiRDB is more likely to terminate forcibly because there are no files in swappable target status. Therefore, you should not execute the pdlogswap command.

3.9.5 Notes

(1) Notes applicable to levels 1 and 2

- When the `pdlogswap` command is used to swap system log files, the percentage of free area for system log files decreases. If the number of system log files is small, care is required because the amount of log information increases dramatically.
- Thought should be given to the advisability of executing the `pdlogswap` command consecutively, because the percentage of free area for system log files decreases.
- When an error occurs in a swappable target file, the percentage of free area for system log files decreases. If the number of system log files is small, care is required because the percentage of free area decreases dramatically.
- If HiRDB terminates abnormally while it is restarting, the HiRDB restart must be performed again. Because availability of swappable target system log files is even more important in this circumstance, an area insufficiency for system log files may cause HiRDB to terminate abnormally. When you design your system log files, you must take into account abnormal situations such as this, and you must realize that using the function for monitoring the free area for system log files does not prevent insufficiencies from occurring.

(2) Notes applicable only to level 2

When the `pdlogcls` command is used to close a swappable target file, the percentage of free area for the system log files decreases. If this causes the percentage of free area to fall below the warning value, the specified file will not close. In such a case, the `KFPS01280-E` will be output (with reason code `712`) and the `pdlogcls` command will terminate abnormally.

3.9.6 Output of status information file for system log files

This section describes how to use the status information file for system log files to identify the cause of an area insufficiency for the system log files.

(1) Content of status information file for system log files

When the percentage of free area for the system log files falls below the warning value, a status information file for system log files is output. This file acquires the status of the system log files, synchronization points, and transactions. The status information file for system log files is created in the `$PDDIR/spool/pdjnlinf` directory. The file name is as shown below:

`pdsnap`.*server-name*.*time-issued*

Remarks

- If acquisition of the date or time information fails, the time issued becomes

999999999999.

- If a file with the same name already exists, that file is overwritten.
- If you change the file name, it may not be possible to use the pdcspool command to delete the file.

Output contents of status information file for system log files

The following example lists the output contents of a status information file for system log files. In this example, the percentage of free area at back-end server bes1 is below the warning value.

```
pdsnap.bes1.030415124350    <- File name[1]
Date:2003/04/15             <- Date issued[1]
Time:12:43:50               <- Time issued[1]
pdlogls -d sys -e -s bes1   <- Command issued internally by HiRDB
 Execution result of the pdlogls command
       :

pdlogls -d spd -e -s bes1
 Execution result of the pdlogls command
       :

pdls -d trn -a -s bes1
 Execution result of the pdls command
       :

pdls -d svr
 Execution result of the pdls command[2]
       :
```

[1] If acquisition of the date or time failed, 99...99 is displayed. Change the file name as necessary.

[2] In the case of a HiRDB/Parallel Server, only information for units at the applicable server is displayed.

(2) How to identify the cause of an area insufficiency for the system log files

Use the status information file for system log files to identify the cause of the area insufficiency for the system log files. The method of identifying the cause is described below.

(a) Identify the status information file for the system log files

Identify the status information file for the system log files from the KFPS01160-E or KFPS01162-W message.

Example: When the KFPS01160-E message is output

```
KFPS01160-E HRDB Insufficient system log space. Transaction
```

```
service stopped. Transactions terminate by force, server =
bes1, output file name = pdsnap.bes1.030401084500
```

The following is the name of the status information file for the system log files in this case:

```
$PDDIR/spool/pdjnlinf/pdsnap.bes1.030401084500
```

(b) Acquire the status of the system log files

Refer to the contents of the status information file for the system log files and check the status of the system log files. The `STATUS` entry in the output results of the `pdlogls -d sys` command shows the status of the system log files.

(c) Causes of area insufficiency for system log files

The cause can be inferred from the status of the system log files. Table 3-12 describes the causes of free area insufficiencies in system log files.

Table 3-12: Causes of free area insufficiencies for system log files

Status of system log files	Causes of insufficient free area for system log files
Many files are in unload wait status	System log files may not have been unloaded. You must unload system log files regularly.
Many files are in overwrite disabled status	One of the following causes may be applicable: • When numerous updates are performed within a single transaction, the update log size increases and the percentage of free area for system log files decreases. You should split up such transactions into multiple transactions. If this is not possible, increase the available area for system log files. • If a transaction is not settled for a long period of time, no synchronization point will be acquired. During such a period, the update log size increases and the percentage of free area for system log files decreases. If there are such transactions, either revise the server's maximum wait time (value of the `PDSWAITTIME` operand in the client environment definition or value of the `pd_watch_pc_client_time` operand) or use the skipped effective synchronization point dump monitoring facility.
Many files are in extracting status	Extraction processing may not have been applied to the system log files. Either revise the design of the system log files or revise the operation method for data linkage that uses the HiRDB Datareplicator.
Many files are in overwriting denied status for online reorganization	Update processing of updatable online reorganization may not have been applied. Either revise the design of the system log files or revise the method of operating updatable online reorganization of the inner replica facility.

Chapter

4. Handling Synchronization Point Dump Files

This chapter describes the procedures for handling the synchronization point dump files.

This chapter contains the following sections:

4.1 Basics

This section provides an overview of the synchronization point dump files. The user must understand synchronization point dump files before actually handling them.

(1) What are synchronization point dump files?

If HiRDB terminates abnormally and recovery processing must be performed with only the system logs, all system logs from the time of HiRDB startup will be required and system recovery will take a long time. By establishing points at regular intervals while HiRDB is operating and saving the HiRDB management information needed for recovery at each such point, the system logs prior to each point will no longer be needed and system recovery time will be shortened. Such a point is called a *synchronization point*. The HiRDB management information acquired at a synchronization point is called a *synchronization point dump*. The file that stores a synchronization point dump is called a *synchronization point dump file*.

HiRDB applies to a database at a given synchronization point all the database updates since either the previous synchronization point or since HiRDB startup. It is the responsibility of the HiRDB administrator to create synchronization point dump files before errors occur.

(2) Status of synchronization point dump files

HiRDB uses the statuses shown in Table 4-1 to manage the synchronization point dump files.

Table 4-1: Synchronization point dump file statuses

Status	Description
Writing	File is currently being used for output of a synchronization point dump.
Overwrite enabled	File is overwritable because it does not contain a synchronization point dump that would be needed for system recovery. A file in this status is not a guaranteed valid generation.
Overwrite disabled	File is not overwritable because it contains a synchronization point dump that would be needed for system recovery. A file in this status is a guaranteed valid generation
Reserved	File is closed and is not subject to output of synchronization point dumps. When a synchronization point dump file name has been specified in the HiRDB system definition but no corresponding synchronization point dump file has been created, that file is also placed in reserved status.

(3) Status of synchronization point dump files during HiRDB startup

When HiRDB is started, all the synchronization point dump files for which `ONL` was specified in the `pdlogadfg -d spd` operand are opened and placed in overwrite

enabled status.

A file that cannot be opened or for which ONL was not specified is placed in reserved status.

(4) Synchronization point dump file status changes

When a synchronization point dump is acquired and validated,* the status of the synchronization point dump file changes. A synchronization point dump is collected at the following times:

1. When server startup or restart processing is completed
2. When server termination preparation processing is completed
3. When system log files are swapped
4. When the number of blocks of system log information specified in the pd_log_sdinterval operand have been collected since the last synchronization point dump acquisition
5. When the amount of time specified in the pd_log_sdinterval operand has elapsed since the last synchronization point dump acquisition
6. When the pdlogsync command is executed

* When a synchronization point dump is validated, the KFPS02183-I message is output. However, if N is specified in the pd_spd_assurance_msg operand, the KFPS02183-I message is not output.

(5) Relationship between number of guaranteed valid generations and file status

HiRDB stores one synchronization point dump generation in one synchronization point dump file. When all the provided files are filled with synchronization point dumps, new data is written over the existing data in the first file.

HiRDB places the file containing the previous synchronization point dump generation in overwrite disabled status. If the number of guaranteed valid generations is set to 2, HiRDB places the files containing the last two generations in overwrite disabled status. Figures 4-1 and 4-2 show the changes in file status that result when a synchronization point dump is output.

Figure 4-1: Changes in file status when a synchronization point dump is output (number of guaranteed valid generations = 1)

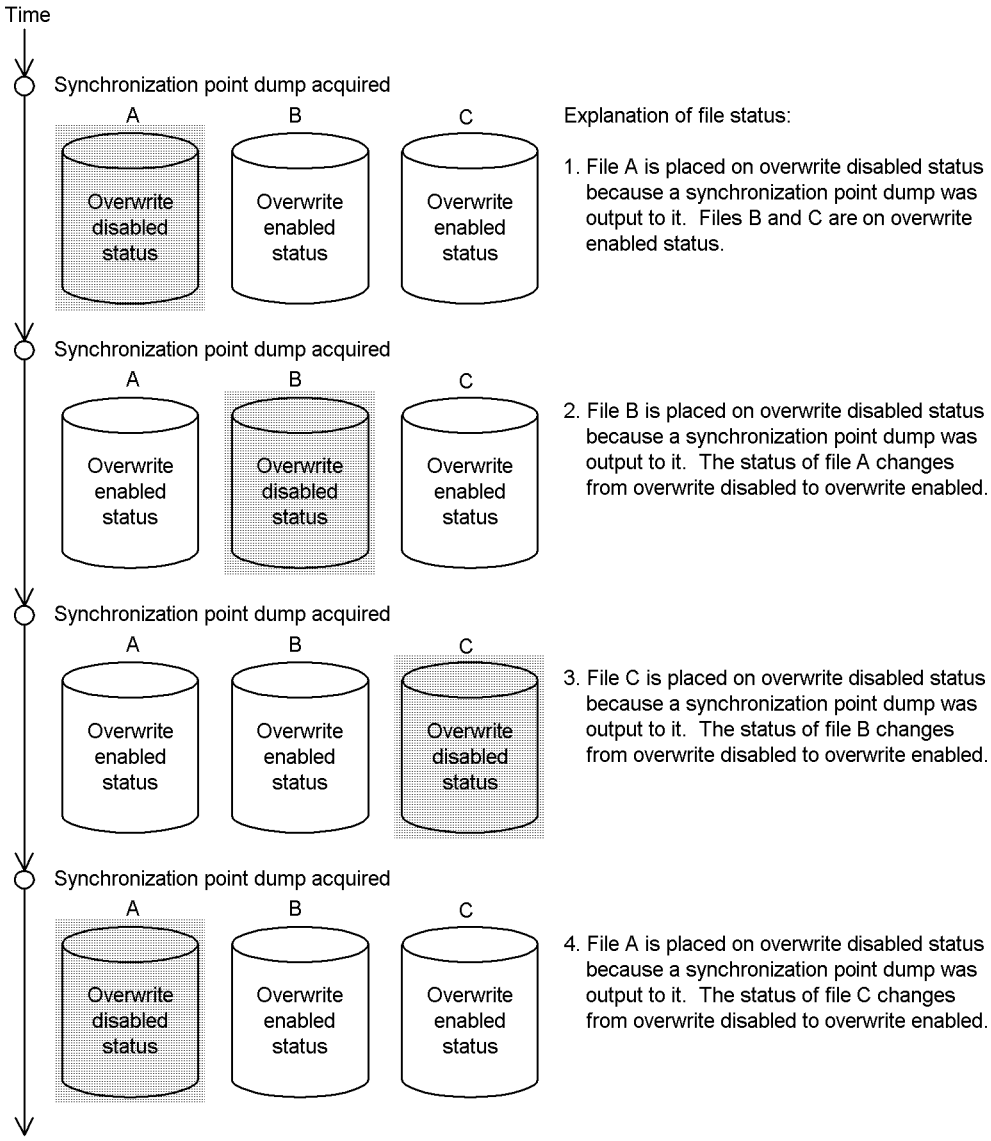
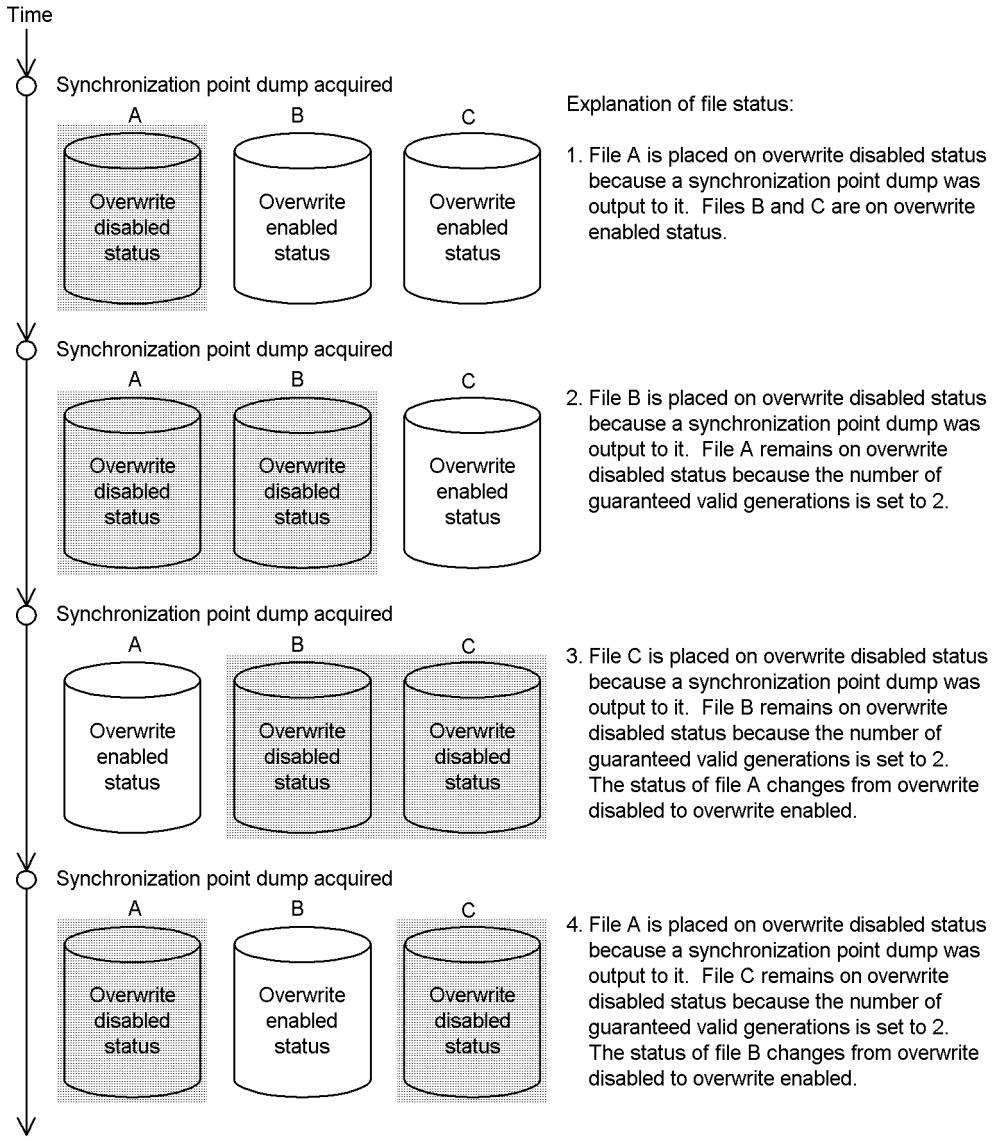


Figure 4-2: Changes in file status when a synchronization point dump is output (number of guaranteed-valid generations = 2)



(6) Commands used to manipulate synchronization point dump files

Table 4-2 lists the commands that are provided for manipulating synchronization point dump files.

Table 4-2: Commands used to manipulate synchronization point dump files

Command	Description
pdloginit	Initializes a synchronization point dump file.
pdlogadpf	Allocates a synchronization point dump file to a file group specified in the HiRDB system definition.
pdlogopen	Opens a synchronization point dump file; also places an overwrite enabled file in reserved status.
pdlogcls	Closes a synchronization point dump file; also places a reserved file in overwrite enabled status.
pdlogls	Displays information about synchronization point dump files.
pdlogrm	Deletes a synchronization point dump file.
pdlogsync	Acquires a synchronization point dump.

4.2 Setting the synchronization point dump interval

Executor: HiRDB administrator

(1) Setting the interval using the pd_log_sdinterval operand

The `pd_log_sdinterval` operand is used to set the synchronization point dump interval. The following considerations are taken into account in order to set the synchronization point dump interval:

- **Amount of system log information that is output**

  A synchronization point dump can be acquired each time the number of blocks of system log information that have been output since the last synchronization point dump was validated equals a value specified in the `pd_log_sdinterval` operand.

- **Elapsed time**

  A synchronization point dump can be acquired when the amount of time that has elapsed since the last synchronization point dump was validated equals a value specified in the `pd_log_sdinterval` operand.

(2) Guidelines for determining the interval

When the synchronization point dump interval is short, the time required for database recovery during a HiRDB restart is reduced. However, online performance may be degraded because synchronization point dumps are acquired more frequently.

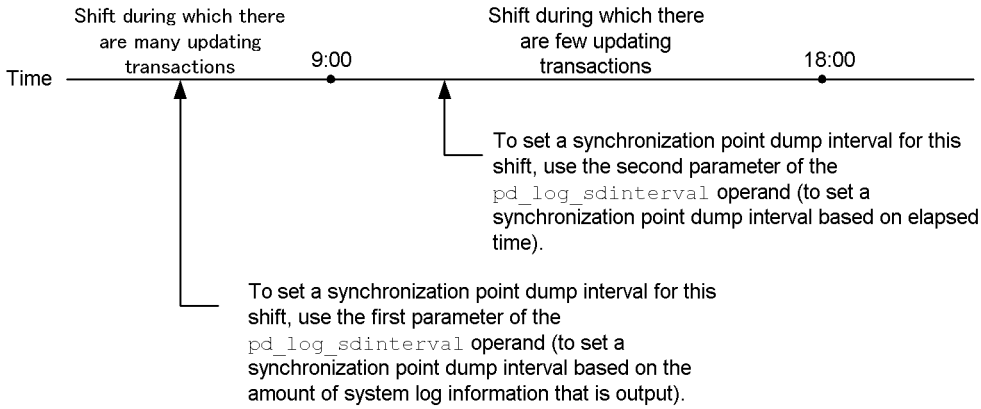
On the other hand, when the synchronization point dump interval is long, the time required for database recovery during a HiRDB restart is increased. However, online performance may be improved because synchronization point dumps are acquired less frequently.

The synchronization point dump interval can be determined by checking the item labeled `SYNC POINT GET INTERVAL` in the statistics analysis utility's system activity statistical information.

For details on tuning the synchronization point dump interval, see *21.4 Tuning the synchronization point dump interval*.

(3) Example of setting a synchronization point dump interval

This example assumes a system that has few updating transactions during the day shift (0900 to 1800 hours) and many updating transactions at night (1800 to 0900 hours):



Explanation

- During the shift in which there are few updating transactions (0900 to 1800 hours), obtain synchronization point dumps based on elapsed time.
- During the shift in which there are many updating transactions (1800 to 0900 hours), obtain synchronization point dumps based on the amount of system log information that is output.
- If additional synchronization point dumps are needed, use the `pdlogsync` command as necessary.

4.3 Procedures for manipulating synchronization point dump files

Executor: HiRDB administrator

This section describes the procedures for manipulating synchronization point dump files. The following topics are covered:

1. When the status of a synchronization point dump file has changed
2. When there are no overwrite enabled files
3. Increasing (or reducing) the synchronization point dump file size during HiRDB operation
4. Changing the file status
5. Adding a new synchronization point dump file
6. Deleting a synchronization point dump file
7. Obtaining the system log file corresponding to a synchronization point dump file
8. Increasing the number of synchronization point dump file guaranteed valid generations

Frequently asked questions about the handling of synchronization point dump files are answered in Q&A format in *A.2 Synchronization point dump files*.

4.3.1 When the status of a synchronization point dump file has changed

The HiRDB administrator must use the `pdlogls` command to check that there is an overwrite enabled file. If there is no overwrite enabled file, a reserved file should be placed in overwrite enabled status. If a synchronization point dump needs to be output, but there are no overwrite enabled files, HiRDB (or the unit for a HiRDB/Parallel Server) terminates abnormally. When HiRDB runs out of overwrite enabled files, it outputs a message to that effect to the message log file.

4.3.2 When there are no overwrite enabled files

Overwriting of a reserved file must be enabled by means of one of the methods described below. If there are no reserved files, you must add synchronization point dump files. For details, see *4.3.5 Adding a new synchronization point dump file*.

(1) When there is a synchronization point dump file entity

There is a synchronization point dump file entity when a synchronization point dump file has been created by the `pdloginit` command.

Procedure

1. Use the pdlogls command to check the reserved files.

   ```
   pdlogls -d spd -s b001
   ```

2. Use the pdlogopen command to enable overwriting of a reserved file that has an entity.

   ```
   pdlogopen -d spd -s b001-g syncfg01
   ```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

(2) When there is no synchronization point dump file entity

There is no synchronization point dump file entity when a file group name has been specified in the pdlogadfg operand but no synchronization point dump file has been created by specifying the pdloginit command.

Procedure

1. Use the pdlogls command to check the reserved files.

   ```
   pdlogls -d spd -s b001
   ```

2. Use the pdloginit command to create a synchronization point dump file. Assign to the created synchronization point dump file a synchronization point dump file name specified in the HiRDB system definition.

   ```
   pdloginit -d spd -s b001 -f /sysfile01/sync01 -n 5000
   ```

3. Use the pdlogadpf command to associate the synchronization point dump file created in step 2 with the file group specified by the pdlogadfg operand.

   ```
   pdlogadpf -d spd -s b001 -g syncfg01 -a /sysfile01/sync01
   ```

4. Use the pdlogopen command to enable overwriting of the synchronization point dump file created in step 2.

   ```
   pdlogopen -d spd -s b001 -g syncfg01
   ```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

4.3.3 Increasing (or reducing) the synchronization point dump file size during HiRDB operation

If a space shortage occurs in a synchronization point dump file, the following procedure can be used during HiRDB operation to increase the synchronization point dump file size:

Procedure

1. Use the pdlogls command to check the status of the synchronization point dump files.

   ```
   pdlogls -d spd -s b001
   ```

2. Use the pdlogcls command to make an overwrite enabled file into a reserved file.

   ```
   pdlogcls -d spd -s b001 -g syncfg01
   ```

3. Use the pdlogrm command to delete the reserved file.

   ```
   pdlogrm -d spd -s b001 -f /sysfile01/sync01
   ```

4. Use the pdloginit command to re-create the synchronization point dump file deleted in step 3.

   Then, increase the record count to a value greater than the record count for the synchronization point dump file before the change. To decrease the file size, decrease the record count.

   ```
   pdloginit -d spd -s b001 -f /sysfile01/sync01 -n 5000
   ```

5. Use the pdlogopen command to enable overwriting of the synchronization point dump file created in step 4.

   ```
   pdlogopen -d spd -s b001 -g syncfg01
   ```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

Note

When you change overwrite enabled files into reserved files, make sure that you do not change all overwrite enabled files into reserved files. If a synchronization point dump needs to be output while there are no overwrite enabled files, HiRDB (or the unit for a HiRDB/Parallel Server) will terminate abnormally. Therefore, it is important to leave at least one overwrite enabled file when you change the size of a synchronization point dump file.

4.3.4 Changing the file status

The pdlogopen command is used to place a reserved file in overwrite enabled status. The pdlogcls command is used to make an overwrite enabled file into a reserved file.

4.3.5 Adding a new synchronization point dump file

This section explains the procedure for adding a new synchronization point dump file. Before a synchronization point dump file is added, it is recommended that the synchronization point dump file design guidelines be reviewed; for details, see the

manual *HiRDB Version 8 Installation and Design Guide*.

(1) When HiRDB can be terminated normally

Procedure

1. Use the `pdfstatfs` command to check for free space in the HiRDB file system area in which the synchronization point dump file will be created.

   ```
   pdfstatfs /sysfile01
   ```

   If there is no free space, create a new HiRDB file system area. For details, see *10.2 Creating (initializing) a HiRDB file system area*.

2. Use the `pdstop` command to terminate HiRDB normally.
3. Add one of the operands listed below, as appropriate, to the HiRDB system definition. Specify in this operand the synchronization point dump file to be added in step 4.

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

4. Use the `pdloginit` command to add (initialize) the synchronization point dump file.

   ```
   pdloginit -d spd -s b001 -f /sysfile01/sync01 -n 5000
   ```

5. Use the `pdconfchk` command to check the HiRDB system definition. If an error is detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
6. Use the `pdstart` command to terminate HiRDB normally.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

(2) When HiRDB cannot be terminated normally

Use the `pdchgconf` system reconfiguration command to change the HiRDB system definition. Note that HiRDB Advanced High Availability must have been installed in order to use this command.

Procedure

1. Use the `pdfstatfs` command to check for free space in the HiRDB file system area in which the synchronization point dump file will be created.

   ```
   pdfstatfs /sysfile01
   ```

   If there is no free space, create a new HiRDB file system area. For details, see *10.2 Creating (initializing) a HiRDB file system area*.

2. Create the `$PDDIR/conf/chgconf` directory.
3. Copy the HiRDB system definition file currently being used to the directory created in step 2.
4. Add one of the operands listed below, as appropriate, to the HiRDB system definition in the `$PDDIR/conf/chgconf` directory. Specify in this operand the synchronization point dump file to be added in step 5.
   - `pdlogadfg -d spd` operand
   - `pdlogadpf -d spd` operand
5. Use the `pdloginit` command to add (initialize) the synchronization point dump file.

   ```
   pdloginit -d spd -s b001 -f /sysfile01/sync01 -n 5000
   ```
6. Use the `pdconfchk` command to check the HiRDB system definition in the `$PDDIR/conf/chgconf` directory. If an error is detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
7. Use the `pdchgconf` command to replace the existing HiRDB system definition with the modified system definition.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

4.3.6 Deleting a synchronization point dump file

The `pdlogrm` command is used to delete a reserved synchronization point dump file. Only a reserved synchronization point dump file can be deleted.

Delete the applicable operand listed below. You delete the operand that corresponds to the synchronization point dump file deleted with the `pdlogrm` command.

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

If this operand is not deleted, the deleted synchronization point dump file will become a virtual file.

4.3.7 Obtaining the system log file corresponding to a synchronization point dump in file

The `pdlogls` command is used to obtain the system log file that corresponds to the synchronization point dump in a synchronization point dump file.

4.3.8 Increasing the number of synchronization point dump file guaranteed-valid generations

The procedure for increasing the number of guaranteed-valid generations for synchronization point dump files is described below.

(1) When HiRDB can be terminated normally

Procedure

1. Use the `pdlogls` command to check the number of synchronization point dump files. Confirm that the number of files is at least one more than the number of guaranteed-valid generations after the change. If there are not enough synchronization point dump files, you must add files.

   ```
   pdlogls -d spd -s b001
   ```

2. Use the `pdstop` command to terminate HiRDB normally.
3. Add one of the operands listed below, as appropriate, to the HiRDB system definition. Specify in this operand the synchronization point dump file to be added in step 4.

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

   Also, specify in the `pd_spd_assurance_count` operand the number of guaranteed-valid generations.
4. Use the `pdloginit` command to add (initialize) the synchronization point dump file.

   ```
   pdloginit -d spd -s b001 -f /sysfile01/sync01 -n 5000
   ```

5. Use the `pdconfchk` command to check the HiRDB system definition. If an error is detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
6. Use the `pdstart` command to start HiRDB normally.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

(2) When HiRDB cannot be terminated normally

Use the `pdchgconf` system reconfiguration command to change the HiRDB system definition. Note that HiRDB Advanced High Availability must have been installed in order to use this command.

Procedure

1. Use the pdlogls command to check the number of synchronization point dump files. Confirm that the number of files is at least one more than the number of guaranteed-valid generations after the change. If there are not enough synchronization point dump files, you must add files.

   ```
   pdlogls -d spd -s b001
   ```

2. Create the $PDDIR/conf/chgconf directory.
3. Copy the HiRDB system definition file currently being used to the directory created in step 2.
4. Add one of the operands listed below, as appropriate, to the HiRDB system definition in the $PDDIR/conf/chgconf directory. Specify in this operand the synchronization point dump file to be added in step 5.

   - pdlogadfg -d spd operand
   - pdlogadpf -d spd operand

   Also, specify in the pd_spd_assurance_count operand the number of guaranteed-valid generations.
5. Use the pdloginit command to add (initialize) the synchronization point dump file.

   ```
   pdloginit -d spd -s b001 -f /sysfile01/sync01 -n 5000
   ```

6. Use the pdconfchk command to check the HiRDB system definition in the $PDDIR/conf/chgconf directory. If an error is detected, correct the HiRDB system definition and re-execute the pdconfchk command.
7. Use the pdchgconf command to replace the existing HiRDB system definition with the modified system definition.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

4.4 Status changes of synchronization point dump files

Table 4-3 and Table 4-4 show the synchronization point dump file status changes during HiRDB operation.

Table 4-3: Status changes of synchronization point dump files while HiRDB is operating (when synchronization point dump files are not duplicated)

Event	Status of synchronization point dump file			
	Open			Closed
	Overwrite enabled	Now writing	Overwrite disabled	Reserved
	1	2	3	4
Synchronization point dump is being output	→2	—	—	—
Synchronization point dump was validated	—	→3	→1*	—
Synchronization point dump output error occurred	—	→4	—	—
`pdlogcls` command was executed*	→4	—	—	—
`pdlogopen` command was executed*	—	—	—	→1

Legend:

—: Does not apply, or status does not change.

→*n*: Status to which synchronization dump file changes after occurrence of the event.

For example, →1 indicates that after the event occurs, the status of the synchronization dump file changes to overwrite enabled (status No. 1).

Note: The status transitions listed in this table assume that the relevant events were processed normally.

* Among files in overwrite disabled status, the indicated status changes are from the oldest generation.

Table 4-4: Status changes of synchronization point dump files while HiRDB is operating (when synchronization point dump files are duplicated)

Event		Status of system log files									
		Overwrite enabled			Now writing			Overwrite disabled			Rsrvd
		Both open	Sys A open Sys B clsd	Sys A clsd Sys B open	Both open	Sys A open Sys B clsd	Sys A clsd Sys B open	Both open	Sys A open Sys B clsd	Sys A clsd Sys B open	Both clsd
		1	2	3	4	5	6	7	8	9	10
A synchronization point dump was just acquired		→4	→5	→6	—	—	—	—	—	—	—
Synchronization point dump was validated		—	—	—	→7	→8	→9	→1*	→2*	→3*	—
Output error occurred in A system file		—	—	—	→6	→10	—	—	—	—	—
Output error occurred in B system file		—	—	—	→5	—	→10	—	—	—	—
`pdlog cls` executd (with `-a` or `-b`)	Omitted	→10	→10	→10	—	—	—	—	—	—	—
	`-a`	→3	→10	—	—	—	—	—	—	—	—
	`-b`	→2	—	→10	—	—	—	—	—	—	—
`pdlog open` executd (with `-a` or `-b`)	Omitted	—	→1	→1	—	—	—	—	→7	→7	→1
	`-a`	—	—	→1	—	—	—	—	—	→7	→2
	`-b`	—	→1	—	—	—	—	—	→7	—	→3

Legend:

—: Does not apply, or status does not change.

→*n*: Status to which synchronization dump file changes after occurrence of the event.

For example, →1 indicates that after the event occurs, the status of the synchronization dump file changes to overwrite enabled (status No. 1).

Note: The status transitions listed in this table assume that the relevant events were processed normally.

* Among files in overwrite disabled status, the indicated status changes are from the oldest generation.

Chapter

5. Handling Status Files

This chapter describes the procedures for handling the status files.

This chapter contains the following sections:

5.1 Basics

Information needed for a HiRDB restart is stored in status files. This section provides an overview of the status files. The user must understand status files before actually handling them.

(1) Status of status files

HiRDB uses the status shown in Table 5-1 to manage the status files.

Table 5-1: Status file statuses

Status	Description
Current	File is currently being used for output of system status information.
Spare	File is not currently being used for output of system status information, but it can be swapped with the current file if the current file becomes unavailable due to an I/O error. This file is in open status.
Reserved	File is closed and is not currently subject to output of system status information. A file is also placed in reserved status when its name has been specified in the HiRDB system definition but no status file has actually been created.
Shutdown	File has been shut down due to an error.

(2) Status of status files during HiRDB startup

When HiRDB is started normally, the first status file specified in the following system definition operands becomes the current file; all other status files become spare files:

- `pd_syssts_file_name_1 to pd_syssts_`*`file_name_7`* (unit status files)
- `pd_sts_file_name_1 to pd_sts_file_`*`name_7`* (server status files)

Notes

- A file resulting in an open error is placed in reserved status. A file that cannot be opened due to an error is placed in shutdown status.
- When HiRDB is restarted, the file that was being used as the current file in the previous session is inherited.

(3) Status file status changes

When status files are swapped, the status of the status files changes. Status files are swapped at the following times:

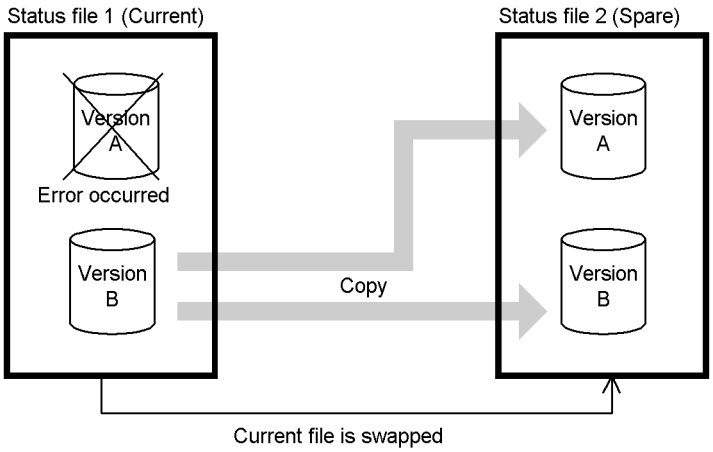
- When an error occurs in the current status file
- When a space shortage occurs in the current status file

- When the pdstsswap command is executed

(4) Status file swapping

If an error occurs in either of the file versions, A or B, of the current file, HiRDB copies the contents of the normal file version into a spare file (the same information is contained in both versions A and B of the current file). HiRDB then places the spare file in current status and places in shutdown status the file that was being used as the current file. This processing is called *status file swapping*. If there is no spare file when HiRDB needs to perform status file swapping, the unit terminates abnormally. Figure 5-1 shows status file swapping.

Figure 5-1: Status file swapping



(5) Commands used to manipulate status files

Table 5-2 lists the commands that are provided for manipulating status files.

Table 5-2: Commands used to manipulate status files

Command name	Description
pdstsinit	Initializes a status file.
pdstsopen	• Opens a status file; also places a reserved file in spare status. • Opens an initialized status file.
pdstscls	Closes a status file; also places a spare file in reserved status.
pdstsswap	Swaps status files and places the former current file in spare status.
pdstsrm	Deletes a status file.

5.2 Procedures for manipulating status files

Executor: HiRDB administrator

This section describes the procedures for manipulating status files. The following topics are covered:

1. When status files are swapped
2. When there are no spare files
3. Increasing (or reducing) the status file size during HiRDB operation
4. Changing the file status
5. Changing the current file
6. Adding a new status file
7. Deleting a status file
8. Checking the information in a status file

Frequently asked questions about the handling of status files are answered in Q&A format in *A.3 Status files*.

5.2.1 When status files are swapped

When status files are swapped, the HiRDB administrator must use the `pdls -d sts` command to check the following:

- If there is a spare file
- File record utilization factor

(1) Checking to see if there is a spare file

The HiRDB administrator must check that there is a spare file. If there is no spare file, a reserved file must be placed in spare status. *If an error occurs in the current status file while there is no spare file, either the status file is placed in the single-operation mode or the unit terminates abnormally.*

(2) Checking the file record utilization factor

If a space shortage occurs in the current status file, HiRDB selects a spare file whose size is greater than the current file and swaps the files. If there is no spare file that is larger than the current file, the unit terminates abnormally. When the largest file is being used as the current file, its record utilization factor should be checked.

5.2.2 When there are no spare files

Use one of the following methods to place a reserved file in spare status. If there are

no reserved files, you must add status files. For details, see *5.2.6 Adding a new status file*.

(1) When there is a status file entity

There is a status file entity when a status file has been created by the `pdstsinit` command.

Procedure

1. Use the `pdls` command to check the reserved files with entities.

   ```
   pdls -d sts -s b001
   ```

2. Use the `pdstsopen` command to place a reserved file in spare status.

   ```
   pdstsopen -s b001 -n sstsfg01
   ```

   It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

(2) When there is no status file entity

There is no status file entity when a status file has been specified in the HiRDB system definition but no status file has been created by specifying the `pdstsinit` command.

Procedure

1. Use the `pdls` command to check the virtual reserved files. Such a file's status is displayed as `NONE`.

   ```
   pdls -d sts -s b001
   ```

2. Use the `pdstsinit` command to create status files. Assign to the created status files status file names specified in the HiRDB system definition.

   ```
   pdstsinit -s b001 -f /sysfile01/ssts1a -l 4096 -c 1000
   pdstsinit -s b001 -f /sysfile01/ssts1b -l 4096 -c 1000
   ```

3. Use the `pdstsopen` command to open the status file created in step 2.

   ```
   pdstsopen -s b001 -n sstsfg01
   ```

   It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

5.2.3 Increasing (or reducing) the status file size during HiRDB operation

If the record utilization factor is high, the following procedure can be used to increase the status file size:

Procedure

1. Use the `pdls` command to check the following:
   - Spare status files (files with status displayed as `STANDBY`)
   - Record count in the status files

   ```
   pdls -d sts -s b001
   ```
2. Use the `pdstscls` command to make spare files into reserved files.

   ```
   pdstscls -s b001 -n sstsfg01
   ```
3. Use the `pdstsrm` command to delete the reserved files.

   ```
   pdstsrm -s b001 -f /sysfile01/ssts1a
   pdstsrm -s b001 -f /sysfile01/ssts1b
   ```
4. Use the `pdstsinit` command to re-create the status files deleted in step 3.

   Then, specify a record count greater than the record count for the status files before the change.

   ```
   pdstsinit -s b001 -f /sysfile01/ssts1a -l 4096 -c 1000
   pdstsinit -s b001 -f /sysfile01/ssts1b -l 4096 -c 1000
   ```
5. Use the `pdstsopen` command to place the status files created in step 4 in spare status.

   ```
   pdstsopen -s b001 -n sstsfg01
   ```
6. Use the `pdstsswap` command to swap out the current file. Then use steps 1-5 of this procedure to increase the size of the file that was the current file.

   ```
   pdstsswap -s b001
   ```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

Note

- When you change spare files into reserved files, do not change all files into reserved files. If status files need to be swapped and there are no spare files, HiRDB (or the unit for a HiRDB/Parallel Server) terminates abnormally. Therefore, you must check that you will still have at least one spare file when you change the size of status files.
- You should be careful about specifying the `pd_syssts_last_active_file` or `pd_sts_last_active_file` operand. You must change the value specified in these operands before the next HiRDB startup because the current file was changed in step 6. If the

value is not changed, it will not be possible for HiRDB (or the unit for a HiRDB/Parallel Server) to start.

5.2.4 Changing the file status

The `pdstsopen` command is used to place a reserved file in spare status. The `pdstscls` command is used change a spare file into a reserved file.

5.2.5 Changing the current file

To use a different file as the current file, the `pdstsswap` command is used to swap status files. The current file becomes a spare file.

5.2.6 Adding a new status file

This section explains the procedure for adding a new status file. Before a status file is added, it is recommended that the status file design guidelines be reviewed; for details, see the manual *HiRDB Version 8 Installation and Design Guide*.

(1) When HiRDB can be terminated normally

Procedure

1. Use the `pdfstatfs` command to check for free space in the HiRDB file system area in which status files are to be created.

   ```
   pdfstatfs /sysfile01
   ```

   If there is no free space, create a new HiRDB file system area. For details, see *10.2 Creating (initializing) a HiRDB file system area*.
2. Use the `pdstop` command to terminate HiRDB normally.
3. Add the applicable operand listed below to the HiRDB system definition. Specify in this operand the status files to be created in step 4.
   - `pd_syssts_file_name` operand (for unit status files)
   - `pd_sts_file_name` operand (for server status files)
4. Use the `pdstsinit` command to add (initialize) the status files.

   ```
   pdstsinit -s b001 -f /sysfile01/ssts1a -l 4096 -c 1000
   pdstsinit -s b001 -f /sysfile01/ssts1b -l 4096 -c 1000
   ```
5. Use the `pdconfchk` command to check the HiRDB system configuration. If errors are detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
6. Use the `pdstart` command to start HiRDB normally.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution

results, see the manual *HiRDB Version 8 Command Reference.*

(2) When HiRDB cannot be terminated normally

Use the `pdchgconf` system reconfiguration command to change the HiRDB system definition. Note that HiRDB Advanced High Availability must have been installed in order to use this command.

Procedure

1. Use the `pdfstatfs` command to check for free space in the HiRDB file system area in which status files are to be created.

   ```
   pdfstatfs /sysfile01
   ```

   If there is no free space, create a new HiRDB file system area. For details, see *10.2 Creating (initializing) a HiRDB file system area*.

2. Create the `$PDDIR/conf/chgconf` directory.
3. Copy the HiRDB system definition files currently being used to the directory created in step 2.
4. Add the applicable operand listed below to the HiRDB system definition in the `$PDDIR/conf/chgconf` directory. Specify in this operand the status files to be added in step 5.
   - `pd_syssts_file_name` operand (for unit status files)
   - `pd_sts_file_name` operand (for server status files)
5. Use the `pdstsinit` command to add (initialize) the status files.

   ```
   pdstsinit -s b001 -f /sysfile01/ssts1a -l 4096 -c 1000
   pdstsinit -s b001 -f /sysfile01/ssts1b -l 4096 -c 1000
   ```

6. Use the `pdconfchk` command to check the HiRDB system definition in the `$PDDIR/conf/chgconf` directory. If an error is detected, correct the HiRDB system definition and re-execute the `pdconfchk` command.
7. Use the `pdchgconf` command to replace the existing HiRDB system definition with the modified system definition.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

5.2.7 Deleting a status file

The `pdstsrm` command is used to delete a reserved file or a file that is in shutdown status. Neither the current nor a spare file can be deleted.

Delete one of the following operands, as applicable. You should delete the operand that

correspond to the status files to be deleted by the pdstsrm command.

- pd_syssts_file_name operand (for unit status files)
- pd_sts_file_name operand (for server status files)

If these operands are not deleted, the deleted status files become virtual files. Therefore, if stop (the default) is specified in the following operands, HiRDB (or the unit for a HiRDB/Parallel Server) cannot be started.

- pd_syssts_initial_error
- pd_sts_initial_error

5.2.8 Checking the information in a status file

(1) Checking the information in a logical file

When you execute the pdls -d sts command, the following information is displayed:

- Logical file name
- Logical file status
- Record usage rate within the file
- Number of contiguous free records within the file
- Number of managed records within the file
- Active physical file
- Physical file status
- Record size
- Number of records
- Physical file name

(2) Checking information in a physical file

When you execute the pdcat -d sts command, the following information is displayed:

- Initialization date and time
- Date and time when the current file was selected
- Record size
- Number of records
- Record usage rate within the file
- Number of contiguous free records within the file

- Number of managed records within the file

5.3 Status changes of status files

Figure 5-2 and Table 5-3 show the status file status changes during HiRDB operation.

Figure 5-2: Status file status changes during HiRDB operation

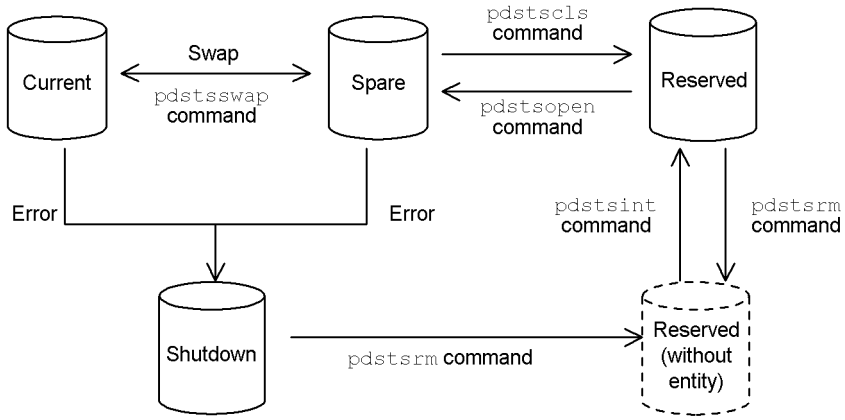


Table 5-3: Status file status changes during HiRDB operation

Event	Status of status file				
	Open		Closed		
	Current	Spare	Reserved		Hold
	ACTIVE*	STAND BY*	CLOSE*	NONE*	BLOCK ADE*
	1	2	3	4	5
Shortage of space for status files occurred	→2	→1	—	—	—
pdstsswap command was executed	→2	→1	—	—	—
pdstscls command was executed	—	→3	—	—	—
pdstsopen command was executed	—	—	→2	—	—

<table>
<tr><th colspan="5" rowspan="4">Event</th><th colspan="5">Status of status file</th></tr>
<tr><th colspan="2">Open</th><th colspan="3">Closed</th></tr>
<tr><th>Current</th><th>Spare</th><th colspan="2">Reserved</th><th>Hold</th></tr>
<tr><th>ACTIVE*</th><th>STAND BY*</th><th>CLOSE*</th><th>NONE*</th><th>BLOCK ADE*</th></tr>
<tr><th colspan="5">1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td rowspan="5">Output error occurred</td><td colspan="4">Single system</td><td>➜5</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td rowspan="4">Dual systems</td><td colspan="3">Spare files exist</td><td>➜5</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td rowspan="3">No spare files exist</td><td rowspan="2">Single system operation possible</td><td>Both systems operating</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>Single system operating</td><td>➜5</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td colspan="2">Single system operation not possible</td><td>➜5</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
</table>

Legend:

—: Does not apply, or status does not change.

➜*n*: Status to which status file changes after occurrence of the event.

For example, ➜1 indicates that after the event occurs, the status of the status file changes to current (status No. 1).

Note

- The status transitions listed in this table assume that the relevant events were processed normally.
- Even if the `pdstsinit` command or `pdstsrm` command is executed on reserved or hold files, the status of status files displayed by the pdls command does not change.

* Indicates the execution result (status of the status files) of the `pdls -d sts` command.

Chapter

6. Backup Procedures

The HiRDB administrator must back up the database in order to be prepared for possible errors. This chapter describes the backup procedures.

This chapter contains the following sections:

6.1 Backup

The HiRDB administrator must back up data in order to be prepared for possible errors. The database copy utility (`pdcopy` command) is used to make backups. This section explains the basics of making backups.

For details about optional functions related to making backups (such as a function whose objective is to reduce the amount of time required to make backups), see *6.1.2 Optional items*.

6.1.1 Basics

(1) Backup unit

The backup unit is specified in an option of the database copy utility. Data can be backed up in the following units:

- By system (all RDAREAs)
- By unit (all RDAREAs in a unit)*
- By server (all RDAREAs in a server)*
- By RDAREA (individual RDAREAs)

* Function applicable to HiRDB/Parallel Servers only.

(2) Backup timing

Data should be backed up daily. If this is not possible, data should be backed up at the following times:

1. Before and after executing the database load utility or database reorganization utility in the no-log mode or the pre-update log acquisition mode
2. After executing the database structure modification utility
3. Before and after executing the dictionary import/export utility (before and after importing table definition information or stored procedures)
4. After executing the optimizing information collection utility
5. After recovering RDAREAs with the database recovery utility (applicable when an unload log or system log information is used as the input information)
6. After executing a definition SQL or the `PURGE TABLE` statement
7. After updating a large amount of table data
8. After registering a plug-in (after executing the `pdplgrgst` command)
9. When utilization of a plug-in is disabled (after the `pdplugin` operand is deleted

from the system common definition)

10. Before installing a new version of HiRDB*
11. Before and after executing a UAP in the no-log mode (PDDBLOG=NO)

* The following RDAREAs should be backed up:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs

(3) Server machine to be used to store a backup file

A backup file can be created in any server machine on which HiRDB is running. There is no requirement that a backup file be created in the same server machine where the RDAREAs being backed up are located. A server machine containing a device such as CMT or DAT may be used.

The server machine to be used to store a backup file can be specified in an option of the database copy utility (pdcopy command).

(4) Backing up RDAREAs without using the database copy utility

When an RDAREA is backed up without using the database copy utility, the following must be completed before the backup operation is started:

- Use the pdhold -c command to place the RDAREA in shutdown and closed status.
- Use the pdhold -b command to place the RDAREA in backup-hold status.
- Terminate HiRDB normally.

(5) List RDAREAs

List RDAREAs cannot be backed up. There is no need to make such backups, because lists can be re-created easily as long as the tables used for the lists are available.

(6) Creating a backup file in a character special file

If a character special file is used for a backup file, the character special file must be placed in a HiRDB file system area for utilities. To do this, UTL must be specified in the -k option of the pdfmkfs command.

(7) Backup file size (important)

If a message reporting a disk space shortage is output during execution of the database copy utility even though ample disk space has been provided, the following are possible causes:

- Use of large files has not been specified (pd_large_file_use = Y is not

displayed).

- The maximum kernel parameter value has been exceeded.

In such a case, specify use of large files or change the kernel parameter value. You can also avoid the problem by creating multiple backup files. However, if the OS does not support large files, you must keep the disk partition size to 2 GB or smaller to be able to handle multiple files.

(8) Checking a backup file's contents

You can use the `pdbkupls` command to check the backup information collected by the database copy utility. For example, you can check the information listed below (for details about the information that can be checked, see the manual *HiRDB Version 8 Command Reference*):

- Backup acquisition date
- Names of RDAREAs for which backup was collected
- Value specified for the backup acquisition mode (`-M` option)

(9) Note

Do not restore from backup information collected by the database copy utility on another HiRDB. For the method of migrating data to another HiRDB, see *12.1 Migrating a table to another HiRDB system*.

6.1.2 Optional items

(1) Differential backup facility

The differential backup facility collects as a backup only the information changed since the previous backup was made. Therefore, this facility reduces the amount of time it takes to make the backup. If the database is large and the amount of data updating has been small, consider use of the differential backup facility. For details on how to use the differential backup facility, see *6.5 Acquiring a differential backup.*

(2) Backup-hold

In the following cases, you must apply backup-hold to RDAREAs subject to being backed up.

- When a backup is to be made without entering the `pdcopy` command (when making a backup using another product's facilities)
- When the Logical Volume Manager (LVM) is used for the database (to execute the `pdcopy` command in the updatable mode, the target RDAREAs must be backed up and held)

For details on backup-hold, see *6.8 Backup acquisition using backup-hold (backup without using the pdcopy command).*

(3) Frozen update command (pddbfrz command)

The *frozen update command* applies to user LOB RDAREAs. It is a function for placing in frozen update status HiRDB files with full data pages (all updating completed). Once HiRDB files are in frozen update status, they do not need to be backed up again. This means that the amount of time required to make subsequent backups of the user LOB area is reduced. For details about making backups when the frozen update command is used, see *6.9 Backup acquisition when the frozen update command (pddbfrz command) is used*.

(4) NetBackup linkage facility

When you use the NetBackup linkage facility, you can create backup files to be used by the database copy utility (`pdcopy`) or the database recovery utility (`pdrstr`) on a medium that is managed by the NetBackup server. The JP1/VERITAS NetBackup Agent for HiRDB License is required in order to use the NetBackup linkage facility. For details about the NetBackup linkage facility, see the manual *JP1/VERITAS NetBackup v4.5 Agent for HiRDB License Description and User's Guide*.

Note:

The NetBackup linkage facility cannot be used under the HP-UX (IPF), Linux (IPF) or Linux (EM64T).

(5) Linkage to JP1/OmniBack II (applicable to HP-UX only)

When JP1/OmniBack II is used, various media such as DLTs can be used as the backup destination. However, JP1/OmniBack II cannot be used for a 64-bit-mode HiRDB. For details on how to make backups using JP1/OmniBack II, see *6.7 Backup acquisition using JP1/OmniBack II (applicable to HP-UX only)*.

6.2 Backup acquisition mode

(1) Backup acquisition modes

The desired backup acquisition mode can be selected by specifying the `-M` option of the database copy utility (`pdcopy` command). Table 6-1 lists the backup acquisition modes.

Table 6-1: Backup acquisition modes

Backup acquisition mode (-M option specification)	Description of mode	Point to which RDAREAs can be restored
Referencing/ updating-impossible mode (`x`)	RDAREAs being backed up cannot be referenced or updated while the backup is being made. Subject RDAREAs must be placed in shutdown and closed status with the `pdhold -c` command before the backup is made.	Using a backup made in this mode, the database can be restored to the point when the backup was made. If a system log is used, the database can be restored to any synchronization point after the backup was made.
Referencing-permitted mode (`r`)	RDAREAs being backed up can only be referenced during backup processing; they cannot be updated.	
Updatable mode (`s`) [1]	RDAREAs being backed up can be referenced and updated during backup processing.	A database cannot be restored to the point when the backup was made. It can be restored only to a synchronization point subsequent to the point when the backup was made. Therefore, database restoration requires both the backup and the system log[2] from a synchronization point immediately prior to when the backup was made.

1

- When the updatable mode is specified, the RDAREAs to be backed up must have been created in a character special file; an RDAREA created in a regular file cannot be backed up in the updatable mode.
- Do not make a backup in the updatable mode during execution of a UAP (including utilities) in the no-log mode or pre-update log acquisition mode.
- To execute the `pdcopy` command in the updatable mode when the Logical Volume Manager (LVM) is used for the database, the target RDAREAs must be backed up and held. For details on backup-hold, see *6.8 Backup acquisition using backup-hold (backup without using the pdcopy command)*.

[2] The run ID and generation number of the system log file needed to recover the RDAREA are output to the database copy utility's processing results output file.

Notes

A backup made after an RDAREA structure change must be made in one of the following backup acquisition modes:

- Referencing/updating-impossible mode (`x`)
- Referencing-permitted mode (`r`)

(2) Relationship to the database update log acquisition mode

The backup acquisition mode that can be specified for backup processing depends on the database update log acquisition mode, as shown in Table 6-2. For details on the database update log acquisition mode, see *7. Operation Without Acquiring a Database Update Log.*

Table 6-2: Backup acquisition mode depending on the database update log acquisition mode

Database update log acquisition mode	Specifiable backup acquisition mode (-M option specification)
Log acquisition mode	Referencing/updating-impossible mode (`x`) Referencing-permitted mode (`r`) Updatable mode (`s`)
Pre-update log acquisition mode	Referencing/updating-impossible mode (`x`) Referencing-permitted mode (`r`)
No-log mode	

Notes

1. Do not make a backup in the updatable mode (`-M s` specification) during execution of a UAP (including utilities) in the no-log mode or pre-update log acquisition mode.
2. After execution of a UAP (including utilities) in the no-log mode or pre-update log acquisition mode, make the backup in one of the following modes:

- Referencing/updating-impossible mode (`-M x` specification)
- Referencing-permitted mode (`-M r` specification)

6.3 RDAREAs to be backed up together

The HiRDB administrator must back up not only RDAREAs that have been processed, but also RDAREAs that have been updated as a result of that processing. Table 6-3 lists the RDAREAs that should be backed up together.

Table 6-3: RDAREAs to be backed up together

Processing executed since previous backup	Type of RDAREA to be backed up									
	MST	DIR	DIC	DIC LOB	USR	USR LOB	REG	REG LOB	LOB indx	LOB data
ALTER PROCEDURE			Y	Y						
ALTER ROUTINE			Y	Y						
ALTER TABLE	Y[1]	Y[1]	Y		Y					
ALTER TRIGGER			Y	Y						
CREATE FUNCTION			Y	Y						
CREATE INDEX	Y	Y	Y		Y				Y[15]	
CREATE PROCEDURE			Y	Y						
CREATE SCHEMA, CREATE CONNECTION SECURITY			Y							
CREATE TABLE	Y	Y[2]	Y	Y[34]	Y	Y[9]				Y[16]
CREATE TRIGGER			Y	Y						
CREATE TYPE			Y	Y[24]						

Processing executed since previous backup	Type of RDAREA to be backed up									
	MST	DIR	DIC	DIC LOB	USR	USR LOB	REG	REG LOB	LOB indx	LOB data
CREATE VIEW	Y		Y							
DROP CONNECTION SECURITY			Y							
DROP DATA TYPE			Y	Y[24]						
DROP FUNCTION			Y	Y						
DROP INDEX	Y	Y	Y		Y				Y[17]	
DROP PROCEDURE			Y	Y						
DROP SCHEMA	Y[3]	Y[3]	Y	Y[14]	Y	Y[10]			Y[18]	Y[19]
DROP TABLE	Y	Y[8]	Y	Y[34]	Y	Y[9]			Y[20]	Y[21]
DROP TRIGGER			Y	Y						
DROP VIEW	Y		Y							
Definition SQL other than above	Y[7]		Y							
PURGE TABLE			Y[35]		Y	Y[9]			Y[20]	Y[21]
Other data manipulation SQL					Y[13]	Y[13]			Y[22]	Y[23]
Database load utility			Y[35]		Y	Y[9]			Y[20]	Y[21]

Processing executed since previous backup		Type of RDAREA to be backed up									
		MST	DIR	DIC	DIC LOB	USR	USR LOB	REG	REG LOB	LOB indx	LOB data
Database structure modif utility	RDAREA addition	Y		Y	Y[30]	Y[5]				Y[11]	Y[11]
	RDAREA expansion	Y		Y	Y[30]	Y[5]	Y[11]	Y[25]	Y[26]	Y[11]	Y[11]
	RDAREA deletion	Y		Y							
	RDAREA re-initial	Y	Y[4]	Y	Y[30]	Y[5]	Y[11, 12]	Y[25]	Y[26]		
Database reorg utility	Reload			Y[6, 35]	Y[6]	Y	Y[31]			Y[20]	Y[21]
	Reorg			Y[36]		Y[36]					
	Re-create an index					Y[32]				Y[33]	Y[33]
	Index reorg					Y					
Dict import/ export utility	Import table definition info	Y	Y	Y	Y[34]	Y	Y[9]				
	Import of stored procedure			Y	Y						
Integrity check utility				Y[35]		Y[35]					
Optimizing information collection utility				Y							
Registry facility initialization utility								Y[25]	Y[26]		
Rebalancing utility				Y	Y	Y	Y[9]			Y[20]	Y[21]
Database definition utility		Same as definition SQL									
`pdplgrgst` command				Y	Y						
`pdorend` command				Y[36]		Y[36]					

Processing executed since previous backup	Type of RDAREA to be backed up									
	MST	DIR	DIC	DIC LOB	USR	USR LOB	REG	REG LOB	LOB indx	LOB data
Deletion of plug-in	Y	Y	Y	Y	Y	Y[27]	Y	Y	Y[28]	Y[29]

modif: modification

re-initial: re-initialization

reorg: reorganization

Re-create: Re-creating

Dict: Dictionary

info: information

Y: RDAREAs to be backed up together

`MST`: Master directory RDAREA

`DIR`: Data directory RDAREA

`DIC`: Data dictionary RDAREA

`DIC LOB`: Data dictionary LOB RDAREA

`USR`: User RDAREA

`USR LOB`: User LOB RDAREA

`REG`: Registry RDAREA

`REG LOB`: Registry LOB RDAREA

`LOB indx`: User LOB RDAREA (applicable if plug-in index is stored)

`LOB data`: User LOB RDAREA (applicable if abstract data type is stored)

Note

To back up a data dictionary `LOB RDAREA` for storing objects, either the `-M` option of the database copy utility must be omitted or `x` or `r` must be specified in the `-M` option.

[1] Required if a column comprising the index was deleted or a column with a low ID was deleted from the index columns. The column IDs can be obtained by searching the `COLUMN_ID` column of the `SQL_COLUMNS` dictionary table.

[2] Required if a row-partitioned table or cluster key was defined.

[3] Required if a table or index was defined for a schema that was deleted.

[4] Required if an index is stored in an RDAREA that was deleted.

[5] Required if a user RDAREA was processed.

[6] Required if a dictionary table was reorganized.

[7] Required if a view table was deleted because access privilege to the base table was lost by `REVOKE` *access-privilege*.

[8] Required in the following cases:

- A row-partitioned table was deleted
- An index was defined for a table that was deleted
- A cluster key was defined for a table that was deleted

[9] Required if a LOB column is defined for a table that was processed.

[10] Required if a schema that was deleted contained a table for which a LOB column was defined.

[11] Required if a user LOB RDAREA was processed.

[12] Required if a user RDAREA that contained a table for which a LOB column was defined was initialized.

[13] If no LOB column is defined for the updated table, the user RDAREAs that contain this table must be backed up. If a LOB column is defined for the updated table, the RDAREAs listed in Table 6-4 must be backed up.

[14] Required if a routine, a table for which a `CASCADE` referential constraint action was defined, or a trigger was defined in a schema that is subject to deletion.

[15] Required if a plug-in index was created.

[16] Required if an abstract data type was defined for a table has the LOB attribute.

[17] Required if a plug-in index was deleted.

[18] Required if a schema that was deleted contained a table for which a plug-in index was defined.

[19] Required if a schema that was deleted contained a table for which the LOB attribute was defined.

[20] Required if a plug-in index was defined for a table that was processed.

[21] Required if the LOB attribute was defined for a table that was processed.

[22] Required if an attribute for which a plug-in index was defined was updated.

[23] Required if the LOB attribute was updated.

[24] Required if there was a function definition specified with an SQL procedure.

[25] Required if a registry RDAREA was processed.

[26] Required if a registry LOB RDAREA was processed.

[27] Required if the LOB attribute was defined for a table for which an abstract data type provided by a plug-in was defined.

[28] Required if a plug-in index was defined.

[29] Required if the LOB attribute was defined for an abstract data type provided by a plug-in.

[30] Required if a data dictionary LOB RDAREA was processed.

[31] Required if a LOB column was defined for the target table. Also, in cases where only a LOB column structure base table is reconfigured (`-j` option is not specified), the LOB column must be acquired together with the user RDAREA.

[32] The table storage RDAREA and the index storage RDAREA must be acquired as a pair.

[33] Also in cases where plug-in indexes are subject to processing, the table storage RDAREAs (user RDAREAs) must be acquired in preparation for no-log hold.

[34] Required if a referential constraint's action is `CASCADE`.

[35] Required if a referencing table, a referenced table, or a table for which a check constraint was defined is the processing target.

[36] Required if the check pending status is changed.

Table 6-4: RDAREAs to be backed up when a LOB column is defined for an updated table

Updating type and condition			For user	For user LOB
`INSERT` statement	Data exists in a column other than LOB column.	LOB column is null.	Y	—
		LOB column contains data.	Y	Y

Updating type and condition				For user	For user LOB
DELETE statement	Data exists in a column other than LOB column.	LOB column is null.		Y	—
		LOB column contains data.		Y	Y
UPDATE statement	Updating occurred in a column other than LOB column.	LOB column was not updated.		Y	—
		LOB column was updated.		Y	Y
	No updating occurred in a column other than LOB column.	LOB column was updated.	Null value was updated to some data.	Y	Y
			Data was updated to null.	Y	Y
			Data A was updated to data B.	—	Y

Y: Backup must be collected.

— : Backup need not be collected.

6.4 Examples of backup

Executor: HiRDB administrator

6.4.1 Example 1 (backing up a system)

All RDAREAs are backed up during operation of a HiRDB/Single Server.

(1) Using the pdlogswap command to swap the system log files

System log files are swapped in order to physically separate the system logs required for database restoration. To restore the RDAREAs from the backup copy to be created in step (2), the system log collected subsequently (the system log in the file to be used as the current file after swapping) will be used as input information.

```
pdlogswap -d sys -w
```

(2) Using the pdcopy command to back up all RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01
-z /pdcopy/logpoint01 -p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the referencing-permitted mode as the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up.

`-b`: Specifies the name of the backup file.

`-z`: Specifies the name of the log point information file. Specify this option when operation without unloading the system log or the automatic log unloading facility is being used.

`-p`: Specifies the output file name for the `pdcopy` command's processing results listing.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

6.4.2 Example 2 (backing up a system)

All RDAREAs are backed up during operation of a HiRDB/Parallel Server.

(1) Using the pdlogswap command to swap system log files at all back-end servers and the dictionary server

System log files are swapped in order to physically separate the system logs required for database restoration. To restore the RDAREAs from the backup copy to be created in step (2), the system log collected subsequently (the system log in the file to be used as the current file after swapping) will be used as input information.

```
pdlogswap -d sys -s bes1 -w
pdlogswap -d sys -s bes2 -w
pdlogswap -d sys -s dic -w
```

(2) Using the pdcopy command to back up all RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the referencing-permitted mode as the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up.

`-b`: Specifies the name of the backup file.

`-p`: Specifies the output file name for the `pdcopy` command's processing results listing.

Remarks

When the automatic log unloading facility is being used, execute the `pdlogatul` command to record the unload log file name that corresponds to the current system log file when the backup is being made.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

6.4.3 Example 3 (backing up a system)

HiRDB is terminated, then the `pdstart -r` command is used to start HiRDB and all RDAREAs are backed up.

Note

A log point information file cannot be created when this method is used. This method should not be used when the system is operating without unloading the

system log.

(1) Entering the pdstop command to terminate HiRDB normally

```
pdstop
```

(2) Entering the pdstart -r command to start HiRDB

```
pdstart -r
```

(3) Using the pdcopy command to back up all RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M x -a -b /pdcopy/backup01 -p pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the referencing/updating-impossible mode as the backup acquisition mode. Because the `pdstart -r` command was used to start HiRDB, there is no need to place the RDAREAs in shutdown and closed status even if `x` is specified.

`-a`: Specifies that all RDAREAs are to be backed up.

`-b`: Specifies the name of the backup file.

`-p`: Specifies the output file name for the `pdcopy` command's processing results listing.

Remark

When the automatic log unloading facility is being used, execute the `pdlogatul` command to record the unload log file name that corresponds to the current system log file when backup is being made.

(4) Entering the pdstop command to terminate HiRDB normally

```
pdstop
```

(5) Entering the pdstart command to start HiRDB normally

```
pdstart
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

6.4.4 Example 4 (backing up a unit)

All RDAREAs in a unit are backed up during operation of a HiRDB/Parallel Server.

(1) Using the pdlogswap command to swap the system log files in the unit

System log files are swapped in order to physically separate the system logs required for database restoration. To restore the RDAREAs from the backup copy to be created in step (2), the system log collected subsequently (the system log in the file to be used as the current file after swapping) will be used as input information.

```
pdlogswap -d sys -s bes1 -w
pdlogswap -d sys -s bes2 -w
```

(2) Using the pdcopy command to back up the entire unit

```
pdcopy -m /rdarea/mast/mast01 -M r -u UNT1 -b /pdcopy/backup01
-p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the referencing-permitted mode as the backup acquisition mode.

`-u`: Specifies that all RDAREAs under the unit (UNT1) are to be backed up.

`-b`: Specifies the name of the backup file.

`-p`: Specifies the output file name for the `pdcopy` command's processing results listing.

Remarks

When the automatic log unloading facility is being used, execute the `pdlogatul` command to record the unload log file name that corresponds to the current system log file when backup is being made.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

6.4.5 Example 5 (backing up a server)

All RDAREAs in a back-end server (`bes1`) are backed up during operation of a

HiRDB/Parallel Server. A log point information file is also obtained, because the system is operating without unloading the system log.

(1) Using the pdlogswap command to swap the system log files in the server that is to be backed up

System log files are swapped in order to physically separate the system logs required for database restoration. To restore the RDAREAs from the backup copy to be created in step (2), the system log collected subsequently (the system log in the file to be used as the current file after swapping) will be used as input information.

```
pdlogswap -d sys -s bes1 -w
```

(2) Using the pdcopy command to back up the entire server

```
pdcopy -m /rdarea/mast/mast01 -M r -s bes1 -b /pdcopy/backup01
-z /pdcopy/logpoint01 -p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the referencing-permitted mode as the backup acquisition mode.

`-s`: Specifies that all RDAREAs under the back-end server (`bes1`) are to be backed up.

`-b`: Specifies the name of the backup file.

`-z`: Specifies the name of the log point information file. Specify this option when operation without unloading the system log or the automatic log unloading facility is used.

`-p`: Specifies the output file name for the `pdcopy` command's processing results listing.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

6.4.6 Example 6 (Backing up RDAREAs)

Specified RDAREAs (`rdarea01` and `rdarea02`) are backed up during HiRDB operation.

(1) Using the pdhold -c command to place the RDAREAs to be backed up in shutdown and closed status

This operation is necessary when `x` (referencing/updating-impossible) is specified as

the backup acquisition mode; it is not necessary when r (referencing-permitted mode) or s (updatable mode) is specified as the backup acquisition mode.

```
pdhold -r rdarea01,rdarea02 -c
```

(2) Using the pdlogswap command to swap the system log files at the server that contains the RDAREAs to be backed up

System log files are swapped in order to physically separate the system logs needed for database restoration. To restore the RDAREAs from the backup copy to be created in step (3), the system log information collected subsequently (the system log information in the file to be used as the current file after swapping) will be used as input information.

```
pdlogswap -d sys -s bes1 -w
```

(3) Using the pdcopy command to back up specified RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M x -r rdarea01,rdarea02
-b /pdcopy/backup01 -p /pdcopy01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the referencing/updating-impossible mode as the backup acquisition mode.

-r: Specifies the RDAREAs (rdarea01 and rdarea02) that are to be backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output file name for the pdcopy command's processing results listing.

Remarks

When the automatic log unloading facility is being used, execute the pdlogatul command to record the unload log file name that corresponds to the current system log file when backup is being made.

(4) Using the pdrels -o command to release shutdown and open RDAREAs

This operation is necessary when x (referencing/updating-impossible) is specified as the backup acquisition mode; it is not necessary when r (referencing-permitted mode)

or s (updatable mode) is specified as the backup acquisition mode.

```
pdrels -r rdarea01,rdarea02 -o
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

6.5 Acquiring a differential backup

Executor: HiRDB administrator

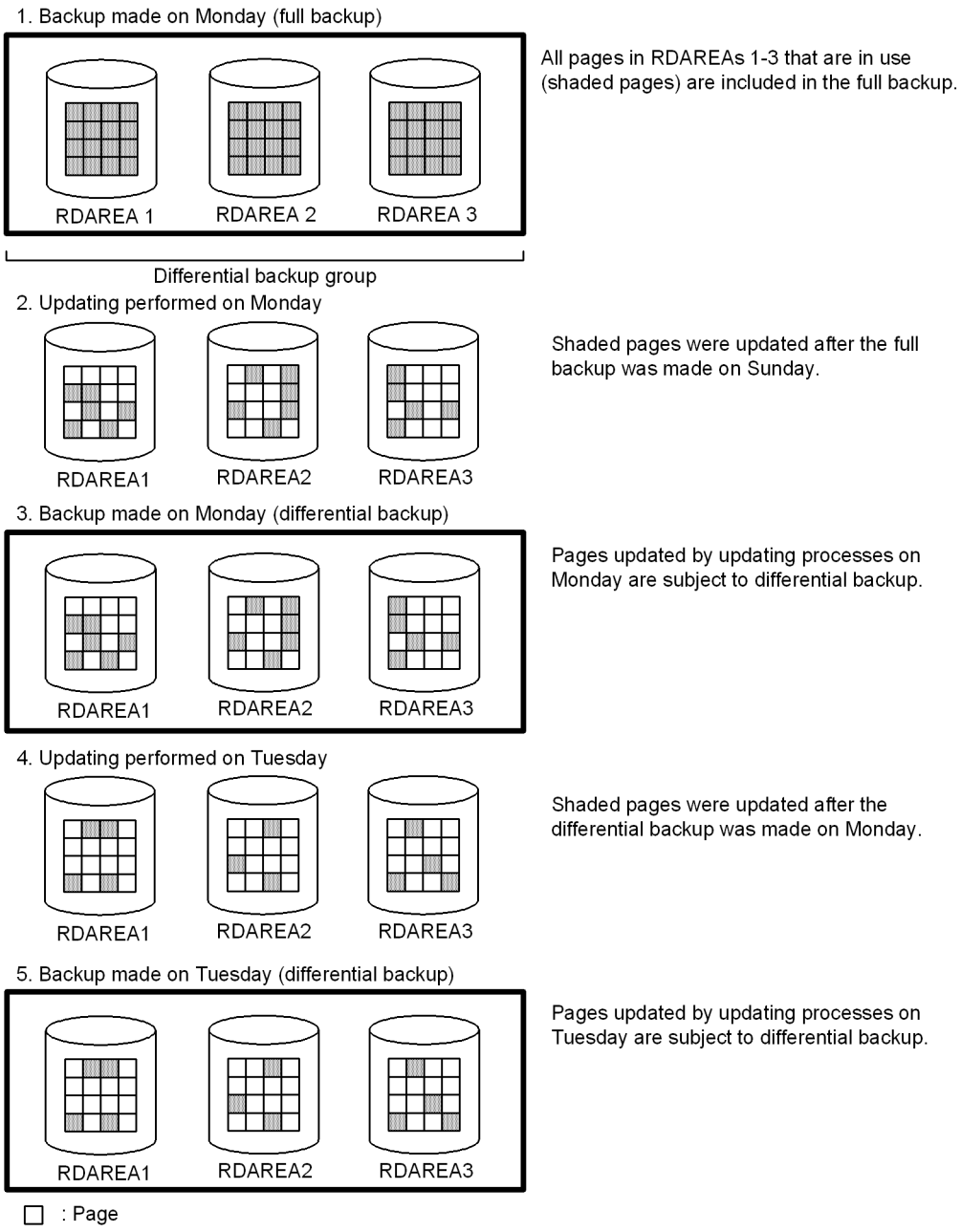
This section explains making backups with the differential backup facility. The following items are explained in this section:

- Differential backup facility overview
- Preparations for using the differential backup facility
- Operation example of the differential backup facility
- Creating an accumulation-differential backup
- Referencing the history file for differential backups
- Restoring a differential backup management file

6.5.1 Differential backup facility overview

Because backups are acquired in units of RDAREAs, both updated pages and unupdated pages are included in a backup. When you use the differential backup facility, only pages updated since the last time a backup was made are included in a new backup. When you make a backup of only the material that differs from the previous backup, the backup processing time is reduced. If the database is large but only a small amount of data has been updated, you should consider using the differential backup facility. Figure 6-1 provides an overview of the differential backup facility.

Figure 6-1: Overview of the differential backup facility



Explanation

1. RDAREAs 1-3 are backed up on Sunday. All pages in RDAREAs 1-3 that are currently in use are backed up. This backup is called a *full backup*, and the RDAREAs in the group that is backed up are referred to as the *differential backup group*.
2. Update processing is performed during the workday on Monday.
3. RDAREAs 1-3 are backed up after the workday is over on Monday. At this time, only the updated pages in RDAREAs 1-3 are backed up. Such a backup is called a *differential backup*.
4. Update processing is performed during the workday on Tuesday.
5. RDAREAs 1-3 are backed up after the workday is over on Tuesday. At this time, only the updated pages in RDAREAs 1-3 are backed up.

Recovering the database

For details about recovering the database when you use the differential backup facility, see *19.4 Database recovery using the differential backup facility*.

Remarks

You can also use the differential backup facility when operating without unloading the system log.

Notes

- You store all backup files (full backup files, differential backup files, and accumulation-differential backup files) from the same differential backup group on the same server machine. For details about accumulation-differential backup files, see *6.5.4 Creating an accumulation-differential backup*.
- A differential backup cannot be made of LOB RDAREAs. Rather, all pages that are in use must be backed up every time a backup is made.
- If the configuration of an RDAREA has changed, a full backup of the RDAREA must be made again. A backup made before an RDAREA's configuration was changed cannot be used to restore the RDAREA.
- You must use JP1/OmniBack II or the NetBackup linkage facility to back up RDAREAs onto a tape device. If you do not use JP1/OmniBack II or the NetBackup linkage facility, it will not be possible to back up RDAREAs onto a tape device.
- When you use JP1/OmniBack II to back up RDAREAs, do not update bar list files after the `pdcopy` command terminates. Also, do not do this without specifying protection for the bar list files.

6.5.2 Preparations for using the differential backup facility

(1) Creating a HiRDB file system area for storing differential backup management files

A file called the *differential backup management file* is output whenever a backup is made. This file stores information about the differential backup, which is used while the backup is being made and when the backup is used by HiRDB for database restoration.

You use the `pdfmkfs` command to create a HiRDB file system area for storing differential backup management files; for a HiRDB/Parallel Server, you must create the HiRDB file system area in the unit in which the system manager is defined:

```
pdfmkfs -n 10 -l 4096 -e 60000 -k UTL /pdcopy/admfile
```

Explanation

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

`-l`: Specifies the maximum number of files.

`-e`: Specifies the number of increments.

`-k`: Specifies the purpose of the HiRDB file system area (a HiRDB file system area for utilities).

`/pdcopy/admfile`:

Specifies a name for the HiRDB file system area. Differential backup management files will be created in this HiRDB file system area. The name of the differential backup management file is the same as the backup group name.

Notes

- Differential backup management files are important files because they contain information about the differential backups that are made. Without these files, the differential backup facility cannot be used and the database cannot be restored. For this reason, it is important not to delete this HiRDB file system area or the differential backup management files it contains. Each time a backup is made, you can use the `pdfbkup` command to back up the differential backup management files.
- When the system switchover facility is being used, you must create this HiRDB file system area (as a character special file) on the hard disk that is shared by the primary and secondary systems. If the HiRDB file system area is created as a regular file, the following steps must be taken:

Procedure

1. Create HiRDB file system areas with the same name in the primary and secondary systems.
2. Each time a backup is made, use the `pdfbkup` command to back up the differential backup management files.
3. Use ftp, etc., to copy into the secondary system the backup of the differential backup management files that you created in Step 2.
4. Use the `pdfrstr` command to restore the differential backup management files in the secondary system.

Remarks

Backup files and differential backup management files can be stored in the same HiRDB file system area.

(2) Differential backup group for RDAREAs

A group of RDAREAs to which the differential backup facility is applied is called a *differential backup group*. This section explains how to group RDAREAs.

(a) Restoring RDAREAs to the most recent differential backup acquisition point, or restoring RDAREAs using an unload log file (system log files)

In this case, it is possible to restore a specific RDAREA in a differential backup group. When the database contains a large amount of data, grouping all RDAREAs together for the purpose of making backups makes it time-consuming to restore a specific RDAREA. This is because of the amount of time required to load the backup files sequentially.

Therefore, grouping RDAREAs into manageable groups and making backups by group is recommended. For example, when a disk error occurs, RDAREAs are restored one disk at a time. Therefore, the RDAREAs stored on the same disk can constitute a group.

(b) Restoring RDAREAs to other than the most recent differential backup acquisition point (restoring RDAREAs using only full backup files)

In this case, it is not possible to restore a specific RDAREA in a differential backup group. All RDAREAs in the group must be restored.

Therefore, it is recommended that RDAREAs that are related constitute each group. For example, one group of RDAREAs might store the data of the same table, while another group of RDAREAs might store the indexes for this table. In this way, both the RDAREAs for the table and the RDAREAs for the indexes can be restored at the same time with a single execution of the `pdrstr` command.

Notes

When RDAREAs are restored to other than the most recent differential backup acquisition point by using only the full backup files, the information for differential backups subsequent to the restoration point in the differential backup management files becomes invalid. Consequently, it will not be possible to restore RDAREAs to the most recent differential backup acquisition point. Note that differential backup information is not invalidated when an error occurs during restoration.

6.5.3 Examples of using the differential backup facility

The following procedure for applying the differential backup facility is illustrated by the examples later in this section:

Procedure

To apply the differential backup facility:

1. Select the RDAREAs to which the differential backup facility is to be applied (determine the differential backup group).
2. Make a full backup of the RDAREAs to be backed up.
3. Make differential backups of the RDAREAs to be backed up (for example, make a backup once a day).
4. Make a full backup of the RDAREAs being backed up on a regular basis (for example, make a full backup once a week).

The operation example of the differential backup facility assumes the following conditions:

Conditions

- The differential backup facility is applied to the two user RDAREAs `rdarea01` and `rdarea02`.
- A full backup is made once a week (on Sunday).
- Update information only is collected as a differential backup every day (Monday through Saturday).

(1) Making a full backup (on Sunday)

Use the `pdcopy` command to make a full backup:

```
pdcopy -m /rdarea/mast/mast01 -M r -r rdarea01,rdarea02 -g 'backupg1(S)'
-b /pdcopy/backup01 -d a -K /pdcopy/admfile -L 5 -o /pdcopy/rfile
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the referencing-permitted mode as the backup acquisition mode.

-r: Specifies the RDAREAs to be backed up.

The RDAREA group specified here becomes the backup group. The RDAREAs constituting the group cannot be changed subsequently.

-g: Specifies a name for the differential backup group.

When the first full backup is made, (S) *must be specified in the differential backup group name*. The differential backup group name specified in this option is used when subsequent differential backups are made.

When (S) is specified, the entire differential backup group name must be enclosed in apostrophes (e.g., '`backupg1`(S)'). However, the enclosing apostrophes must not be used when the -g option is specified in a control statements file.

-b: Specifies a name for the backup file (full backup file name).

-d: Specifies the backup type:

a: Full backup

b: Accumulation-differential backup since the most recent full backup

c: Accumulation-differential backup since either the previously collected accumulation-differential backup or the previously collected full backup, whichever is most recent

d: Differential backup

For details on accumulation-differential backups, see *6.5.4 Creating an accumulation-differential backup.*

-K: Specifies the name of the HiRDB file system area in which the differential backup management file is to be stored.

-L: Specifies (in megabytes) the size of the differential backup management file.

-o: Specifies the name of the history file for differential backups.

For details on the history file for differential backups, see *6.5.5 Referencing the history file for differential backups.*

(2) Making a backup of the differential backup management file (every Sunday)

Use the pdfbkup command to make a backup of the created differential backup management file:

```
pdfbkup /pdcopy/admfile/backupg1 /backup/pdcopy/admfile/backupg1
```

Explanation

`/pdcopy/admfile/backupg1`:

Specifies the name of the differential backup management file. This name is the same as the differential backup group name.

`/backup/pdcopy/admfile/backupg1`:

Specifies a name for the backup file of the differential backup management file.

(3) Making a differential backup (daily, Monday through Saturday)

Use the `pdcopy` command to make a differential backup:

```
pdcopy -m /rdarea/mast/mast01 -M r -g backupg1 -b /pdcopy/backup02
-d d -K /pdcopy/admfile -o /pdcopy/rfile
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the referencing-permitted mode as the backup acquisition mode.

`-g`: Specifies the name of the differential backup group; (S) need not be specified.

`-b`: Specifies a name for the backup file (differential backup file name).

`-d`: Specifies the backup type:

`a`: Full backup

`b`: Accumulation-differential backup since the most recent full backup

`c`: Accumulation-differential backup since either the previously collected accumulation-differential backup or the previously collected full backup, whichever is most recent

`d`: Differential backup

For details on accumulation-differential backup, see *6.5.4 Creating an accumulation-differential backup.*

`-K`: Specifies the name of the HiRDB file system area in which the differential backup management file is to be stored.

`-o`: Specifies the name of the history file for differential backups.

For details on the history file for differential backups, see *6.5.5 Referencing the history file for differential backups.*

(4) Making a backup of the differential backup management file (daily, Monday through Saturday)

Use the `pdfbkup` command to make a backup of the created differential backup management file:

```
pdfbkup -r /pdcopy/admfile/backupg1 /backup/pdcopy/admfile/backupg1
```

Explanation

`-r`: Specifies that the last backup of the differential backup management file is to be overwritten.

`/pdcopy/admfile/backupg1`:

Specifies the name of the differential backup management file. This name is the same as the differential backup group name.

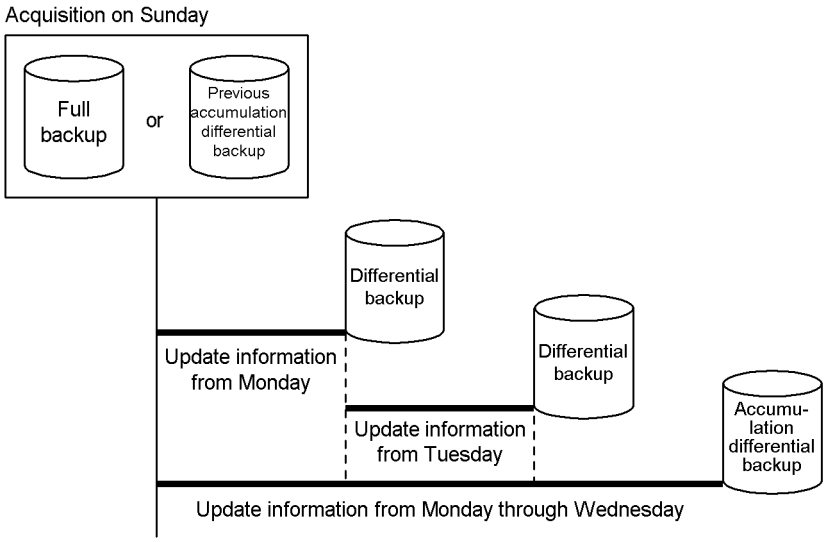
`/backup/pdcopy/admfile/backupg1`:

Specifies a name for the backup file of the differential backup management file.

6.5.4 Creating an accumulation-differential backup

To create an accumulation-differential backup, you execute the `pdcopy` command with `c` specified in the `-d` option. Creating an accumulation-differential backup may reduce the size of the backup file to be used for restoration. For example, if the same page is updated every day, all the update information for that page is merged, so the size of the backup file to be used for restoration is reduced. Figure 6-2 illustrates the concept of an accumulation-differential backup.

Figure 6-2: Concept of an accumulation-differential backup



Explanation

- When the accumulation-differential backup is made on Wednesday, the differential backups made on Monday and Tuesday will not be used for database restoration. The backups necessary for database restoration are the full backup and the accumulation-differential backup made on Wednesday.
- Whether an accumulation-differential backup is to be made from the full backup or the previously collected accumulation-differential backup is specified in the `-d` option of the `pdcopy` command.

(1) Precautions

Once an accumulation-differential backup is created, you should use the `pdfbkup` command to make a backup of the differential backup management files.

(2) Examples of making accumulation-differential backups

As an example, you might wish to perform the following operations when making accumulation differential backups:

- Make a full backup on Sunday.
- Make differential backups on Monday and Tuesday.
- On Wednesday, make an *accumulation-differential backup* for Monday through Wednesday.*
- Make differential backups on Thursday and Friday.

- On Saturday, make an *accumulation-differential backup* for Thursday through Saturday.*

* Following is an example of specifying the pdcopy command for the above Wednesday and Saturday tasks:

```
pdcopy -m /rdarea/mast/mast01 -M r -g backupg1 -b /pdcopy/backup03
-d c -K /pdcopy/admfile -o /pdcopy/rfile
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the referencing-permitted mode as the backup acquisition mode.

-g: Specifies the name of the differential backup group; (S) need not be specified.

-b: Specifies a name for the backup file (accumulation-differential backup file name).

-d: Specifies the backup type:

a: Full backup

b: Accumulation-differential backup since the most recent full backup

c: Accumulation-differential backup since either the previously collected accumulation-differential backup or the previously collected full backup, whichever is most recent

d: Differential backup

-K: Specifies the name of the HiRDB file system area in which the differential backup management file is to be stored.

-o: Specifies the name of the history file for differential backups.

6.5.5 Referencing the history file for differential backups

The history file for differential backups contains differential backup acquisition information in time series order. This file can be referenced with the pdcopy command. The statistical information file can be referenced with an OS text editor, etc.The following is an example of the output from a history file for differential backups:

```
pdcopy (06-00) ***** DB COPY *****  2000-10-26 21:30:20[1] utl3[2]

/hirdb/pdcopy/admfile[3]
backupg1[4]
2000-10-19 21:00:20[5]
2000-10-26 21:18:20[6]
RDAREA NAME :rdarea01,rdarea02[7]
a[8] 2000-10-19 21:08:20[9],2000-10-19 21:09:25[10] r[11]u[12] host01:/hirdb/backup01[13]
d[8] 2000-10-20 21:01:20[9],2000-10-20 21:01:38[10] r[11]u[12] host01:/hirdb/backup02[13]
c[8] 2000-10-21 21:05:20[9],2000-10-21 21:06:10[10] r[11]u[12] host01:/hirdb/backup03[13]
b[8] 2000-10-22 21:01:20[9],2000-10-22 21:01:45[10] r[11]u[12] host01:/hirdb/backup04[13]
```

Explanation

1. `pdcopy` command's execution date
2. HiRDB identifier
3. Name of the HiRDB file system area containing the differential backup management file
4. Backup group name
5. Backup group's creation date
6. Date of the most recent update of the backup group
7. Names of the RDAREAs belonging to the backup group
8. Backup type:

   `a`: Full backup

   `b`: Accumulation-differential backup since the most recent full backup

   `c`: Accumulation-differential backup collected since either the previously collected accumulation-differential backup or the previously collected full backup, whichever is most recent

   `d`: Differential backup
9. Backup acquisition start time
10. Backup acquisition completion time
11. Backup acquisition mode:

    `x`: Updating/referencing-impossible mode

    `r`: Referencing-permitted mode

    `s`: Updatable mode
12. Backup file type:

u: Regular file

i: HiRDB file system area for backup files

o: JP1/OmniBack II object

13. Backup file name or object name:

    Displayed in the format *host-name*:*backup-file-name*

    If the name is an object name, it is followed by (-G *bar-list-file-name*).

6.5.6 Restoring a differential backup management file

If an error occurs in a differential backup management file, the differential backup facility cannot be used. Therefore, in the event of an error in a differential backup management file, one of the following methods must be used to correct the error:

- Restore the differential backup management file from a backup.
- Make a new full backup with (s) specified in the differential backup group name.

6.6 Example of shell for backing up after synchronization point dump validation

Example

While the system is operating without unloading the system log, data is backed up after a synchronization point dump has been validated so that the system log files created before the backup can be released:

```
#!/bin/sh
$PDDIR/bin/pdlogsync -d sys -w
if [ $status = 0 ] ; then
    $PDDIR/bin/pdcopy -m host2:/dbarea/area1/rdmt1 -i -p /usr/ofile
    -f /usr/seifile/cofl01
    -z /usr/logpoint/logp01
else
    echo "synchronization-point-dump-validation-error"
fi
```

Explanation

Because the `-w` option is specified in the `pdlogsync` command, the synchronization point dump has been validated if the return code is 0.

When the synchronization point dump is validated, the database copy utility (`pdcopy` command) is used to execute backup processing. If validation failed, backup processing is not executed.

6.7 Backup acquisition using JP1/OmniBack II (applicable to HP-UX only)

When JP1/OmniBack II is used, various media such as DLT can be used as the backup destination. This section explains the use of JP1/OmniBack II to make backups. This section assumes that the reader is familiar with JP1/OmniBack II.

Conditions

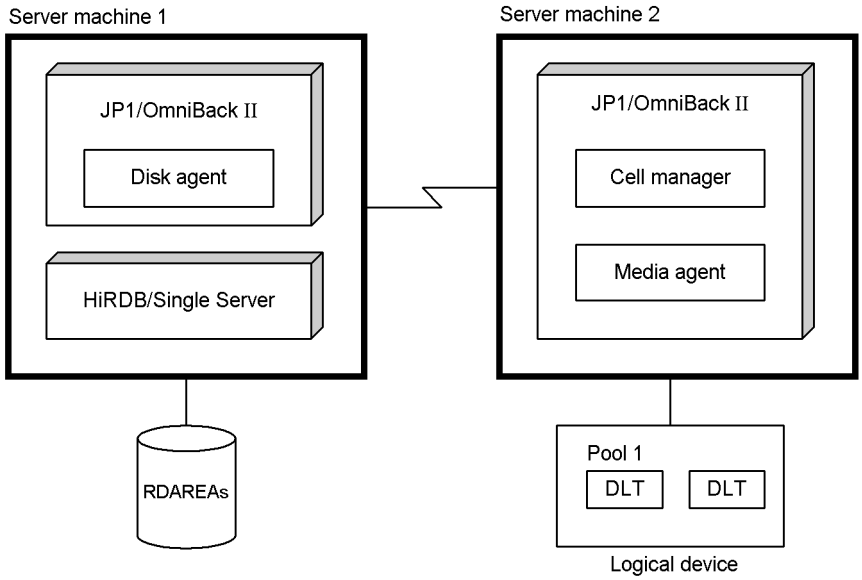
- Use of JP1/OmniBack II Version 05-20 is assumed.
- JP1/OmniBack II cannot be used in 64-bit-mode HiRDB.

6.7.1 System configuration example

(1) HiRDB/Single Server

Figures 6-3 and 6-4 show examples of a system configuration when JP1/OmniBack II is used to make backups.

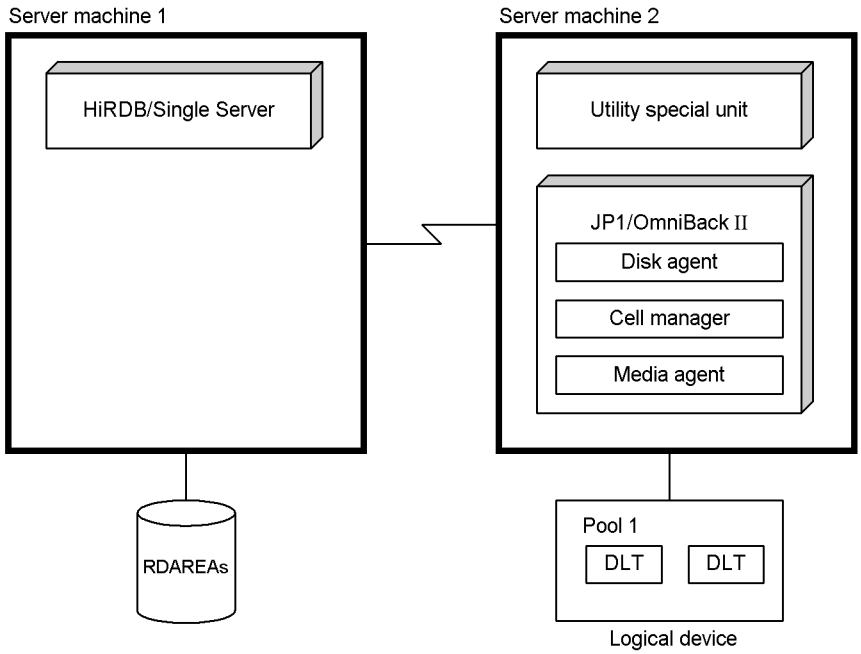
Figure 6-3: System configuration example for backup made using JP1/OmniBack II: JP1/OmniBack II handles communications between server machines



Explanation

JP1/OmniBack II must be installed at each server machine.

Figure 6-4: System configuration example for backup made using JP1/OmniBack II: HiRDB handles communications between server machines



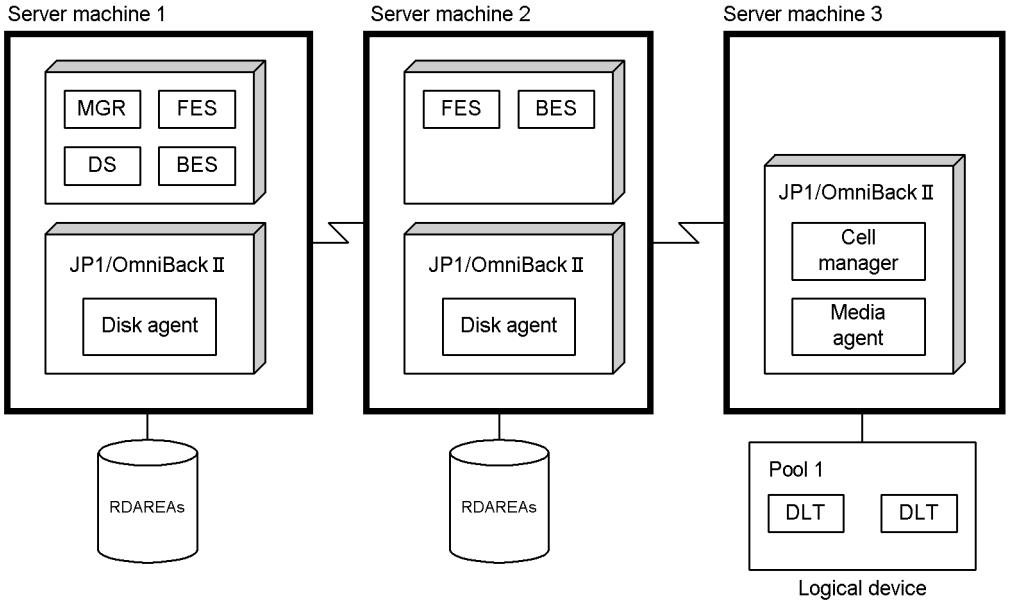
Explanation

When HiRDB handles communications between server machines, JP1/OmniBack II must be installed at the server machine in which the utility special unit is located.

(2) HiRDB/Parallel Server

Figures 6-5 and 6-6 show examples of a system configuration when JP1/OmniBack II is to be used to make backups.

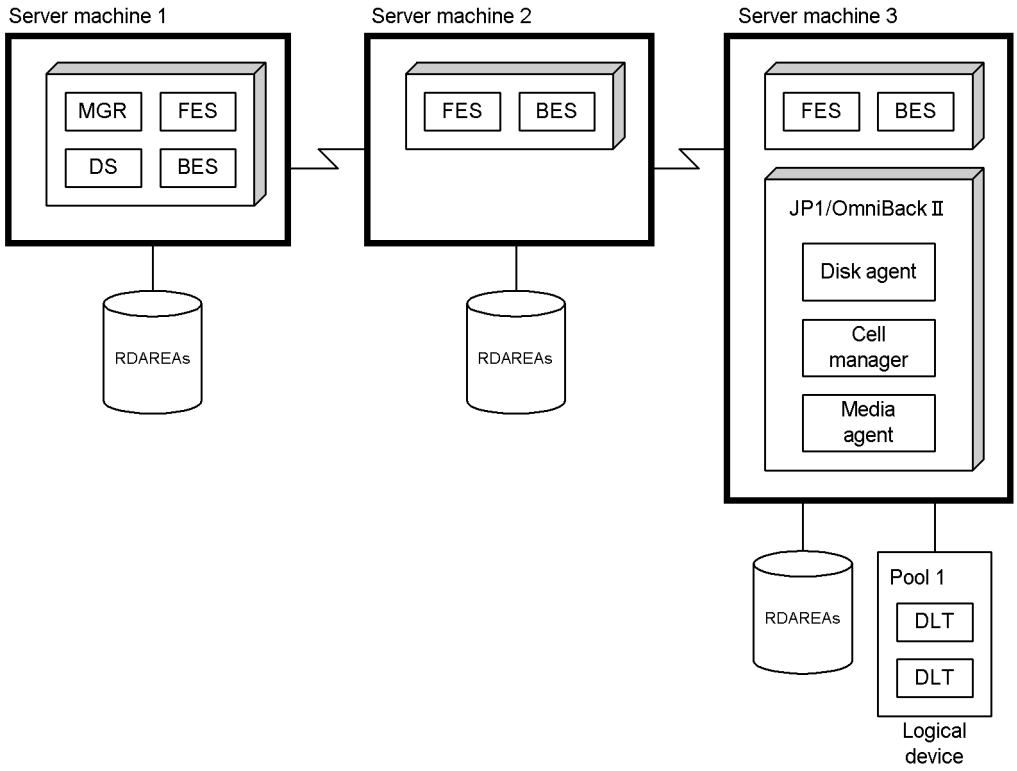
Figure 6-5: System configuration example for backup made using JP1/OmniBack II: JP1/OmniBack II handles communications between server machines



Explanation

JP1/OmniBack II must be installed at each server machine.

Figure 6-6: System configuration example for backup made using JP1/OmniBack II: HiRDB handles communications between server machines



Explanation

When HiRDB handles communications between server machines, JP1/OmniBack II must be installed at the server machine in which the backup device is located.

6.7.2 Environment setup

The following environment settings are required in order to make backups.

(1) Obtaining a license for and installing JP1/OmniBack II

Obtain a license for and install JP1/OmniBack II. For details on licensing, see the materials included with the JP1/OmniBack II software.

After installing JP1/OmniBack II, execute the `pdsetup` command in each server machine. In this step, specify `OmniBack` in the `-p` option. If the `pdsetup` command has already been executed, first execute the `pdsetup -d` command, then execute the `pdsetup` command. For details on the `pdsetup` command, see the manual *HiRDB*

Version 8 Command Reference. OmniBack must be specified in the `-p` option for all server machines comprising HiRDB.

(2) Distributing the disk agent and media agent

For details about distributing the disk agent and media agent, see the manual for JP1/OmniBack II.

(3) Registering users into JP1/OmniBack II

Register the HiRDB administrator as a JP1/OmniBack II user in the admin user class.

(4) Creating a logical device

For details about creating a logical device, see the manual for JP1/OmniBack II.

(5) Creating a directory for storing barlist files

Create the applicable directory listed below (directory for storing barlist files) in the server machine in which the cell manager of JP1/OmniBack II is located:

- HP-UX: `/etc/opt/omni/barlists/stream`

Set up the directory privileges, etc., as follows:

- User: `root`
- Group: `sys`
- Privilege: `0755`

(6) Creating a barlist file

Create a barlist file in the directory created in Step (5). Set up the barlist file privilege, etc., as follows:

- User: `root`
- Group: `sys`
- Privilege: `0644`

Up to 64 alphanumeric characters can be used for a barlist file name; the following is an example of creating a barlist file:

```
BARLIST "barlist-file-name"                          1
DEFAULTS
{
}
DEVICE "logical-device-name"                         2
{
}
CLIENT HiRDB host-name                               3
{
  -public                                            4
} -protect protection-specification                  5
```

Explanation

1. Specifies a name for the barlist file.

   Example:

   ```
   BARLIST "DLT01FILE"
   ```

2. Specifies the name of the logical device.

   Example:

   ```
   DEVICE "DLT01"
   ```

3. Specifies the name of the host where the disk agent is located.

   Example:

   ```
   CLIENT HiRDB host01
   ```

4. This is required.

5. Specifies the object protection type; no protection, specified duration, until a specified date, or permanent protection can be specified.

   Example:

   ```
   -protect: none
   ```

   - `none`: No protection
   - `days n`: Specify for n a protection duration in terms of number of days.
   - `weeks n`: Specify for n a protection duration in terms of number of weeks.
   - `until Date`: Specify for date the date on which protection is to expire (use the format *YYYY/MM/DD*).
   - `permanent`: Specify for permanent protection.

About object protection specification

When object protection is specified, the medium cannot be reused until protection for all objects on the medium has been released. Because this fact should be taken into consideration when protection is specified, the following guidelines for specifying protection are provided:

Guidelines

- If the medium is to be overwritten each time, specify there be no protection (`none`). In this case, only one object can be stored on the medium.
- When protection is specified, multiple objects can be stored on the same medium. In this case, store objects with approximately the same protection expiration dates. This makes it possible for the medium to be reused efficiently, because protection of all objects on the medium will expire at about the same time.

(7) Setting automatic processing of the mount prompt

Set automatic processing of JP1/OmniBack II's mount prompt. For details about automatic processing of the mount prompt, see the manual for JP1/OmniBack II.

6.7.3 Notes on backup acquisition

(1) Backup acquisition units

If there is a large volume of data, do not back up all RDAREAs into a single object. If all RDAREAs are backed up into a single object, restoration of a specific RDAREA will take a long time because the RDAREAs must be loaded sequentially. Therefore, it is recommended that RDAREAs be divided into groups for making backups.

For example, when a disk error occurs, RDAREAs are restored one disk at a time, so it is recommended that the backup of a single disk be collected into a single object.

(2) Checking the status of JP1/OmniBack II

Execute the `omnistat` command of JP1/OmniBack II to check the status of JP1/OmniBack II during execution of the `pdcopy` or `pdrstr` command.

(3) Backup acquisition example

In this example, multiple backup generations are saved.

Conditions

- Slots 1, 2, and 3 of the DLT library are defined in logical device 1 of JP1/OmniBack II.
- Slots 4, 5, and 6 of the DLT library are defined in logical device 2 of JP1/OmniBack II.

Operation example

1. The first backup is collected as object 1 on November 30, 2000, into logical device 1. The protection expiration date is set to December 13, 2000.
2. The second backup is collected as object 2 on December 7, 2000, into logical device 2. The protection expiration date is set to December 20, 2000.
3. The third backup is collected as object 1 on December 14, 2000, into logical device 1. The protection expiration date is set to December 27, 2000.

When the protection expiration dates are specified in this overlapping manner, the medium is reused efficiently.

Note that if permanent protection is specified, you must use JP1/OmniBack II's `omnidb` command to release object protection when the third and subsequent backups are made.

6.7.4 Example 1 (HiRDB/Single Server)

All RDAREAs are backed up during operation of a HiRDB/Single Server. The system configuration is shown in Figure 6-3.

(1) Using the pdlogswap command to swap system log files

System log files are swapped in order to physically separate the system logs required for database restoration. To restore the RDAREAs from the backup copy to be created in step (2), the system log collected subsequently (the system log in the file to be used as the current file after swapping) will be used as input information.

```
pdlogswap -d sys -w
```

(2) Using the pdcopy command to back up all RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -k o -b host02:backup001 -G DLT01
-z /lopoint/point01 -p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up.

`-k`: Specifies the backup file type. `o` is specified because backup is to be made into a JP1/OmniBack II object.

`-b`: Specifies the name of the JP1/OmniBack II object as the backup file name. The format is host-name:object-name.

-G: Specifies the barlist file name.

-z: Specifies the name of the log point information file. Specify this option when operation without unloading the system log or the automatic log unloading facility is being used.

-p: Specifies the file name for the pdcopy command's processing results listing.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

6.7.5 Example 2 (HiRDB/Parallel Server)

RDAREAs under a unit are backed up during operation of a HiRDB/Parallel Server.

The system configuration is shown in Figure *6-3 System configuration example for backup made using JP1/OmniBack II: JP1/OmniBack II handles communications between server machines*. The RDAREAs under unit UNT2 in server machine 2 are backed up. The backup is to be stored in a JP1/OmniBack II object in server machine 3.

(1) Using the pdlogswap command to swap system log files under the unit

System log files are swapped in order to physically separate the system logs needed for database restoration. To restore the RDAREAs from the backup copy to be created in step (2), the system log collected subsequently (the system log in the file to be used as the current file after swapping) will be used as input information.

```
pdlogswap -d sys -s bes3 -w
```

(2) Using the pdcopy command to back up the unit

```
pdcopy -m /rdarea/mast/mast01 -M r -u UNT2 -k o -b host03:backup002 -G DLT02
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-u: Specifies that all RDAREAs under unit UNT2 are to be backed up.

-k: Specifies the backup file type. o is specified because backup is to be made into a JP1/OmniBack II object.

-b: Specifies the name of the JP1/OmniBack II object as the backup file name. The format is host-name:object-name.

`-G`: Specifies the barlist file name.

Remarks

When the automatic log unloading facility is being used, execute the `pdlogatul` command to record the name of the unload log file that corresponds to the current system log file when backup is being made.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

6.8 Backup acquisition using backup-hold (backup without using the pdcopy command)

This section explains how to make a backup without using the `pdcopy` command (making a backup using another product's facilities). When a backup is made without using the `pdcopy` command, the RDAREAs must be backed up and held.

For details on how to restore a database without using the `pdcopy` command, see *19.5 Recovery procedure when the backup was not made with the pdcopy command.*

6.8.1 About backup-hold

In the following cases, the target RDAREAs must be backed up and held:

- When a backup is made without using the `pdcopy` command (when backup is made using another product's facilities; e.g., when a backup is made using the backup acquisition facility of JP1/OmniBack II)
- When the Logical Volume Manager is used for the database (to execute the `pdcopy` command in the updatable mode, the target RDAREAs must be backed up and held)

When RDAREAs are backed up and held, the backup can be made even during an online session. Specify the `-b` option in the `pdhold` command to back up and hold RDAREAs.

(1) Types of backup-hold

The four types of backup-hold are explained in Table 6-5.

Table 6-5: Types of backup-hold

Backup-hold type	Explanation
Reference-possible backup-hold	During backup-hold, RDAREAs that have been backed up and held can be referenced; an updating attempt results in an SQL error (-920). *Characteristics and precautions* • Using the backup made here, the database can be restored to the backup acquisition point. • No deadlock with an updating transaction occurs. • If a no-log-mode updating transaction is executed for an RDAREA that is in reference-possible backup-hold status, the updating transaction goes into lock-release wait status for the RDAREA, causing an error after lock release. As a result, the RDAREA goes into error shutdown status.

Backup-hold type	Explanation
Reference-possible backup-hold (update WAIT mode)	During backup-hold, RDAREAs that have been backed up and held can be referenced; an updating attempt goes into lock-release wait status until the backup-hold is released. *Characteristics and precautions* • Using the backup made here, the database can be restored to the backup acquisition point. • Specify in the `pd_lck_wait_timeout` operand or the `PDCWAITTIME` operand of the client environment definition a time value that is longer than the backup-hold period. If the updating transaction times out, an SQL error (-770) occurs. • Deadlock may occur with an updating transaction. To ensure that the updating transaction does not terminate with an error, specify `pd_deadlock_priority_use = Y`. If deadlock occurs, all locks in the backup-hold collected by the `pdhold` command are released, and the same backup-hold processing is attempted again for up to five times. If deadlock results on the fifth retry, all locks in the backup-hold collected by the `pdhold` command are released, and the `pdhold` command terminates in an error. • If a no-log-mode updating transaction times out or terminates in an error after a deadlock, an updated RDAREA goes into error shutdown status.
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 in updatable backup-hold status without placing the `pdhold` command in wait status. *Characteristics and precautions* • The database cannot be restored to the backup acquisition point; it can only be restored to a synchronization point after the backup acquisition point. To restore the database, the backup and the system log from a synchronization point immediately prior to backup acquisition are required. • If an updating transaction in the pre-update log acquisition mode or the no-log mode is executed between the previous synchronization point and the current time, do not apply updatable backup-hold; the database cannot be restored using this backup.

Backup-hold type	Explanation
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 pdhold command is kept waiting until the transaction terminates. *Characteristics and precautions* • When the contents of a buffer that was updated by an update transaction occurring during backup-hold are applied to an RDAREA, the KFPH00157-W message is output when backup-hold is released. When this message has been output, the database cannot be restored to the backup acquisition point; it can only be restored to a synchronization point subsequent to the backup acquisition point. Therefore, to restore the database, you must have a system log from the time at which execution of the backup hold command began. • If the KFPH00157-W message was not output, you can use the backup made here to recover the database to its status at the time of the backup. However, if the backup was made in the pdcopy command's updatable mode (-M s specified), the database cannot be restored to the backup acquisition point; it can only be restored to a synchronization point after the backup acquisition point. To restore the database, the backup and the system log from the synchronization point immediately prior to backup acquisition are required. • An updating transaction in the pre-update log acquisition mode or the no-log mode is placed in lock-release wait status at the RDAREA that has been backed up and held. Therefore, specify in the pd_lck_wait_timeout operand or the PDCWAITTIME operand of the client environment definition a time value that is longer than the backup-hold period. If the updating transaction times out, it terminates in an error. • Deadlock may occur between an updating transaction in the pre-update log acquisition mode and an updating transaction in the no-log mode. To ensure that the updating transaction does not terminate in an error, specify pd_deadlock_priority_use=Y. If deadlock occurs, all locks in the backup-hold collected by the pdhold command are released, and the same backup-hold processing is attempted again for up to five times. If deadlock results on the fifth retry, all locks in the backup-hold collected by the pdhold command are released, and the pdhold command terminates in an error. • If an updating transaction in the pre-update log acquisition mode or the no-log mode times out or terminates in an error after a deadlock, an updated RDAREA goes into error shutdown status.

Reference note:

Reference-possible backup hold and reference-possible backup hold (update WAIT mode) are also referred to as *committing a database*; they are required when the inner replica facility is used. For details about the inner replica facility, see the manual *HiRDB Staticizer Option Version 7 Description and User's Guide*.

(2) Conditions (RDAREA configuration)

1. Place the files comprising the RDAREAs to be backed up on a disk, so that they can all be backed up in a single backup operation.

2. When creating a HiRDB file system area for RDAREAs in a regular file, specify the `-i` option in the `pdfmkfs` command.
3. To restore the database to the most recent synchronization point after an error has occurred (including restoration with a range specified), the master directory RDAREA and other RDAREAs must be restored separately. Therefore, take one of the following measures so that the master directory RDAREA can be restored separately:
   - Create a partition and logical volume for storing the master directory RDAREA only, thus enabling backup exclusively for the master directory RDAREA.
   - Use the `pdcopy` command to back up the RDAREAs in the partition containing the master directory RDAREA.

(3) Notes

1. If HiRDB is terminated abnormally or forcibly while the disk being used by HiRDB is being backed up, that backup becomes invalid. In this case, make the backup again.
2. If the RDAREA is in updatable backup hold or updatable backup hold (WAIT mode) status, RDAREA automatic extension is suppressed. For this reason, you should refrain from executing a job to add or update a large volume of data that requires allocation of new pages while the RDAREA is in either of these hold statuses. To cancel suppression of RDAREA automatic extension, use the `pdrels` command to release the RDAREA from the hold.
3. When you use a function other than the `pdcopy` command (i.e., a function provided by a different product) to make a backup, do not execute the database structure modification utility (`pdmod` command) while the backup processing is underway.
4. If HiRDB is terminated abnormally or forcibly during backup-hold, the backup-hold may not be inherited when HiRDB is restarted, depending on the backup-hold type. In this case, make a backup by implementing the backup-hold again. Table 6-6 shows the backup-hold inheritance status during HiRDB reactivation.

Table 6-6: Backup-hold inheritance states during HiRDB reactivation

Backup-hold type	Backup-hold status inheritance during reactivation
Reference-possible backup-hold	Yes
Reference-possible backup-hold (update WAIT mode)	No
Updatable backup-hold	No

Backup-hold type	Backup-hold status inheritance during reactivation
Updatable backup-hold (WAIT mode)	No

Yes: Status is inherited.

No: Status is not inherited.

(4) Relationship to lock

(a) Lock-release timeout during backup-hold

Note that a lock-release timeout might occur when the following backup-hold modes are used:

- Reference-possible backup-hold (update WAIT mode)
- Updatable backup-hold
- Updatable backup-hold (WAIT mode)

For a reference-possible backup-hold (update WAIT mode), an updating transaction waits for lock release at the RDAREA that has been backed up and held (resource type 0602) until the backup-hold is released. In the updatable backup-hold and updatable backup-hold (WAIT mode) modes, an updating transaction in the no-log mode or pre-update log acquisition mode waits for lock release at the RDAREA that has been backed up and held (resource type 0602) until the backup-hold is released.

Consequently, if the value specified in the `pd_lck_wait_timeout` operand or the `PDCWAITTIME` operand of the client environment definition is shorter than the time between backup-hold and its release, the updating transaction terminates in an error. If such an error occurs, change the specification values of these operands.

Backup acquisition in these modes is recommended when the mirror disk facility with a short duration between shutdown and release (the time needed for dividing the logical volume only) is used for making backups.

(b) Deadlock during backup-hold

Note that a deadlock might occur when backup and hold is applied to multiple RDAREAs in the following modes:

- Reference-possible backup-hold (update WAIT mode)
- Updatable backup-hold
- Updatable backup-hold (WAIT mode)

In the reference-possible backup-hold (update WAIT mode) mode, deadlock with an updating transaction may occur. In the updatable backup-hold and updatable backup-hold (WAIT mode) modes, deadlock may occur with an updating transaction

in the no-log mode or pre-update log acquisition mode. To ensure that the updating transaction does not terminate in an error, specify `pd_deadlock_priority_use = Y`. In this case, if deadlock occurs, the `pdhold` command terminates in an error. Therefore, re-execute the `pdhold` command after a short while. If multiple RDAREAs are specified in the `pdhold` command, backup-and-hold processing for all RDAREAs becomes invalid. If multiple RDAREAs are backed up and held using the `pdhold` command multiple times, deadlock may occur in the RDAREAs that have been backed up and held. In this case, first release the RDAREAs that have been backed up and held and then re-execute the `pdhold` command. Therefore, when applying backup and hold to multiple RDAREAs, specify multiple RDAREAs in a single `pdhold` command (repeat execution of the command until successful).

In the case of a HiRDB/Parallel Server, global deadlock can occur if RDAREAs on different servers are specified. HiRDB cannot detect global deadlock, so the transactions result in timeout errors. To avoid global deadlock, execute the `pdhold` command in units of servers when you specify more than one RDAREA.

(5) Volume of system log information output during updatable backup-hold

When the database is updated during updatable backup-hold, the volume of system log information that is output increases by the amount determined by the following formula:

$$\sum_{i=1}^{a} (Si + 200) \times Ti$$

Note

If HiRDB is terminated abnormally or forcibly during backup-hold, the backup-hold is not inherited during HiRDB reactivation (except when reference-possible backup-hold is used). Consequently, the amount of system log information indicated by this formula is not output in this case.

a: Number of RDAREAs updated during updatable backup-hold

S_i: RDAREA page size (bytes)

T_i: Number of pages updated in the RDAREAs updated during updatable backup-hold

The number of updated pages in the RDAREAs is determined according to the following procedure.

Procedure

To determine the number of updated pages in the RDAREAs:

1. Execute the statistics analysis utility (HiRDB file statistical information related to database operations) at the following times:

- At the start of updatable backup-hold
- At the release of updatable backup-hold

2. Refer to the number of synchronous WRITEs (SYNC-W) and determine the number of updated pages from the difference.

(6) Determining the system logs needed for database restoration

To restore the database to the most recent synchronization point (including restoration with a range specified), the system logs after the backup acquisition are needed. Table 6-7 shows how to determine the system logs needed for database restoration.

Table 6-7: Determining the system logs needed for database restoration (when backup-hold is being used)

Condition	Determining the system logs needed for database restoration
• Referencing-possible backup-hold • Referencing-possible backup-hold (update WAIT mode)	The system log subsequent to when the backup was made is needed. Before making a backup, execute the `pdlogls` command to record the following information about the current system log: • System log service run ID (Run ID) • System log file generation number (Gen No.) • System log file's file group name (Group) Also record the date and time of backup. Specify this time if the `-T` option is to be specified in the `pdrstr` command.
Updatable backup hold (WAIT mode)	The system log from the point at which execution of the backup hold command began is required. Before executing backup hold, execute the `pdlogls` command to record the following information: • System log service run ID (Run ID) • System log file generation number (Gen No.) • System log file's file group name (Group) Also record the date and time at which execution of backup hold began. Specify this time if the `-T` option is to be specified in the `pdrstr` command.

Condition	Determining the system logs needed for database restoration
Updatable backup-hold	Refer to the KFPS02183-I message, which indicates the starting position of the system logs needed for database restoration.* The system log file file group name, generation number, and synchronization point dump acquisition start time indicated in the KFPS02183-I message output before the updatable backup-hold indicate the starting position of the system logs needed for database restoration. The system log service run ID can be checked with the pdlogls command. Specify the synchronization dump acquisition start time if the -T option is to be specified in the pdrstr command.
The Logical Volume Manager (LVM) is used and the pdcopy command is executed in the updatable mode.	Refer to the pdcopy command's processing results listing.

* When pd_spd_assurance_msg=N is specified, the KFPS02183-I message is not output.

6.8.2 Example 1 (using another product's backup facilities)

Back up RDAREAs (rdarea01 and rdarea02) while HiRDB is operating. Use another product's backup facility to back up the HiRDB file system area that comprises these RDAREAs.

(1) Using the pdhold command to back up and hold the RDAREAs to be backed up

```
pdhold -r rdarea01,rdarea02 -b -w
```

Explanation

-r: Specifies the RDAREAs to be backed up.

-b -w: Specifies reference-possible backup hold (update WAIT mode).

(2) Making the backup

Use another product's backup facility to back up the HiRDB file system area that comprises the RDAREAs (rdarea01 and rdarea02).

Remarks

When the automatic log unloading facility is being used, execute the pdlogatul command to record the unload log file name that corresponds to the current

system log file.

(3) Using the pdrels command to release backup and hold of RDAREAs

```
pdrels -r rdarea01,rdarea02
```

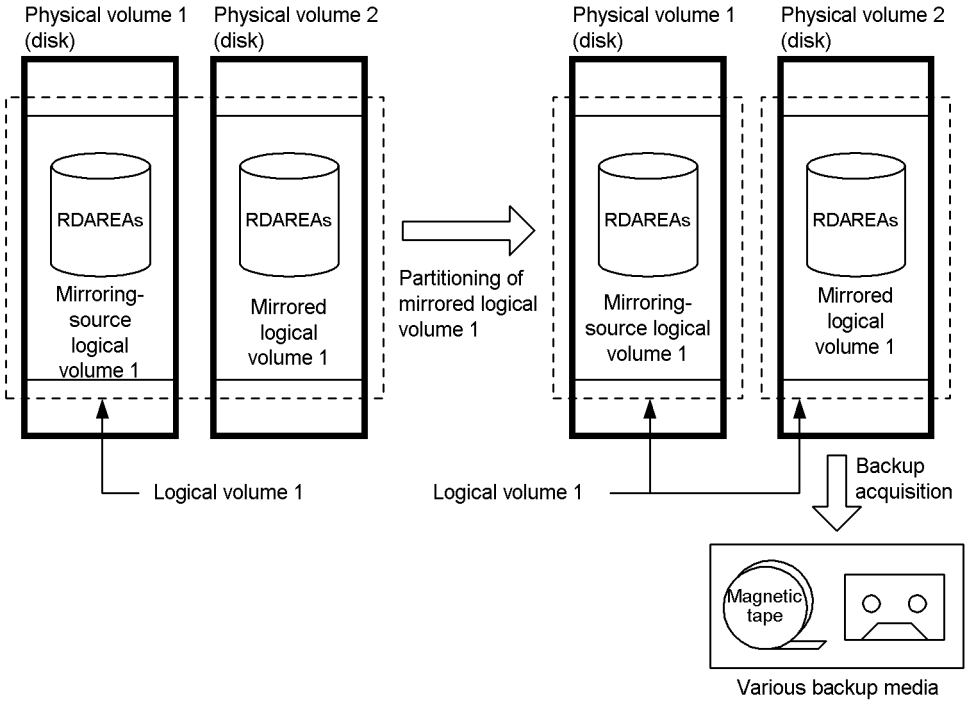
It is recommended that, after the command has executed, you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

6.8.3 Example 2 (using the mirror disk facility)

This example assumes that data is mirrored using the Logical Volume Manager (LVM) and a mirror facility (it is assumed that the reader is familiar with LVM). For details on data mirroring using LVM and execution of online backup through logical volume partitioning, see the appropriate HP-UX manual). Figure 6-7 shows an example of backup using a mirror facility.

To avoid input/output contention between the backup disk and the online disk when mirroring is used, separate the I/O channels.

Figure 6-7: Example of backup using a mirror facility



The two RDAREAs rdarea01 and rdarea02 are backed up while HiRDB is running.

(1) Using the pdhold command to back up and hold the RDAREAs to be backed up

```
pdhold -r rdarea01,rdarea02 -b -w
```

Explanation

`-r`: Specifies the RDAREAs to be backed up.

`-b -w`: Specifies reference-possible backup hold (update WAIT mode).

(2) Partitioning the mirrored logical volume

The logical volume containing the HiRDB files comprising the RDAREAs to be backed up is partitioned.

(3) Using the pdrels command to release backup and hold of RDAREAs

```
pdrels -r rdarea01,rdarea02
```

(4) Making the backup from the offline logical volume

The backup is made from the offline logical volume. When a mirror facility is used to make a backup, backup-hold occurs only during logical volume partitioning. Therefore, the amount of physical log information for the pages to be output is reduced.

Remarks

When the automatic log unloading facility is being used, execute the `pdlogatul` command to record the unload log file name that corresponds to the current system log file.

6.9 Backup acquisition when the frozen update command (pddbfrz command) is used

The function that supports reducing the amount of time required to back up user LOB RDAREAs provides the frozen update command (`pddbfrz` command). This section describes how to back up user LOB RDAREAs when you use the frozen update command.

6.9.1 Operation subject to the frozen update command

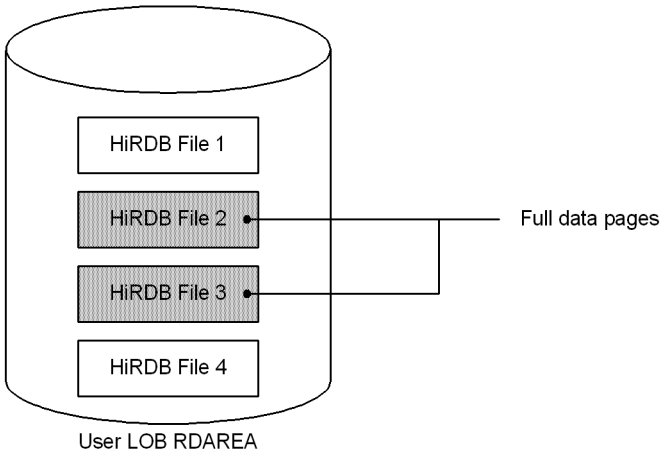
Under the following operational conditions, the frozen update command (`pddbfrz` command) can be used in conjunction with backing a up user LOB RDAREA as quickly as possible:

- A method other than the database copy utility (`pdcopy` command) is used to back up the user LOB RDAREA.
- Backup is in units of HiRDB files.
- Updating and deletion of registered LOB data do not normally occur (the typical operation is to add LOB data).
- The user LOB RDAREA to be backed up consists of multiple HiRDB files.

6.9.2 Operation of the frozen update command (pddbfrz command)

When the frozen update command is executed, all HiRDB files with full data pages (completely allocated pages) in the user LOB RDAREA are placed in frozen update status. Data in HiRDB files that are in frozen update status cannot be updated or deleted. Figure 6-8 provides an overview of the processing by the frozen update command.

Figure 6-8: Overview of frozen update command processing



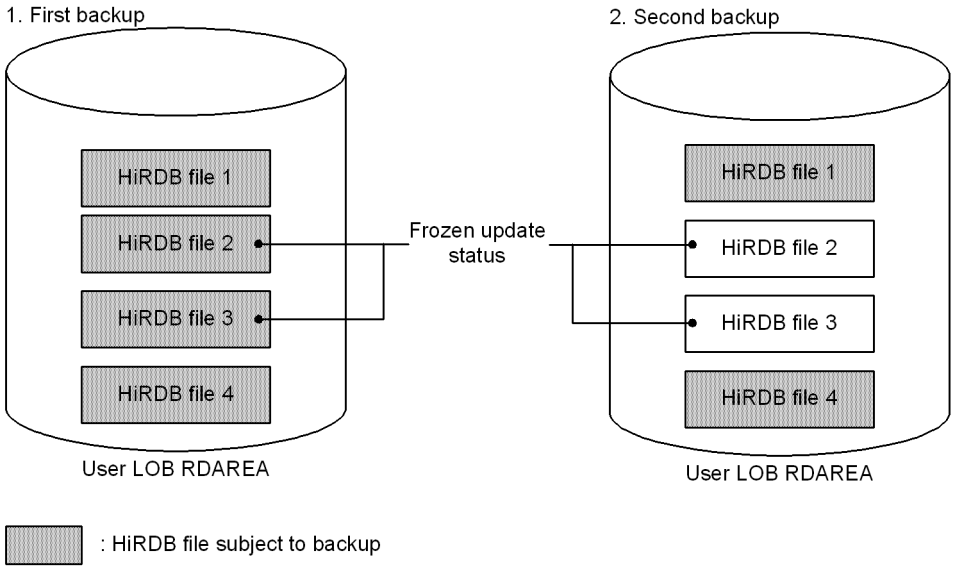
Explanation

- The user LOB RDAREA consists of HiRDB files 1-4. Files 2 and 3 have full data pages. For details about checking for HiRDB files with full data pages, see *6.9.4 Checking for files with full data pages*.
- When the frozen update command is executed for this user LOB RDAREA, HiRDB files 2 and 3 are placed in frozen update status. HiRDB files placed in frozen update status are displayed in the `KFPH27024-I` message.
- HiRDB files 1 and 4 are in permit update status.

(1) Using the frozen update command in order to make backups

Figure 6-9 shows how to use the frozen update command in order to make backups.

Figure 6-9: Using the frozen update command in order to make backups



Explanation

The frozen update command is executed before making backups, thereby placing HiRDB files 2 and 3 in frozen update status.

1. Back up all HiRDB files (HiRDB files 1-4) when the first backup is made.
2. Because HiRDB files 2 and 3 are in frozen update status, their contents will not change after that first backup is made. Thus, it will not be necessary to back up HiRDB files 2 and 3 when the next backup is made; only HiRDB files 1 and 4 will need to be backed up.

Remarks

Because the lead HiRDB file (HiRDB file 1 in Figure 6-9) contains management records, write transactions always occur in it even if its data portion is full. Consequently, the lead HiRDB file is never placed in frozen update status and must always be backed up.

(2) Procedure for making a backup

The procedure for making a backup is described below. For an operation example, see *6.9.3 Operation example*.

Procedure

1. Use the frozen update command to place the HiRDB files with full data pages in frozen update status.

2. Use the KFPH27024-I message or the database condition analysis utility (pddbst command) to check the HiRDB files that are in frozen update status.
3. Make a backup. Include all HiRDB files in the first backup. In the second and subsequent backups, include only the HiRDB files that are in permit update status; it will not be necessary to include the HiRDB files that are in frozen update status.

(3) Releasing frozen update status

To release a HiRDB file from frozen update status, execute the pddbfrz -d command.

Reinitializing RDAREAs also releases frozen update status.

(4) Checking for HiRDB files in frozen update status

Execute the database condition analysis utility (the pddbst command) to check which HiRDB files are in frozen update status.

(5) HiRDB files that cannot be placed in frozen update status

The following HiRDB files cannot be placed in frozen update status:

- HiRDB files in other than user LOB RDAREAs
- Lead HiRDB files in user LOB RDAREAs
- HiRDB files in user LOB RDAREAs that store plug-in indexes
- HiRDB files in unused user LOB RDAREAs (without any table definitions)
- HiRDB files with logical files created by data type plug-ins*

* In the case of user LOB RDAREAs used by data type plug-ins, HiRDB files with logical files used by plug-ins cannot be placed in frozen update status. A logical file is normally found only in the lead HiRDB file, but when expansion of the logical files occurs, a logical file may be found in an HiRDB file other than the lead file. Therefore, if you are using data type plug-ins, you should estimate the size for the lead HiRDB file so that it can store logical files for plug-ins created by CREATE TABLE.

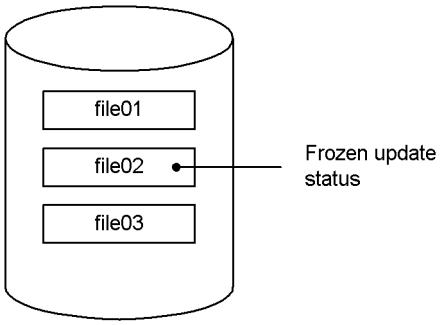
6.9.3 Operation example

This section provides an example of backing up a user LOB RDAREA (LOB001) under the following operational conditions:

- The user LOB RDAREA (LOB001) consists of three HiRDB files.
- As data is added, HiRDB files are added and the user LOB RDAREA (LOB001) is expanded.

(1) Execute the frozen update command on the user LOB RDAREA (LOB001)

```
pddbfrz -r LOB001
```

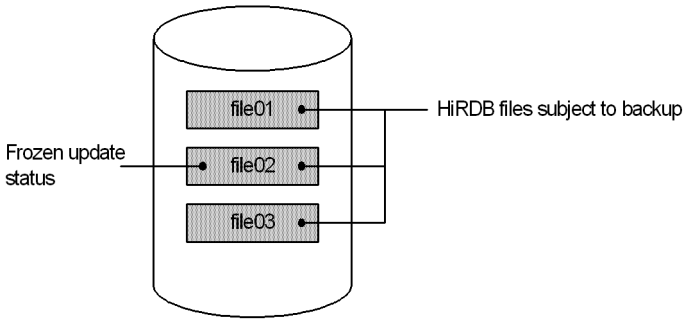


Explanation

The HiRDB file with full data pages (`file02`) is placed in frozen update status. Refer to the `KFPH27024-I` message or use the database condition analysis utility (`pddbst` command) to confirm the HiRDB files that have been placed in frozen update status.

(2) Make a backup

Back up the user LOB RDAREA (`LOB001`) using some method other than the database copy utility. Make the backup in units of the HiRDB files comprising the user LOB RDAREA (`LOB001`).



Explanation

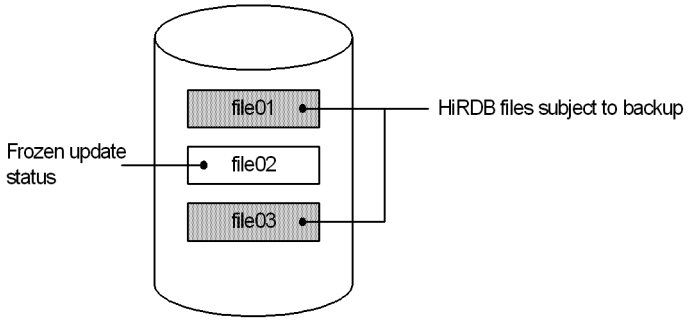
All the HiRDB files are backed up (`file01`, `file 02`, and `file 03`).

For details about making a backup using methods other than the database copy utility (pdcopy command), see *6.8 Backup acquisition using backup-hold (backup without*

using the pdcopy command).

(3) Make backups regularly

Make backups on a regular schedule, such as once a week.

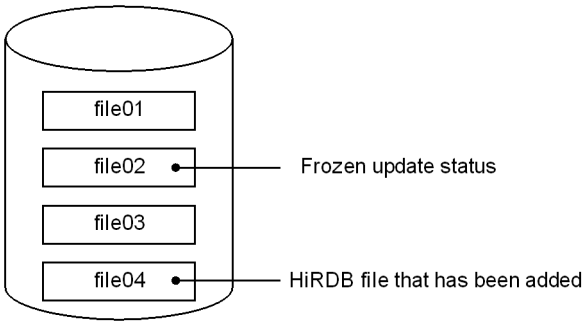


Explanation

Only the HiRDB files in permit update status are backed up (`file01` and `file 03`). You do not need to back up the HiRDB file in frozen update status (`file02`).

(4) Add HiRDB files and expand the user LOB RDAREA (LOB001)

For details about expanding the user LOB RDAREA (`LOB001`), see *15.3 Increasing the size of an RDAREA (RDAREA expansion)*.

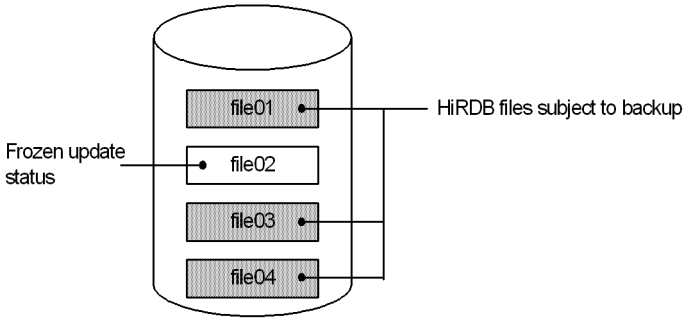


Explanation

A HiRDB file (`file04`) is added and the user LOB RDAREA (`LOB001`) is expanded.

(5) Make a backup

After expanding the user LOB RDAREA (`LOB001`), back it up. Always make a backup, even if the number of segments used in a file is 0; for the reason, see *6.9.7 Notes*.



Explanation

HiRDB files in permit update status (`file01`, `file 03`, and `file 04`) are backed up. Backing up the HiRDB file in frozen update status (`file 02`) is not necessary.

(6) Use the database condition analysis utility (pddbst command) to check for full data pages

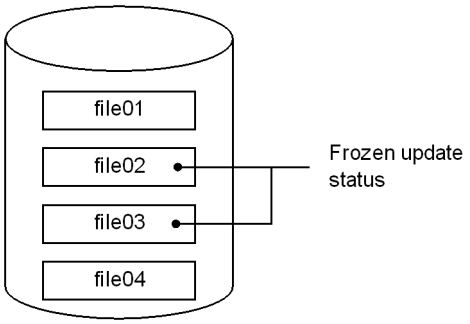
When LOB data is added, additional HiRDB files with full data pages may be issued. Use the database condition analysis utility (`pddbst` command) to check for files with full data pages. For details about checking for files with full data pages, see *6.9.4 Checking for files with full data pages*.

```
pddbst -r LOB001
```

(7) If more files with full data pages have been issued, execute the frozen update command

If more files with full data pages have been issued, as determined in step (6), execute the frozen update command.

```
pddbfrz -r LOB001
```

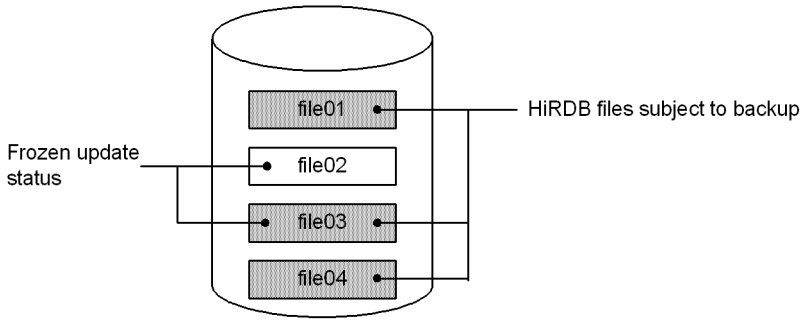


Explanation

HiRDB files with full data pages (`file02` and `file03`) are placed in frozen update status. Refer to the `KFPH27024-I` message or use the data condition analysis utility (`pddbst` command) to confirm the HiRDB files that have been placed in frozen update status.

(8) Make a backup

Back up the user LOB RDAREA.

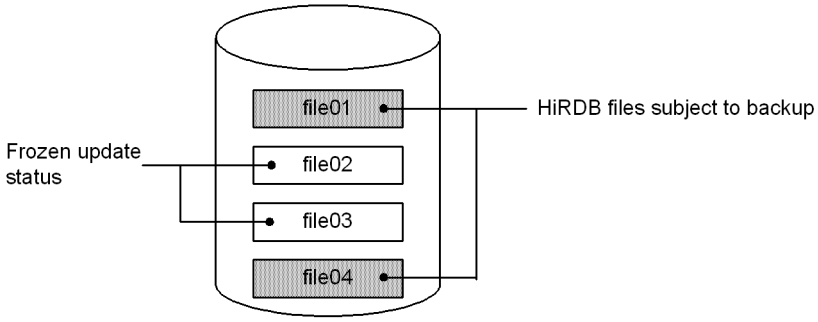


Explanation

HiRDB files `file01`, `file03`, and `file04` are backed up; it is not necessary to back up HiRDB file `file02`.

(9) Make backups regularly

Make backups on a regular schedule, such as once a week.



Explanation

Only the HiRDB files in permit update status are backed up (`file01` and `file04`). You do not need to back up the HiRDB files in frozen update status (`file02` and `file03`).

6.9.4 Checking for files with full data pages

You use the database condition analysis utility (pddbst command) to execute physical analysis of RDAREAs in order to determine if there are HiRDB files that are full.

■ Execution example of the database condition analysis utility

```
RD Area Name   : LOB001
    :
    :
 Freeze Specified : Y
  HiRDB File Name : /rdarea/lob001/file01
   File Size      :       8192 segments   Extent Count :   1/ 24
   Segment        :       8190/      8192
   Freeze Status  : U
  HiRDB File Name : /rdarea/lob001/file02
   File Size      :       8192 segments   Extent Count :   1/ 24
   Segment        :       8000/      8192
   Freeze Status  : F    2001/12/06    20:37:05
  HiRDB File Name : /rdarea/lob001/file03
   File Size      :       8192 segments   Extent Count :   1/ 24
   Segment        :       1000/      8192
   Freeze Status  : U
```

Explanation

The information in the underlined portions above tells you that the HiRDB file currently being used (where date is being stored currently) is `/rdarea/lob001/file03`. Therefore, the previous HiRDB file (`/rdarea/lob001/file02`) has become full.

Remarks

- LOB data is stored in consecutive pages as much as possible, so if data

will not fit onto a single page, there may be some pages at the end of the HiRDB file that are not usable, as shown in this example.

- When LOB data is deleted, the page that was used does not become 100% free. This is why the example shows 8000 out of 8192 segments.
- When the page of the HiRDB file being used (such as in `/rdarea/lob001/file03`) is the last page, that HiRDB file is also considered to be full.

Other items in the example that relate to frozen update status are explained below.

`Freeze Specified`: Indicates whether or not the frozen update command has been executed for the RDAREA:

`Y`: Frozen update command has been executed.

`N`: Frozen update command has not been issued.

`Freeze Status`: Indicates whether the HiRDB file is in frozen update status of permit update status:

`F`: File is in frozen update status.

`U`: File is in permit update status.

6.9.5 Manipulating user LOB RDAREAs for which the frozen update command has been executed

Some manipulations cannot be executed on user LOB RDAREAs for which the frozen update command has been executed, as indicated in Table 6-8. If you must execute on a LOB RDAREA a manipulation that Table 6-8 shows cannot be executed, use the frozen update command with the `-d` option specified to release the user LOB RDAREA's frozen update status. After executing the manipulation, you must back up all HiRDB files in the user LOB RDAREAs again.

Table 6-8: Manipulations executable on user LOB RDAREAs for which the frozen update command has been executed

Manipulation		Status of HiRDB files	
		Frozen update status	Permit update status
SQL (data manipulation)	`INSERT`	—	Yes
	`UPDATE`	No	Yes
	`DELETE`	No	Yes
SQL (table manipulation)	`PURGE TABLE`	No	No
	`DROP TABLE`	No	No

Manipulation		Status of HiRDB files	
		Frozen update status	Permit update status
Database load utility	Add mode	—	Yes
	Load mode	No	No
Database reorganization utility (table reorganization, table reloading)		No	No
Rebalancing utility		No	No
Database structure modification utility		Yes	Yes
Database recovery utility		Yes	Yes
Operation command		Yes	Yes

Legend:

Yes: Can be executed

No: Cannot be executed

—: Not applicable.

6.9.6 Relationship between RDAREAs and automatic extensions

If an automatic extension is set up for the last HiRDB file of a user LOB RDAREA and that HiRDB file is placed in frozen update status, the automatic extension of the RDAREA is disabled. If you add HiRDB files and extend the user LOB RDAREA, the automatic extension is operable.

You cannot set up an RDAREA automatic extension if the last HiRDB file of the user LOB RDAREA is in frozen update status. You must add HiRDB files and set up the RDAREA automatic extension when you extend the user LOB RDAREA.

6.9.7 Notes

(1) HiRDB files to be backed up

You should back up HiRDB files in frozen update status at least once. The management area of HiRDB files that are not in frozen update status is used regardless of whether data pages are being used. Even if the database condition analysis utility (`pddbst` command) is executed and the results indicate that no HiRDB files are in use, be sure that you always have a backup of the HiRDB files.

(2) Timing at which status changes to frozen update status

The status of the HiRDB files changes only when the frozen update command is

executed. If a HiRDB file does not have full data pages when the frozen update command is executed but its data pages become full later after data has been added, its status will not change (that is, it will not change to frozen update status). The status of such a HiRDB file will change only if you execute the frozen update command again after the file's data pages have become full.

Chapter

7. Operation Without Acquiring a Database Update Log

This chapter describes the procedures for executing a UAP or utility in the pre-update log acquisition and no-log modes.

This chapter contains the following sections:

7.1 Database update log acquisition modes

HiRDB collects in a system log file historical information on database updating by UAPs and utilities*. The user can specify that such a database update log not be collected. Processing time is reduced when a database update log is not collected, thereby reducing the UAP or utility execution time.

* Applies to the following utilities only:

- Database load utility
- Database reorganization utility
- Rebalancing utility

(1) Database update log acquisition modes

Three database update log acquisition modes are available during UAP or utility execution:

Log acquisition mode

Database update log information necessary for rollback and rollforward is acquired.

Pre-update log acquisition mode

Database update log information necessary for rollback only is acquired.

No-log mode

Database update log information is not acquired.

(2) Specifying the database update log acquisition mode

Table 7-1 lists the database update log acquisition modes.

Table 7-1: Database update log acquisition modes

Condition	Specification of database update log acquisition mode
UAP	Specify the database update log acquisition mode in the `PDDBLOG` operand of the client environment definition. The log acquisition mode or the no-log mode can be specified; the pre-update log acquisition mode cannot be specified.
Database load utility	Specify the database update log acquisition mode in the `-l` option of the database load utility.
Database reorganization utility	Specify the database update log acquisition mode in the `-l` option of the database reorganization utility.

Condition	Specification of database update log acquisition mode
Rebalancing utility	Specify the database update log acquisition mode in the -l option of the rebalancing utility. The log acquisition mode or the no-log mode can be specified; the pre-update log acquisition mode cannot be specified.
User LOB RDAREA being used	For the data stored in a user LOB RDAREA, specify the database update log acquisition mode in the RECOVERY operand of CREATE TABLE.

(3) RECOVERY operand

The database update log acquisition mode specified with the RECOVERY operand may be changed by the specification of the PDDBLOG operand or the -l option. Table 7-2 shows the relationship between the RECOVERY operand and the PDDBLOG operand or -l option.

Table 7-2: Relationship between the RECOVERY operand and the PDDBLOG operand or -l option

PDDBLOG operand or -l option specification	RECOVERY operand specification	Value assumed during UAP (or utility) execution
ALL a (Log acquisition mode)	ALL	ALL
	PARTIAL	PARTIAL
	NO	NO
p (Pre-update log acquisition mode)*	ALL	PARTIAL
	PARTIAL	PARTIAL
	NO	NO
NO n (No-log mode)	ALL	NO
	PARTIAL	NO
	NO	NO

ALL or a: Log acquisition mode

PARTIAL or p: Pre-update log acquisition mode

NO or n: No-log mode

* The pre-update log acquisition mode cannot be specified with the PDDBLOG operand.

(4) Operational differences depending on the database update log acquisition mode

The database update log acquisition mode that is used determines how the following

operations are performed:

- HiRDB processing and administrator actions when a UAP or utility terminates abnormally
- Database recovery point

(a) HiRDB processing and administrator actions when a UAP or utility terminates abnormally

Table 7-3 describes the HiRDB processing and administrator actions when a UAP or utility terminates abnormally.

Table 7-3: HiRDB processing and administrator actions when a UAP or utility terminates abnormally

Database update log acquisition mode	HiRDB processing	User's action
Log acquisition mode	Rolls back any updated RDAREA to its status before the UAP executed or to its status immediately before the abnormal termination.	If the RDAREA was rolled back to its status before the UAP executed, re-execute the UAP. If it was rolled back to its status immediately before the abnormal termination, re-execute the processing subsequent to the synchronization point.
Pre-update log acquisition mode		
No-log mode	Does not perform rollback. Places any updated RDAREA in error shutdown status (logless hold). The contents of the RDAREA are damaged.	Use the database recovery utility to restore the RDAREA from a backup made before the UAP executed, and then re-execute the UAP.

(b) Database recovery point

Table 7-4 shows the point to which the database can be recovered by the database recovery utility.

Table 7-4: Database recovery point

Database update log acquisition mode	Database recovery point
Log acquisition mode	Backup acquisition point or any synchronization point subsequent to the backup acquisition point
Pre-update log acquisition mode	Backup acquisition point
No-log mode	

(5) Notes on backup (important)

1. Do not make a backup in the updatable mode (`-M s` specified) during execution of a UAP (including utilities) in the no-log mode or the pre-update log acquisition

mode.

2. After execution of a UAP (including utilities) 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)

7.2 Procedure for executing a UAP or utility in the pre-update log acquisition mode

Executor: HiRDB administrator

This section explains the procedure for executing a UAP or utility in the pre-update log acquisition mode. In the case of a UAP, the pre-update log acquisition mode can be used only for a user LOB RDAREA (`RECOVERY` operand of `CREATE TABLE`).

This section focuses on the operations when a utility is executed in the pre-update log acquisition mode.

(1) Advantages

In this mode, database update logs are not collected after an RDAREA is updated, so processing time is reduced by a comparable amount of time, which reduces UAP or utility execution time (as compared with execution in the log acquisition mode).

(2) Criteria

This is the default mode; its use is recommended when the database load utility or database reorganization utility is executed.

(3) Notes (important)

(a) Back up any updated RDAREAs after you execute the UAP or utility

If the system log information used as input by the `pdrstr` command for recovery of an RDAREA includes any logs that were collected in the pre-update log acquisition mode, the `pdrstr` command will result in an error. Consequently, unless you back up the RDAREA before executing the UAP or utility, you will not be able to recover that RDAREA to its most recent status, if the need arises to do so (with the `pdrstr` command). This means that you will not be able to restore updates that occurred after the UAP or utility executed; you will only be able to restore the RDAREA to its status before the UAP or utility executed.

(b) Ensure no other users can update the RDAREA you are updating

Assume that tables `T1` and `T2` are stored in `RDAREA1`, table `T1` is updated in the pre-update log acquisition mode and table `T2` is updated in the log acquisition mode. If the need arises to recover `RDAREA1` with the `pdrstr` command, it will not be possible to restore this RDAREA to its most recent status because the `pdrstr` command can only recover table `T1` to a synchronization point prior to the updating. This is because execution of the `pdrstr` command results in an error if the system log information used for input includes any logs collected in the pre-update log acquisition mode.

Therefore, before executing a utility, use the `pdhold` command to place the RDAREA

to be updated on hold. For a UAP, do not update an RDAREA that may be updated by other users.

(4) Operating procedure

For details about using the database reorganization utility in the pre-update log acquisition mode, see *13.2.3 Selecting an update log acquisition mode for a database* and *13.3 Reorganizing a table (examples)*.

7.3 Procedure for executing a UAP or utility in the no-log mode

Executor: HiRDB administrator

This section explains the procedure for executing a UAP or utility in the no-log mode.

(1) Advantages

Because database update log information is not collected, the processing time is reduced, thereby reducing the UAP or utility execution time.

(2) Criteria

1. When a UAP that adds, updates, or deletes a large amount of data is to be executed
2. When a large amount of data is loaded
3. When table data involving a large number of tables is reorganized (reloaded)
4. When RDAREAs that are subject to UAP update processing in the no-log mode can be placed in exclusive use status

(3) Notes (important)

1. When `PDDBLOG=NO` is specified in the client environment definition, a table that is updated by a UAP running in the no-log mode is placed in lock mode (`EX`). Such a table must not be updated concurrently by another UAP.
2. When `PDDBLOG=NO` is specified (or specification is omitted) in the client environment definition and an RDAREA contains a user LOB RDAREA that is to be updated by a UAP running in the no-log mode, the user LOB RDAREA must not be updated concurrently by another UAP if `NO` is specified in the `RECOVERY` operand in the table definition. If it is updated by another UAP and the UAP in the no-log mode terminates abnormally, the update by the other UAP is invalidated.
3. If a UAP or utility that is running in the no-log mode terminates abnormally, any RDAREA storing a table updated by the UAP or utility is placed in error shutdown status (logless hold). HiRDB will not restore such an RDAREA; it must be restored by the HiRDB administrator. Therefore, any RDAREA placed in error shutdown status by a no-log mode UAP or utility that terminated abnormally cannot be accessed by other UAPs or utilities until the HiRDB administrator has restored it. In addition, the RDAREA can be restored only to its status at the time a backup was made.
4. Synchronization point dumps are not collected while a UAP or utility is executing in the no-log mode. If a system failure occurs while a UAP or utility is executing in the no-log mode concurrently with another UAP or utility, more time is required for system restart. Therefore, while a UAP or utility is being executed in

the no-log mode, it is best that no other UAP or utility be executing.

(4) Relationship with other facilities

1. When `CREATE INDEX` is executed by the database definition utility and the table for which the index is to be created contains row data, the index is created in the batch mode during execution of `CREATE INDEX`. If `PDDBLOG=NO` is specified in the client environment definition in such a case, the index will be created in the batch mode without database update log information being collected. The HiRDB administrator must therefore apply the operating procedures for the no-log mode.
2. An RDAREA's error shutdown status is inherited at the next normal startup of HiRDB. Its open or closed status is also inherited.

(5) Operating procedure

For details about using the database reorganization utility in the no-log acquisition mode, see *13.2.3 Selecting an update log acquisition mode for a database* and *13.3.7 Example 7: Reorganizing in no-log mode*.

The following is the operating procedure for executing a UAP in the no-log mode:

Procedure

1. Use the `pdcopy` command to back up the RDAREA to be updated.
2. Execute the UAP in the no-log mode.
3. Use the `pdlogswap` command to swap system log files.
4. Use the `pdcopy` command to back up the updated RDAREA.

Note

If there are backup and system log files that can be used to restore the RDAREA to its status before the UAP was executed, you do not need to make the backup described in step 1. An exception to this is that you must always back up any user LOB RDAREA for which a plug-in index has been created in the batch mode.

(6) In the event of abnormal termination of a UAP or utility

When a UAP or utility terminates abnormally, any RDAREA that the abnormally terminated UAP or utility has updated is placed in error shutdown status (logless hold). The HiRDB administrator must recover RDAREAs that are in error shutdown status. For details about recovering RDAREAs, see *19.2 Recovering a database to the point at which a backup was made*.

(a) Identifying RDAREAs placed in error shutdown status

The `pddbls` command can be used to identify RDAREAs that have been placed in error shutdown status.

(b) When data was not backed up before a UAP or utility was executed

When the data was not backed up before a UAP or utility was executed, a *range specification* must be used to recover the applicable RDAREAs; For details on recovery using a range specification, see *19.3 Recovering a database to the most recent synchronization point*.

- The input information for such recovery is the most recent backup copy of the RDAREAs that are in error shutdown status and the unload log file containing the information collected subsequently to the backup (system log file).
- The UAP's execution start time is specified as the recovery end time in the `-T` option of the database copy utility.
- After the database has been recovered by this procedure, the UAP or utility can be re-executed.

(c) Restoring an RDAREA from its initial data (in the event of abnormal termination of a UAP)

The following is the procedure for recovering an RDAREA from its initial data without using the database recovery utility. However, if the program that terminated abnormally is a utility, this procedure cannot be used to recover an affected RDAREA.

Procedure

1. Use the `PURGE TABLE` statement to delete all table row data that was updated by the UAP.
2. Use the `pdrels` command to release the RDAREA from error shutdown status.
3. Use the database load utility (`pdload` command) to store the initial data in the table again.

(d) Restoring an RDAREA from its initial data (in the event of abnormal termination of a utility)

The following is the procedure for recovering an RDAREA from its initial data without using the database recovery utility when the program that terminated abnormally is a utility.

Procedure

1. Use the `pdclose` command to close the RDAREA to be recovered.
2. Use the `pdmod` command to re-initialize the RDAREA.
3. Use the `pdopen` command to open the RDAREA.
4. Use the `pdload` command to reload the initial data into the table.
5. Use the `pdrels` command to release the RDAREA from error shutdown

status.

Chapter

8. Obtaining the System Operating Environment (Monitoring the System Status)

This chapter explains how to obtain information about the system operating environment (monitoring the system status).

This chapter contains the following sections:

8.1 Using the message log to check the system execution status

Executor: HiRDB administrator

The system's execution status can be determined by checking the message log. The message log provides a historical record of HiRDB messages for the following purposes:

- For checking message issued by HiRDB
- For checking on errors or the causes of an error in the event of a HiRDB failure

8.1.1 Referencing the message log (message log output destination)

Messages that are issued by HiRDB are output as message logs to message log files (under the file name `$PDDIR/spool/pdlog1` or `pdlog2`) and to `syslogfile`.

You can view messages that have been output to message log files by executing the `pdcat` command. To view messages that have been output to `syslogfile`, you must use the procedure provided by the OS.

Reference note:

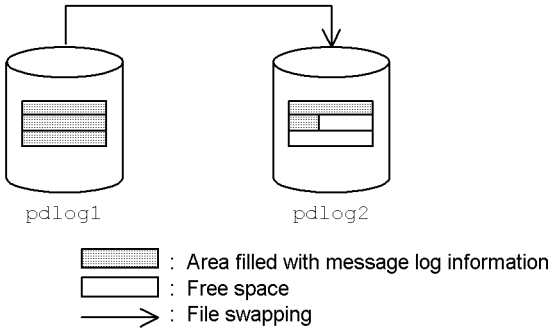
- When the `pdcat` command is executed, HiRDB merges the message log information contained in the message log files and outputs the messages in chronological order.
- For a HiRDB/Parallel Server, you can select a message log output method (whether to output to the server machine of the system manager or to each server machine). For details, see *8.1.3 Selecting a message log output method (applicable only to a HiRDB/Parallel Server)*.

8.1.2 Using the message log files

HiRDB uses two message log files (`pdlog1` and `pdlog2`) alternately. When one message log file becomes full, HiRDB stops writing messages to that file and starts writing messages to the other file. When this occurs, existing messages in the latter file are erased.

Figure 8-1 shows message log file swapping.

Figure 8-1: Message log file swapping



Explanation

When the `pdlog1` message log file becomes full, the message log output destination is switched to `pdlog2`. When this occurs, existing messages in `pdlog2` are erased.

Reference note:

- To determine which message log file is the current file, use the OS's `ls` command to check the date and time of the most recent update in each file. The file with the most recent update date and time is the current message log file.
- Message log files are not swapped at the time of HiRDB startup (all startup modes). Messages continue to be output to the message log file that was the output destination when HiRDB was previously terminated.

■ Action to be taken by the HiRDB administrator when message log files are swapped

When the message log files are swapped, the `KFPS01910-I` message is output to the message log file that is about to be swapped. To save the message logs in the message log file that is about to be swapped, you must make a backup of that message log file.

■ Changing the message log file size

You use the `pd_mlg_file_size` operand to specify the message log file size. Change the size as needed.

8.1.3 Selecting a message log output method (applicable only to a HiRDB/Parallel Server)

If the system manager unit shuts down or a communication error occurs between the system manager unit and other units, message logs cannot be output normally and the following problems may arise:

- No messages are displayed.
- The order in which messages are displayed changes.
- Message log information is output to `syslogfile` at the unit that issued the message, but its output is delayed.

To avoid these problems, you should consider use of message log output dispersion.

Reference note:

For a HiRDB/Parallel Server, message log files are output normally to the server machine where the system manager is defined.

(1) Message log output dispersion

For a HiRDB/Parallel Server, you can select one of the following message log output methods (message log output destinations):

1. Message logs are output to the message log files and to `syslogfile` of the server machine where the system manager is defined.
2. Message logs are output to the message log files and to `syslogfile` of a server machine. In this case, a message is output to the server machine that issued the message.

Normally, message logs are output using method 1. If desirable, you may change to method 2. Method 2 is called *message log output dispersion*.

You use the `pd_mlg_msg_log_unit` operand to select the desired method. If this operand is omitted, the default is method 1.

Figures 8-2 and 8-3 show the message output methods.

Figure 8-2: Normal message log output method (method 1)

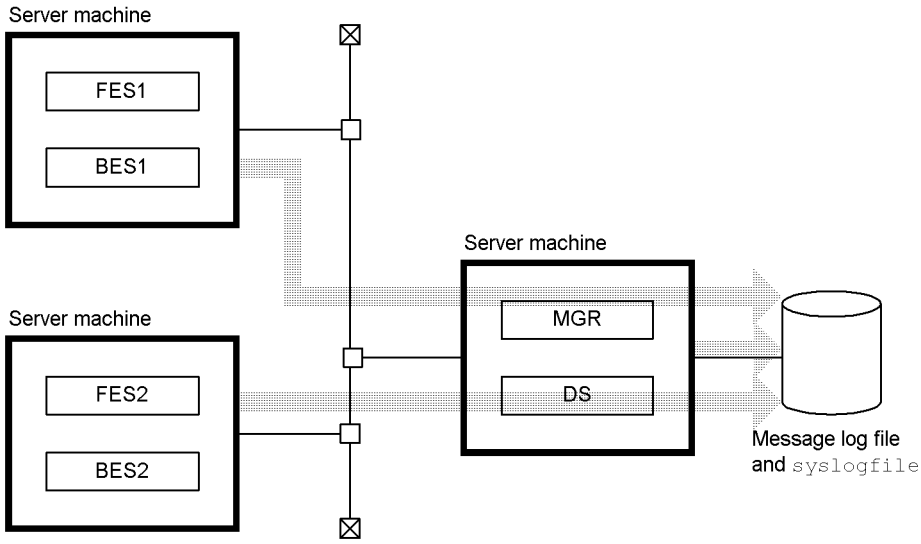
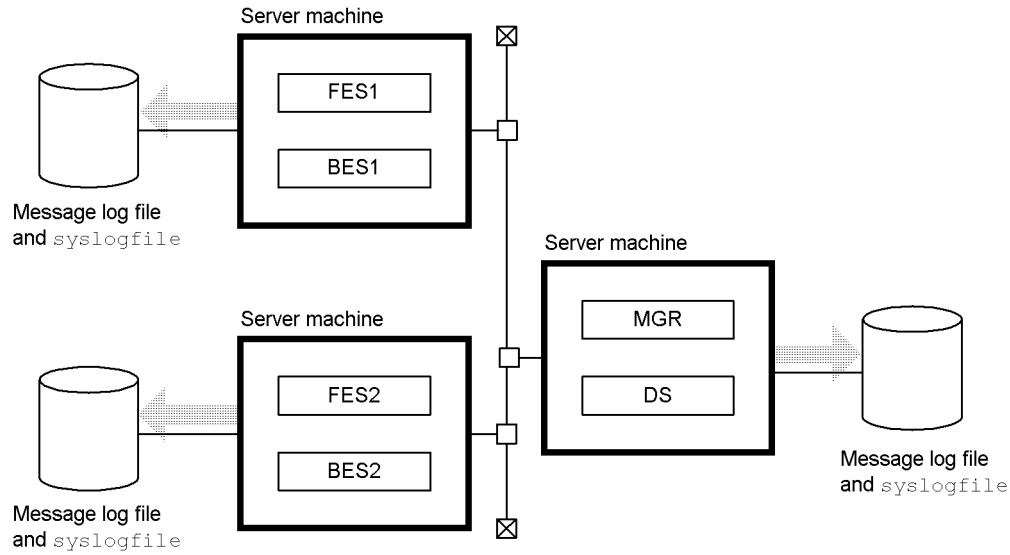


Figure 8-3: Message log output method when message log output dispersion is used (method 2)



Reference note:

When you use the `pdcat` command to view message logs, you will view the message logs of the unit in which the `pdcat` command is executed. For this reason, if message log output dispersion is used, you must execute the `pdcat` command at the unit where the desired message logs to be viewed are located.

(2) Criteria for selecting a message log output method

Table 8-1 shows the message log output methods; Table 8-2 shows the advantages and disadvantages of each message log output method.

Table 8-1: Message log output methods

Method selected*	Condition	Message log output destination
Method 1 (`manager`)	During normal operation	Message logs are output to the message log files and to `syslogfile` of the server machine where the system manager is defined.
	When an error or communication error occurs at the manager unit	Message logs are output to `syslogfile` of each server machine. Only some message log information is output. Also, the correct message logs may not be output.
Method 2 (`local`)	During normal operation	Message logs are output to the message log files and to `syslogfile` of each server machine.
	When an error or communication error occurs at the manager unit	

* The value in parentheses ( ) is the value specified in the `pd_mlg_msg_log_unit` operand.

Table 8-2: Advantages and disadvantages of the message log output methods

Method selected*	Advantage	Disadvantage
Method 1 (`manager`)	Because message logs can be centrally managed from the server machine of the system manager, the job of monitoring messages is simpler than when `local` is specified.	If the system manager unit shuts down or a communication error occurs between the system manager unit and other units, the correct message logs may not be output.
Method 2 (`local`)	Correct message logs are output even if the system manager unit shuts down or a communication error occurs between the system manager unit and other units.	Because message logs are output to each server machine, the job of monitoring messages is more complicated than when `manager` is specified.

* The value in parentheses ( ) is the value specified in the pd_mlg_msg_log_unit operand.

(3) Environment settings

The following shows the environment settings for using message log output dispersion:

Procedure

1. Because message log files are created in each server machine, you must re-evaluate your estimate of message log file size. Specify the revised result in the pd_mlg_file_size operand.
2. Specify local in the pd_mlg_msg_log_unit operand.
3. Synchronize the time-zone settings in all server machines.

Reference note:

If the time-zone settings are not synchronized among the server machines, the times in the additional information in the message logs that are output to the various server machines will be different. Therefore, when you view the message logs, you will have to take into account the time-zone differences among the server machines.

(4) Notes on selecting message log output dispersion

- When you use JP1/BASE (or JP1/SES) to monitor messages, you must monitor syslogfile of all server machines.
- When you specify PD_MLOG for the records to be collected by JP1/Performance Management - Agent Option for HiRDB, only the message logs that are output by the server machine where the system manager is defined are collected.

8.1.4 Suppressing message output to syslogfile

HiRDB outputs information such as error information, transaction information, and system file information to syslogfile as messages that indicate the system's operating status. Although these types of information are important for determining the operating status of HiRDB, a large volume of message information may be output depending on the operating environment, making it difficult and time-consuming to find relevant messages. You can improve message retrieval efficiency by suppressing unnecessary messages from being output to syslogfile.

(1) Environment settings

You use the pdmlgput operand to specify the messages that are to be suppressed. This operand can also be used to specify the following:

- Suppress output of all messages

- Specify the messages that are to be output

(2) Processing when message output is suppressed

When message output suppression is specified, the specified messages are not output to `syslogfile`. Suppressed messages are output to the message log files. Table 8-3 shows the differences in message output processing depending on whether or not message output suppression is in effect.

Table 8-3: Differences in message output processing depending on whether or not message output suppression is in effect

Output suppression specification	Server machine in which the system manager is defined		Server machine in which the system manager is not defined	
	syslogfile	Message log file	syslogfile	Message log file
Output suppression is not specified	Y	Y	Y[1]	N
Output suppression is specified	N	Y[2]	N	Y[1,3]

Legend:

Y: Messages are output.

N: Messages are not output.

Note

For a HiRDB/Single Server, use the columns for *Server machine in which the system manager is defined*.

[1] For a HiRDB/Parallel Server, message logs are output to `syslogfile` and the message log files of the server machine where the system manager is defined. However, in the following cases, messages may not be output at all or may be output to `syslogfile` of the server machine that issued the messages:

- The system monitor unit has not started yet or has encountered an error.
- The message log server has terminated abnormally or has not been restarted yet.
- The server machine is a standby system.

[2] Messages that are normally output to both `syslogfile` and the message log files are now output only to the message log files.

[3] For a HiRDB/Parallel Server, `pdlog1` and `pdlog2` (message log files) are created under `$PDDIR/spool` of the server machine in which the system manager is not

defined, and messages that used to be output to `syslogfile` are now output to these message log files.

(3) Exceptions to message output suppression

In the cases below, message output is not suppressed; instead, messages are output to `syslogfile`.

(a) When an error occurs in a message log file

When an error occurs in a message log file, an error message is output to `syslogfile`. Messages that cannot be output to the message log file because of the error are output to `syslogfile`.

(b) When the HiRDB system definition contains an error

Messages that report a HiRDB system definition error are not suppressed. Such error messages are output to `syslogfile`.

(c) Messages that are output when commands are executed

Messages that are output when the following commands are executed are not suppressed:

- `pdsetup`
- `pdplgset`
- `pdlodsv`
- `pddbadset`

(4) Usage examples

(a) Example 1

No messages are output to `syslogfile` and all messages are output to the message log files. The following an example of specifying the `pdmlgput` operand for this case:

```
pdmlgput -s N -c ALL
```

(b) Example 2

The `KFPS01820` and `KFPS00105` messages are not output to `syslogfile`, but instead are output to the message log files. The following is an example of specifying the `pdmlgput` operand for this case:

```
pdmlgput -s N -m KFPS01820,KFPS00105
```

(c) Example 3

Only the KFPH00211 and KFPH00212 messages are output to syslogfile and all other messages are output to the message log files. The following is an example of specifying the pdmlgput operand for this case:

```
pdmlgput -s N -c ALL                          …1
pdmlgput -s Y -m KFPH00211, KFPH00212         …2
```

Explanation

1. No messages are to be output to syslogfile and all messages are to be output to the message log files.
2. The KFPH00211 and KFPH00212 messages are to be output to both syslogfile and the message log files.

(5) Notes

(a) Processing when a message log file becomes full

When a message log file's capacity (value of the pd_mlg_file_size operand) is reached, the output destination for messages is swapped to the other message log file. When this swapping occurs, the KFPS01910-I message is output to syslogfile. If output of the KFPS01910-I message is suppressed, it is output to the message log file that was being used before swapping. If you are monitoring the message log file, you must swap the monitoring-target message log file whenever the KFPS01910-I message is output.

(b) Message log file size

When message output suppression is in effect, you must allocate a sufficient size for the message log file. You use the pd_mlg_file_size operand to specify a message log file size (the default is 1024 KB).

When message output suppression is in effect, a large volume of messages may be output to the message log file depending on the operating environment. Therefore, if the message log file size is insufficient, message log files are overwritten frequently, and there is a possibility that important messages, such as system operating status and troubleshooting information, that are output only to message log files may be erased.

(6) Relationship to other facilities (applicable only to a HiRDB/Parallel Server)

Table 8-4 shows the processing by HiRDB when message output suppression is used in combination with message log output dispersion.

Table 8-4: Processing by HiRDB when message output suppression is used in combination with message log output dispersion

Condition		Message output destination	
		When message output suppression is not used	When message output suppression is used
Message log output dispersion	Not used	• Message log file of the server machine where the system manager is defined • `syslogfile` of the server machine where the system manager is defined	• Message log file of the server machine where the system manager is defined
	Used	• Message log file of each server machine • `syslogfile` of each server machine	• Message log file of each server machine

8.2 When a UAP or utility execution takes too long

Executor: HiRDB administrator

When the processing time of a UAP or utility takes longer than expected, an operation command should be used to check the execution status of the UAP or utility. Figure 8-4 shows the procedure for checking UAP or utility execution status.

It must be noted that the execution status of the following utilities cannot be checked using the procedure shown in Figure 8-4:

- Database initialization utility
- Database copy utility
- Database recovery utility
- Statistics analysis utility

For these utilities, the `pdls -d prc` command can be used to check only whether or not processing by the utility is underway.

Figure 8-4: Procedure for checking the UAP or utility execution status

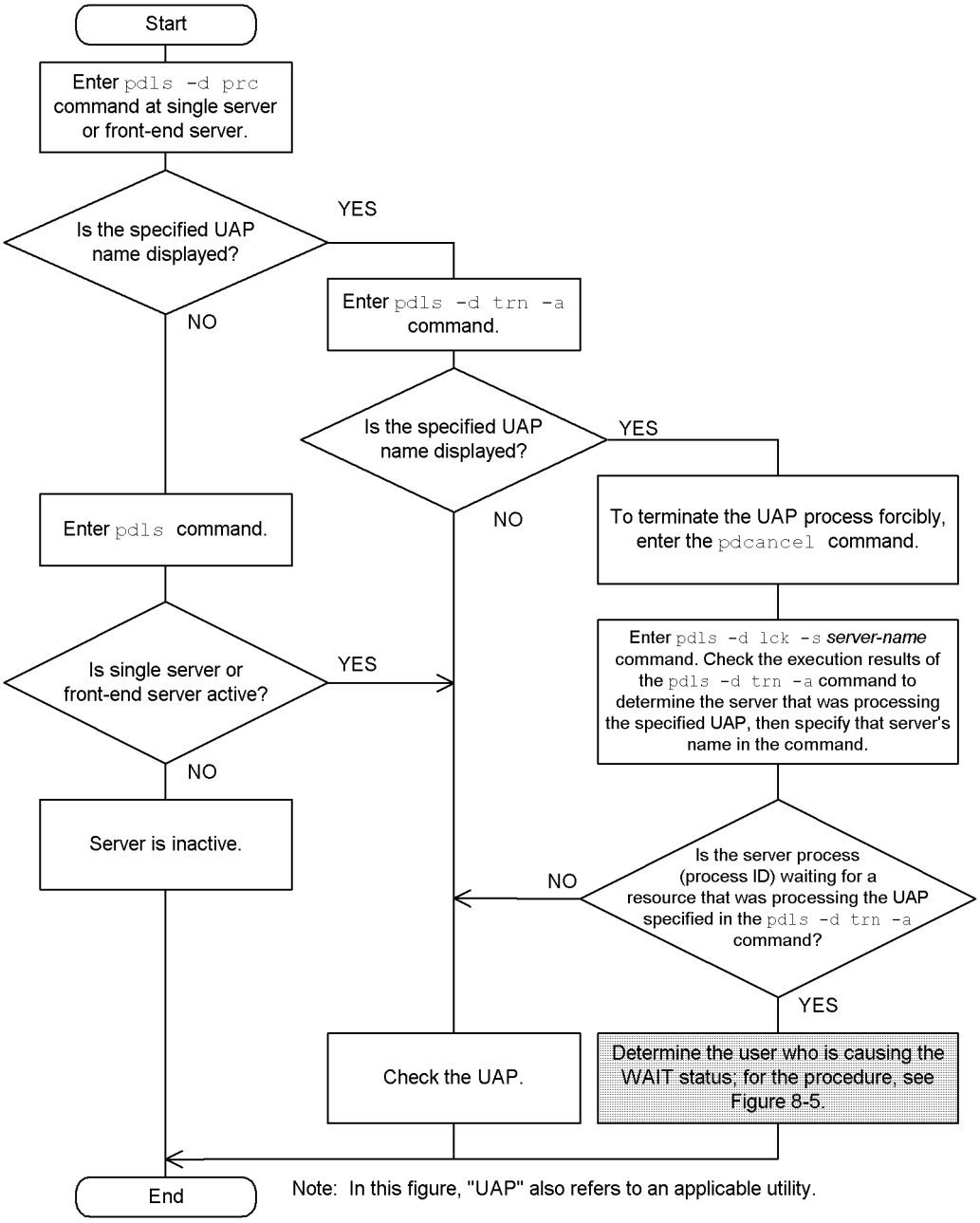
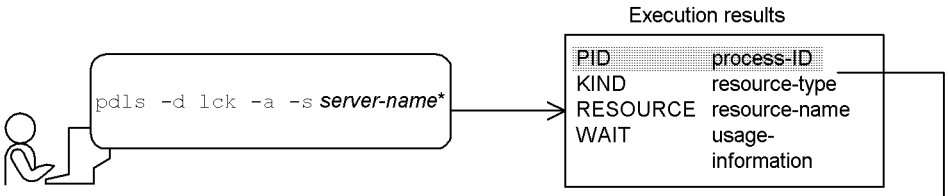
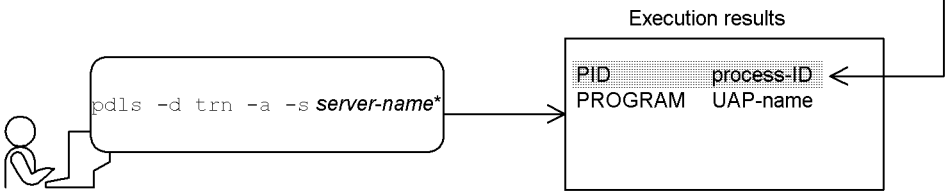


Figure 8-5: Procedure for determining the user who is causing a WAIT status

1. To obtain the resource usage status, enter the `pdls -d lck -a -s` *server-name* command. This command returns the type and name of a resource the process is waiting for and the ID of the process that is currently using that resource.



2. To determine which UAP is using the resource, enter the `pdls -d trn -a -s` *server-name* command. The UAP using the resource, as indicated by the process ID, is the cause of the WAIT status.



*Specify the same server name that was specified in Figure 8-4.

8.3 When HiRDB startup or termination processing takes too long

Executor: HiRDB administrator

If HiRDB startup or termination processing takes too long, the `pdls -d svr` command can be used to check the execution status of the HiRDB startup or termination processing. For details about interpreting the execution results of this command, see the manual *HiRDB Version 8 Command Reference*.

8.4 Obtaining RDAREA status

Executor: HiRDB administrator

Before a UAP or utility is executed, the HiRDB administrator should determine whether or not the RDAREAs to be processed are available for UAP or utility execution. The `pddbls` command is used to check RDAREA status.

Notes

- If many RDAREAs have been defined, specifying the `ALL` option in the `pddbls` command will result in a long command processing time. In this case, either the RDAREA names or the server name should be specified.
- The `-b` option is specified to display only RDAREAs in shutdown status.

8.5 Obtaining shared memory utilization status

Executor: HiRDB administrator

(1) Checking the shared memory utilization status

The utilization status of shared memory can be obtained by the `pdls -d mem` command.

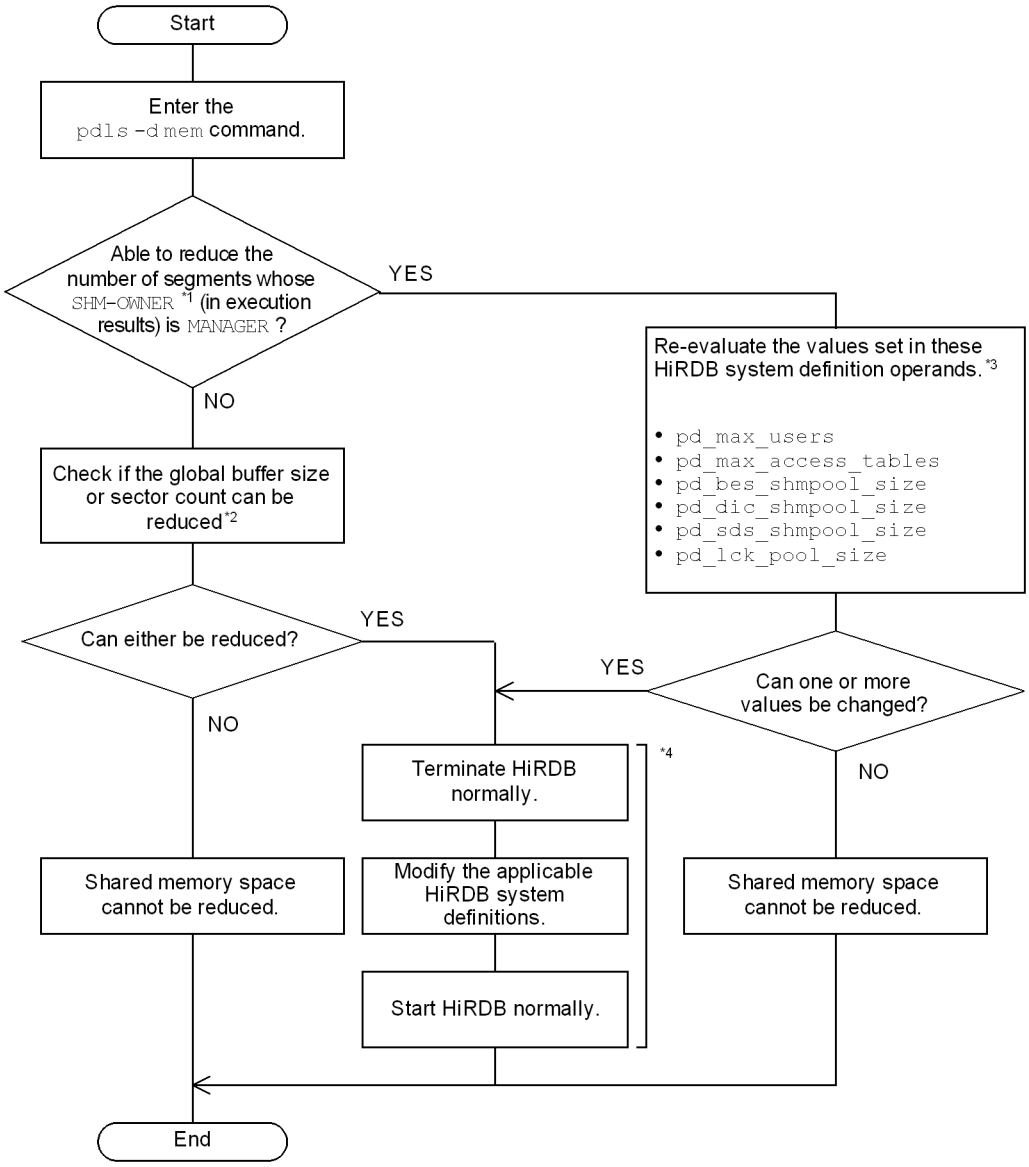
(2) Determining whether or not shared memory space can be saved

If a process will not execute because of a shortage of shared memory space, the utilization status of the shared memory should be checked. If the problem is with the OS, check if it is possible to use an OS function to resolve the problem.

For example, if the problem is with an OS parameter, correct the parameter. For details on the values to be specified in the OS parameters, see the manual *HiRDB Version 8 Installation and Design Guide*.

If the problem cannot be solved with an OS function, determine if it is possible to save shared memory space used by HiRDB; the procedure for making this determination is shown in Figure 8-6.

Figure 8-6: Procedure for determining whether or not shared memory space can be saved



[1] This is in the execution results from the `pdls -d mem` command.

[2] For details of global buffer design, see the manual *HiRDB Version 8 Installation and Design Guide*.

[3] For specification guidelines for these operands, see the manual *HiRDB Version 8 System Definition*.

[4] The system reconfiguration command (`pdchgconf` command) can be used to change HiRDB system definitions while HiRDB is running. Note that HiRDB Advanced High Availability is required in order to use this command. For details about changing HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

8.6 In the event of deadlock

Executor: HiRDB administrator

This section describes the information that is output in the event of a deadlock or timeout. The following topics are covered:

- Basics
- Deadlock information that is output
- Timeout information that is output
- Resource types and resource information
- Interpreting resource information

8.6.1 Basics

When a deadlock or timeout occurs, HiRDB outputs the information shown in Table 8-5.

Table 8-5: Information output when a deadlock or timeout occurs

Output information	Description
Deadlock information	• This is information that is output about a deadlock that has occurred between transactions within a server. • For a HiRDB/Parallel Server, a deadlock between servers is reported as timeout information. • For more information on deadlock, see the manual *HiRDB Version 8 UAP Development Guide*.
Timeout information	This is information that indicates that the lock-release wait time has elapsed.

(1) Output destination of deadlock and timeout information

Deadlock information and timeout information are output to the following files, which are called the deadlock/timeout information files:

- `$PDDIR/spool/pdlckinf/`*file-name*

The *file-name* is determined by HiRDB as follows on the basis of the date and time the deadlock or timeout occurred:

Example:

- Deadlock or timeout occurred at 09:29:56 on October 3:
  file-name: `Oct3092956`
  mmmd hhmmss
- Deadlock or timeout occurred at 18:06:00 on October 10:
  file-name: `Oct10180600`
  mmmdd hhmmss

(2) Actions to be taken by the HiRDB administrator

To output deadlock/timeout information, the following specifications must be made in the system common definition:

1. Specify in the `pd_lck_deadlock_info` operand that deadlock information and timeout information are to be output.
2. Specify a lock-release wait time value in the `pd_lck_wait_timeout` operand.

(3) Referencing deadlock and timeout information

When the following messages are output, the deadlock or timeout information should be referenced:

- `KFPA11911-E` (Message indicating that deadlock has occurred)
- `KFPS00441-I` (Message indicating that deadlock information has been output)
- `KFPS00451-I` (Message indicating that timeout information has been output)

Deadlock or timeout information can be referenced with an OS command (`cat` command, `vi` command, etc.). The `KFPS00441-I` and `KFPS00451-I` messages display the file name to be specified in these commands. For details on the `cat` and `vi` commands, see the OS manual.

For details on the deadlock and timeout information that is output, see *8.6.2 Deadlock information that is output* and *8.6.3 Timeout information that is output*.

(4) Using deadlock and timeout information

It may be possible to reduce the frequency of deadlock and timeout occurrences by changing the UAP access sequence or widening the UAP lock range. It is important to use the deadlock and timeout information that is output to reevaluate the resources that cause the deadlock or timeout. For details on the actions to be taken with respect to the resources that result in deadlock or timeout, see the manual *HiRDB Version 8 UAP Development Guide*.

(5) Deleting unneeded deadlock/timeout information files

HiRDB does not delete deadlock/timeout information files. Such files must be deleted

by the HiRDB administrator when the files are no longer needed. The following deletion procedures are provided:

(a) With a HiRDB command

The `pdcspool` command can be used to delete unneeded deadlock/timeout information files. However, the `pdcspool` command deletes all troubleshooting information files under `$PDDIR/spool`. To delete only deadlock/timeout information files, use the method described below in *(c) With an OS command*.

(b) With a HiRDB function

If `all` is specified in the `pd_spool_cleanup_interval_level` operand, HiRDB deletes deadlock/timeout information files periodically. The default is that HiRDB deletes them every 24 hours, but you can change this deletion interval with the `pd_spool_cleanup_interval` operand.

If `all` is also specified in the `pd_spool_cleanup_level` operand, HiRDB also deletes deadlock/timeout information files when it starts.

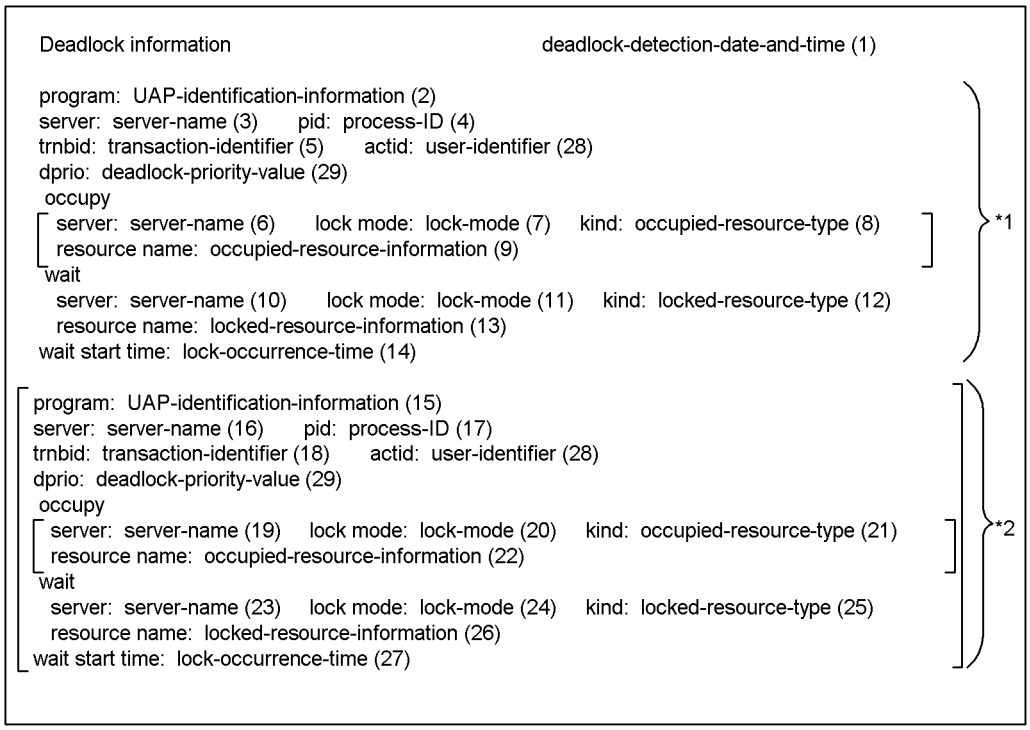
(c) With an OS command

Use an OS function (OS's `rm` command, etc.) to delete a deadlock/timeout information file. For details on the `rm` commands, see the OS manual.

8.6.2 Deadlock information that is output

Figure 8-7 shows the deadlock information that is output.

Figure 8-7: Deadlock information that is output



Note: Items in square brackets may be output more than once.
*1: Information on the server that sent the transaction that resulted in the deadlock error.
*2: Information on the server that sent the transaction involved in deadlock.

The following explains the deadlock information that is output:

1. deadlock-detection-date-and-time

   Displays the date and time (*mmm dd hh:mm:ss yyyy*) when HiRDB detected the deadlock.

   Information on the server that sent the transaction that resulted in deadlock

   Items 2 to 14 provide information about the first resource that was occupied by the transaction that resulted in the deadlock.

2. UAP-identification-information

   Displays the identification name of the UAP connected to the server that sent the transaction that resulted in the deadlock.

   The information displayed here corresponds to the PROGRAM information

displayed by the `pdls -d prc` or `pdls -d trn` command. Some utilities cannot display this information; in this case, `********` is displayed. `Rerun` is displayed for a transaction that is being restored during restart processing.

3. server-name

   Displays the name of the server that sent the transaction that resulted in the deadlock.

4. process-ID

   Displays the ID of the server process that sent the transaction that resulted in the deadlock.

5. transaction-identifier

   Displays the identifier of the transaction that resulted in the deadlock.

   If the identifier begins with `_cmd`, the lock was secured by the `pdhold` command. If the identifier begins with `_utl`, the lock was secured by the `pdcopy` command.

   Information about all resources occupied by the transaction that resulted in deadlock

   Items 6 to 9 provide information about all resources occupied by the transaction that caused the deadlock. If this transaction did not occupy any resources, these items are left blank.

6. server-name

   Displays the name of the server used by the transaction that resulted in the deadlock to issue the resource occupancy request.

7. lock-mode

   Displays the lock mode applied to the resource that was occupied by the transaction that resulted in the deadlock. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

8. occupied-resource-type

   Displays the type of resource that was occupied by the transaction that resulted in the deadlock. For details on the resource types, see *8.6.4 Resource types and resource information*.

9. occupied-resource-information

   Displays information about the resource that was occupied by the transaction that resulted in the deadlock. For details on the resource information, see *8.6.4 Resource types and resource information*.

   Information about the resource that caused the transaction that resulted in the deadlock to be placed in lock-release wait status

Items 10 to 14 provide information about the resource that caused the deadlock.

10. server-name

    Displays the name of the server used by the transaction that resulted in the deadlock to issue the resource occupancy request.

11. lock-mode

    Displays the lock mode the transaction that resulted in the deadlock attempted to apply to the resource in lock-release wait status. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

12. locked-resource-type

    Displays the type of resource that caused the transaction that resulted in the deadlock to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

13. locked-resource-information

    Displays information about the resource that caused the transaction that resulted in the deadlock to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

14. lock-occurrence-time

    Displays the time (*hh:mm:ss*) the transaction that resulted in the deadlock was placed in lock-release wait status.

    Information about the server that sent the transaction involved in deadlock

    Items 15 to 29 provide information about the first resource that was occupied by the transaction involved in the deadlock. This information may be output more than once.

15. UAP-identification-information

    Displays the identification name of the UAP connected to the server that sent the transaction involved in the deadlock.

    The information displayed here corresponds to the PROGRAM information displayed by the `pdls -d prc` or `pdls -d trn` command. Some utilities cannot display this information; in this case, `********` is displayed. `Rerun` is displayed for a transaction that is being restored during restart processing.

16. server-name

    Displays the name of the server that sent the transaction involved in the deadlock.

17. process-ID

    Displays the ID of the server process that sent the transaction involved in the deadlock.

18. transaction-identifier

    Displays the identifier of the transaction involved in the deadlock.

    Information about all resources occupied by the transaction involved in deadlock

    Items 19 to 22 provide information about all resources that caused the deadlock. If this transaction did not occupy any resources, these items are left blank.

19. server-name

    Displays the name of the server used by the transaction involved in the deadlock to issue the resource occupancy request.

20. lock-mode

    Displays the lock mode applied to the resource that was occupied by the transaction involved in the deadlock. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

21. occupied-resource-type

    Displays the type of resource that was occupied by the transaction involved in the deadlock. For details on the resource types, see *8.6.4 Resource types and resource information*.

22. occupied-resource-information

    Displays information about the resource that was occupied by the transaction involved in the deadlock. For details on the resource information, see *8.6.4 Resource types and resource information*.

    Information about the resource that caused the transaction involved in the deadlock to be placed in lock-release wait status

    Items 23 to 29 provide information about the resource that caused the deadlock.

23. server-name

    Displays the name of the server used by the transaction involved in the deadlock to issue the resource occupancy request.

24. lock-mode

    Displays the lock mode the transaction involved in the deadlock attempted to apply to the resource in lock-release wait status. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

25. locked-resource-type

    Displays the type of resource that caused the transaction involved in the deadlock to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

26. locked-resource-information

    Displays information about the resource that caused the transaction involved in the deadlock to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

27. lock-occurrence-time

    Displays the time (*hh:mm:ss*) the transaction involved in the deadlock was placed in lock-release wait status.

28. user-identifier

    Displays the serial number assigned dynamically by HiRDB to identify the individual user.

29. deadlock-priority-value

    Displays the deadlock priority value for the transaction that resulted in the deadlock.

Figure 8-8 shows an output example of deadlock information.

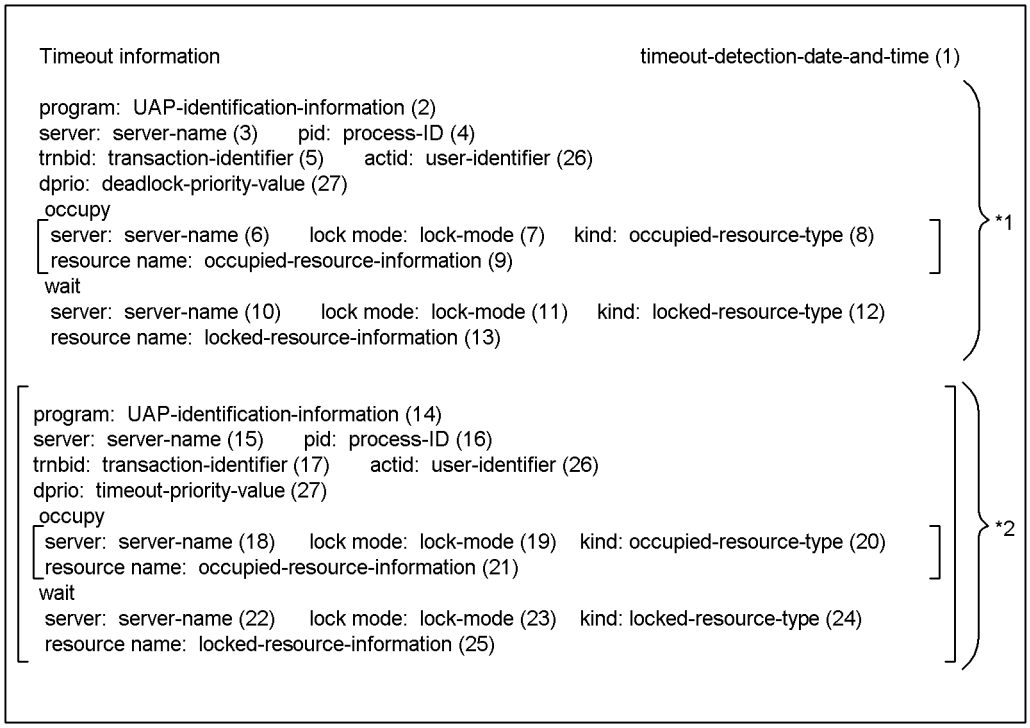
Figure 8-8: Output example of deadlock information

```
Deadlock information                                    Jun  2 06:12:43: 2000

program : SPPY415
server : SDS    pid : 5251
trnbid : q192u19200000000       actid : 1-1-4    dprio : 64
 occupy
  server : SDS  lock mode : PR   kind : 0007
    resource name : 0000060000001901000 2007d0000
 wait
  server : SDS  lock mode : EX   kind : 0007
    resource name : 0000060000001901000 2007d0000
  wait start time : 06:12:43

program : SPPE201
server : SDS    pid : 5249
trnbid : q192u19200000003       actid : 1-1-6    dprio : 64
 occupy
  server : SDS  lock mode : PR   kind : 0007
    resource name : 0000060000001901000 2007d0000
 wait
  server : SDS  lock mode : EX   kind : 0007
    resource name : 0000060000001901000 2007d0000
  wait start time : 06:12:43
```

8.6.3 Timeout information that is output

Figure 8-9 shows the timeout information that is output.

Figure 8-9: Timeout information that is output



Note: Items in square brackets may be output more than once.
*1: Information on the server that sent the transaction that resulted in the timeout.
*2: Information on the server that sent the transaction involved in the timeout.

The following explains the timeout information that is output:

1. timeout-detection-date-and-time

   Displays the date and time (*mmm dd hh:mm:ss yyyy*) when HiRDB detected the timeout.

   Information on the server that sent the transaction that resulted in the timeout

   Items 2 to 13 provide information about the first resource that was occupied by the transaction that resulted in the timeout.

2. UAP-identification-information

Displays the identification name of the UAP connected to the server that sent the transaction that resulted in the timeout. If the transaction was sent by a utility, the name of the utility is displayed.

The information displayed here corresponds to the PROGRAM information displayed by the `pdls -d prc` or `pdls -d trn` command. Some utilities cannot display this information; in this case, `********` is displayed. `Rerun` is displayed for a transaction that is being restored during restart processing.

3. server-name

   Displays the name of the server that sent the transaction that resulted in the timeout.

4. process-ID

   Displays the ID of the server process that sent the transaction that resulted in the timeout.

5. transaction-identifier

   Displays the identifier of the transaction that resulted in the timeout.

   If the identifier begins with `_cmd`, the lock was secured by the `pdhold` command. If the identifier begins with `_utl`, the lock was secured by the `pdcopy` command.

   Information about all resources occupied by the transaction that resulted in the timeout

   Items 6 to 9 provide information about all resources occupied by the transaction that caused the timeout. If this transaction did not occupy any resources, these items are left blank.

6. server-name

   Displays the name of the server used by the transaction that resulted in the timeout to issue the resource occupancy request.

7. lock-mode

   Displays the type of lock mode applied to the resource that was occupied by the transaction that resulted in the timeout. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

8. occupied-resource-type

   Displays the type of resource that was occupied by the transaction that resulted in the timeout. For details on the resource types, see *8.6.4 Resource types and resource information*.

9. occupied-resource-information

   Displays information about the resource that was occupied by the transaction that

resulted in the timeout. For details on the resource information, see *8.6.4 Resource types and resource information*.

Information about the resource that caused the transaction that resulted in the timeout to be placed in lock-release wait status

Items 10 to 13 provides information about the resource that caused the transaction that resulted in the timeout to be placed in lock-release wait status.

10. server-name

    Displays the name of the server used by the transaction resulting in the timeout to issue the resource occupancy request.

11. lock-mode

    Displays the lock mode the transaction that resulted in the timeout attempted to apply to the resource in lock-release wait status. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

12. locked-resource-type

    Displays the type of resource that caused the transaction that resulted in the timeout to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

13. locked-resource-information

    Displays information about the resource that caused the transaction that resulted in the timeout to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

Information about the server that sent the transaction that caused the timeout

Items 14 to 27 provide information about the first resource that was occupied by the transaction that caused the timeout. This information may be output more than once.

14. UAP-identification-information

    Displays the identification name of the UAP connected to the server that sent the transaction that caused the timeout. If the transaction was sent by a utility, the name of the utility is displayed.

    The information displayed here corresponds to the PROGRAM information displayed by the `pdls -d prc` or `pdls -d trn` command. Some utilities cannot display this information; in this case, `********` is displayed. `Rerun` is displayed for a transaction that is being restored during restart processing.

15. server-name

    Displays the name of the server that sent the transaction that caused the timeout.

16. process-ID

    Displays the ID of the server process that sent the transaction that caused the timeout.

17. transaction-identifier

    Displays the identifier of the transaction that caused the timeout.

    Information about all resources occupied by the transaction that caused the timeout

    Items 18 to 21 provide information about all resources that caused the timeout. If this transaction did not occupy any resources, these items are left blank.

18. server-name

    Displays the name of the server used by the transaction that caused the timeout to issue the resource occupancy request.

19. lock-mode

    Displays the lock mode applied to the resource that was occupied by the transaction that caused the timeout. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

20. occupied-resource-type

    Displays the type of resource that was occupied by the transaction that caused the timeout. For details on the resource types, see *8.6.4 Resource types and resource information*.

21. occupied-resource-information

    Displays information about the resource that was occupied by the transaction that caused the timeout. For details on the resource information, see *8.6.4 Resource types and resource information*.

    Information about the resource that caused the transaction that caused the timeout to be placed in lock-release wait status

    Items 22 to 27 provide information about the resource that caused the timeout. Item `wait` and the subsequent items may not be displayed.

22. server-name

    Displays the name of the server used by the transaction that caused the timeout to issue the resource occupancy request.

23. lock-mode

    Displays the lock mode the transaction that caused the timeout attempted to apply to the resource in lock-release wait status. For details on the lock modes, see the manual *HiRDB Version 8 UAP Development Guide*.

24. locked-resource-type

    Displays the type of resource that caused the transaction that caused the timeout to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

25. locked-resource-information

    Displays information about the resource that caused the transaction that caused the timeout to be placed in lock-release wait status. For details on the resource types, see *8.6.4 Resource types and resource information*.

26. user-identifier

    Displays the serial number assigned dynamically by HiRDB to identify the individual user.

27. deadlock-priority-value

    Displays the deadlock priority value for the transaction that resulted in the deadlock.

Figure 8-10 shows an output example of timeout information.

Figure 8-10: Output example of timeout information

```
Timeout information                                          Jun  2 06:12:43: 2000

program : SPPY415
server : SDS    pid : 4510
trnbid : q192u19200000001       actid : 1-1-5    dprio : 64
 occupy
  server : SDS  lock mode : PR   kind : 3001
  resource name : 000006000000190100002007d0000
  server : SDS  lock mode : SR   kind : 0001
  resource name : 00000003000000000000000000000
  server : SDS  lock mode : SR   kind : 0002
  resource name : 00020008000000000000000000000
  server : SDS  lock mode : SU   kind : 0001
  resource name : 00000005000000000000000000000
  server : SDS  lock mode : SU   kind : 0002
  resource name : 0002007c00000000000000000000
  server : SDS  lock mode : EX   kind : 0007
  resource name : 0000050000001b020002007c0000
 wait
  server : SDS  lock mode : EX   kind : 0007
  resource name : 0000050000001b020002007c0000

program : SPPE201
server : SDS    pid : 4481
trnbid : q192u19200000003       actid : 1-1-4    dprio : 64
 occupy
  server : SDS  lock mode : EX   kind : 0007
  resource name : 0000050000001b020002007c0000
```

8.6.4 Resource types and resource information

Table 8-6 shows the resource types and resource information included in the deadlock or timeout information.

Table 8-6: Resource types and resource information

Resource type	Type name	Resource name 1	Resource name 2	Resource information	Contents
0001	RDAR	RDAREA	—	Digits 1-8: RDAREA number Digits 9-28: Fixed to 00	RDAREA
0002	TABL	Table name	—	Digits 1-8: Table number Digits 9-16: Generation number Digits 17-28: Fixed to 00	Table

Resource type	Type name	Resource name 1	Resource name 2	Resource information	Contents
0003	INDX	Index name	RDAREA name	Digits 1-8: Index number Digits 9-16: RDAREA number Digits 17-28: Fixed to 00	Index
0004	PAGE	Table name or index name	RDAREA name	■ For HP-UX, Solaris, and AIX 5L versions Digits 1-6: RDAREA number Digits 7-8: File number Digits 9-16: Page number Digits 17-24: Table number or index number Digits 25-28: Fixed to 00 ■ For Linux version Digits 1-2: File number Digits 3-8: RDAREA number Digits 9-16: Page number Digits 17-24: Table number or index number Digits 25-28: Fixed to 00	Page
0007	ROW	Table name	RDAREA name	■ For HP-UX, Solaris, and AIX 5L versions Digits 1-6: RDAREA number Digits 7-8: File number Digits 9-14: Page number Digits 15-16: Slot number Digits 17-24: Table number Digits 25-28: Fixed to 00 ■ For Linux version Digits 1-2: File number Digits 3-8: RDAREA number Digits 9-10: Slot number Digits 11-16: Page number Digits 17-24: Table number Digits 25-28: Fixed to 00	Row

Resource type	Type name	Resource name 1	Resource name 2	Resource information	Contents
0008	NWROW	Table name	RDAREA name	■ For HP-UX, Solaris, and AIX 5L versions Digits 1-6: RDAREA number Digits 7-8: File number Digits 9-14: Page number Digits 15-16: Slot number Digits 17-24: Table number Digits 25-28: Fixed to 00 ■ For Linux version Digits 1-2: File number Digits 3-8: RDAREA number Digits 9-10: Slot number Digits 11-16: Page number Digits 17-24: Table number Digits 25-28: Fixed to 00	Row (for controlling `WITHOUT LOCK NOWAIT` retrieval)
000B	TABN	Table name	—	Digits 1-8: Table number Digits 9-16: Generation number Digits 17-28: Fixed to 00	Table (during `NO WAIT` retrieval)
000C	NKEY	Index name	—	Digits 1-8: Index number Digits 9-28: Fixed to 00	Key value (`NULL`)
000D	DKEY	Index name	—	Digits 1-8: Index number Digits 9-28: Key value or encoded key value	Key value (non-NULL)
000E	LFID	RDAREA name	Logical file number	Digits 1-8: RDAREA number Digits 9-16: Logical file number Digits 17-28: Fixed to 00	Logical file
000F	IXIF	Index name	RDAREA name	Digits 1-8: Index number Digits 9-16: RDAREA number Digits 17-28: Fixed to 00	Index information file
0010	TRWT	Table name	Name of the RDAREA storing the index	Digits 1-8: Table number Digits 9-16: Number of the RDAREA storing the index Digits 17-28: Fixed to 00	Transaction completion pending

Resource type	Type name	Resource name 1	Resource name 2	Resource information	Contents
0011	SHWT	Table name (**** is output if the system is waiting for RDAREA update completion)	RDAREA name	■ For HP-UX, Solaris, and AIX 5L versions Digits 1-6: RDAREA number Digits 7-8: File number Digits 9-14: Page number Digits 15-16: Slot number Digits 17-24: Table number Digits 25-28: Fixed to 00 ■ For Linux version Digits 1-2: File number Digits 3-8: RDAREA number Digits 9-10: Slot number Digits 11-16: Page number Digits 17-24: Table number Digits 25-28: Fixed to 00 If the system is waiting for RDAREA update completion, all digits except for the RDAREA number are 0.	Shared RDAREA transaction completion pending
0102	RRAMB	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-28: Fixed to 00	RDAREA management information
0111	MFCB	—	—	Digits 1-8: Page number Digits 9-12: File number Digits 13-28: Fixed to 00	Master directory segment information
0112	MTCB	Table name	—	Digits 1-8: Table number Digits 9-28: Fixed to 00	Master directory table information
0113	MICB	Index name	—	Digits 1-8: Index number Digits 9-28: Fixed to 00	Master directory index information
0121	RATM	Table name	RDAREA name	Digits 1-8: RDAREA number Digits 9-16: Table number or table management number Digits 17-28: Fixed to 00	Dictionary table information or user directory table information
0122	RAIM	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-16: Index number or index management number Digits 17-28: Fixed to 00	Dictionary index information or user directory index information

Resource type	Type name	Resource name 1	Resource name 2	Resource information	Contents
0132	SBMB	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-16: Segment number Digits 17-20: File number Digits 21-28: Fixed to 00	Dictionary segment information or user directory segment information
0143	RDLF	RDAREA name	Last file number	Digits 1-8: RDAREA number Digits 9-12: Last file number Digits 13-28: Fixed to 00	RDAREA increment
0152	SGMB	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-16: Page number Digits 17-20: File number Digits 21-28: Fixed to 00	Database copy utility
0300	MENT	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-16: Page number Digits 17-20: File number Digits 21-24: Entry number Digits 25-28: Fixed to 00	User LOB RDAREA management information
0301	LOBID	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-16: LOB number Digits 17-28: Fixed to 00	User LOB RDAREA management information
0601	RDAS	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-28: Fixed to 00	Database condition analysis utility
0602	HOLD	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-28: Fixed to 00	Backup-hold RDAREA
0603	INRP	Server name	—	Bytes 1-8 indicate the server name. If the server name does not occupy 8 bytes, the unused bytes are filled with NULLs.	Inner replica configuration management information
0604	RPGP	Original RDAREA name	—	Digits 1-8: RDAREA number Digits 9-28: Fixed to 00	Replica group configuration management information
0900	PLGR	—	—	Digits 1-8: Plug-in ID Digits 9-28: Plug-in's unique resource number	Plug-in resource number

Resource type	Type name	Resource name 1	Resource name 2	Resource information	Contents
2002	ROMB	—	—	Digits 1-14: SQL object number Digits 15-28: Fixed to 00	SQL object management information
2003	SPCH	—	—	Fixed to 'SPCH' Extra digits are zero-filled.	SQL object cache
3001	PTBL	"****"	"****"	Digits 1-8: Table number Digits 9-28: Fixed to 00	Pre-processing table
3001	AUDL	"****"	"****"	Fixed to AUDL	Audit trail information pool
3005	DICT	"****"	"****"	Fixed to 'DICT' Extra digits are zero-filled.	Dictionary table
3006	VIEW	"****"	"****"	Digits 1-8: View table number Digits 9-28: Fixed to 00	View table
3008	TBPL	"****"	"****"	Bytes 1-5[1] are an authorization identifier. Bytes 6-14[2] are a table identifier.	Table definition information buffer
3009	ALAS	"****"	"****"	Bytes 1-5[1] are an authorization identifier. Bytes 6-14[2] are a table alias.	Table alias definition information buffer
3010	AUTH	"****"	"****"	Bytes 1-14 bytes are an authorization identifier.	User privilege information buffer
3011	OBJI	"****"	"****"	Routine object number	Routine object
3012	DTYP	"****"	"****"	Fixed to 'DTYP' Extra digits are zero-filled.	Data type definition information
3013	RTPL	"****"	"****"	Bytes 1-12 are a routine identifier. Bytes 13-14 are a parameters count.	Routine definition information buffer
3014	TPPL	"****"	"****"	Bytes1-5[1] are an authorization identifier. Bytes 6-14[2] are a data type identifier.	User-defined type information buffer

Resource type	Type name	Resource name 1	Resource name 2	Resource information	Contents
3015	DICR	"****"	"****"	Fixed to DICR	Dictionary RDAREA modification
3016	CONS	"****"	"****"	Fixed to CONSEC	CONNECT-related security definition information
5001	DICU	—	—	Fixed to 'DICTMODUTL' Extra digits are zero-filled.	Database structure modification utility
5002	LCBF	RDAREA name	—	Digits 1-8: RDAREA number Digits 9-28: Fixed to 00	—
5003	EXP1	—	—	Fixed to 'EXPIMPMDL' Extra digits are zero-filled.	Database import/ export utility
5004	RBAL	Table name	—	Digits 1-8: Table number Digits 9-28: Fixed to 00	Rebalancing utility
5005	RRAMB	Server name	—	Digits 1-16: Server name Digits 17-28: Fixed to 00	ZRRAMB updates
5006	RCLM	RDAREA name	Table name or index name	Digits 1-8: RDAREA number Digits 9-16: Table number or index number Digits 17-28: Fixed to 00	Free page release utility

—: Not applicable

Notes

- The resource type is displayed in hexadecimal (4 digits).
- The resource information is displayed in hexadecimal (28 digits). For details on interpreting resource information, see *8.6.5 Interpreting resource information*.
- The RDAREA name corresponding to an RDAREA number can be referenced with the pddbls command.
- The generation number is displayed if the table has been duplicated with the inner replica facility.

[1] Authorization identifiers (6 bytes or longer) are output in the following format:

first-3-bytes-of-authorization-identifier + *last-2-bytes-of-authorization-identifier*

For example, if the authorization identifier is `k87m341`, `k8741` is output.

Note that the information is in ASCII codes and one byte is output as two digits.

[2] Table identifiers, table aliases, and data type identifiers (10 bytes or longer) are output in the following format:

first-5-bytes-of-table-identifier-or-table-alias +
last-4-bytes-of-table-identifier-or-table-alias

For example, if the table identifier is `TABLE002498`, `TABLE2498` is output.

Note that the information is in ASCII codes, and one byte is output as two digits.

8.6.5 Interpreting resource information

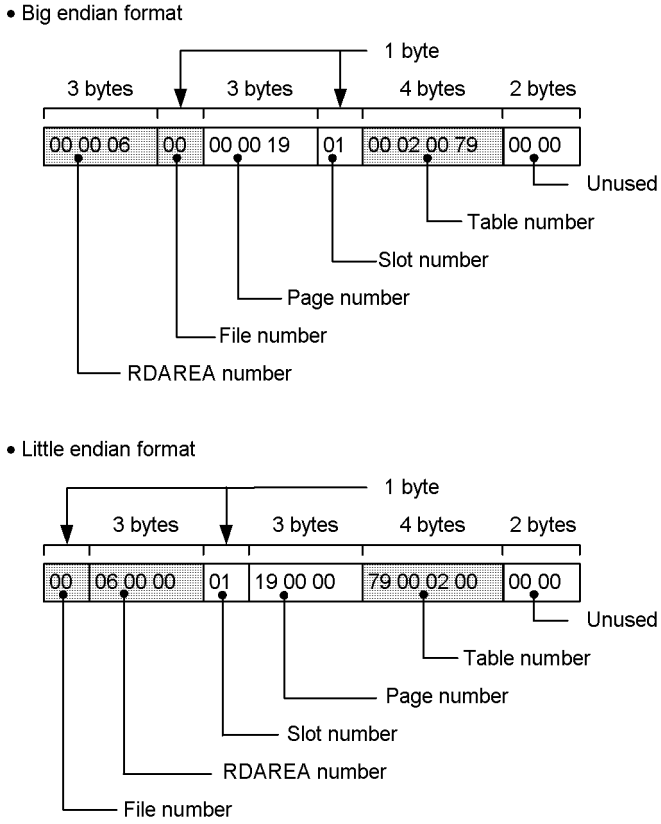
Resource information is displayed in hexadecimal (28 digits). If multiple resource information items are specified, they are displayed consecutively in the order of their specification. If the resource information is fewer than 28 digits, the extra digits are zero-filled. All characters are in ASCII codes, and one byte is output as two digits.

Note:

For the HP-UX, Solaris, and AIX 5L versions of HiRDB, resource information is output in big endian format. For the Linux version of HiRDB, resource information is output in little endian format.

Figure 8-11 shows an output example of resource information (when the resource type is 0007).

Figure 8-11: Resource information output example (when the resource type is 0007)

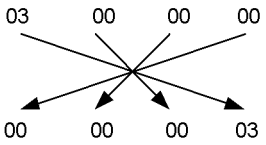


(1) Determining the RDAREA name from an RDAREA number

The following shows how to determine the RDAREA name from the RDAREA number that is output in the resource information:

Procedure

1. If the output information is in little endian format, convert it to big endian format. Suppose that `03000000` is output as the RDAREA number. Convert it to big endian format as shown below.



The RDAREA number becomes 3.

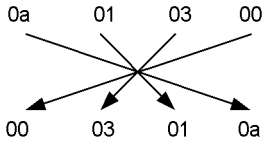
2. Because the RDAREA number that is output in the resource information is in hexadecimal, convert it to decimal.
3. Use the pddbls command to determine the RDAREA name that corresponds to the RDAREA number.

(2) Determining the index name from an index number

The following shows how to determine the index name from the index number that is output in the resource information:

Procedure

1. If the output information is in little endian format, convert it to the big endian format. Suppose that 0a010300 is output as the index number. Convert it to big endian format as shown below.



The index number becomes 3010a.

2. Because the index number that is output in the resource information is in hexadecimal, convert it to decimal.
3. Retrieve the INDEX_ID column of the SQL_INDEXES dictionary table and determine the index name that corresponds to the index number.

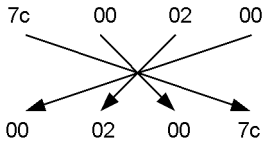
   For details about how to retrieve the SQL_INDEXES dictionary table, see the manual *HiRDB Version 8 UAP Development Guide*.

(3) Determining the table name (view table name) from a table number (view table number)

The following shows how to determine the table name (view table name) from the table number (view table number) that is output in the resource information:

Procedure

1. If the output information is in little endian format, convert it to big endian format. Suppose that 7c000200 is output as the table number. Convert it to big endian format as shown below.



The table number becomes `2007c`.

2. Because the table number (view table number) that is output in the resource information is in hexadecimal, convert it to decimal.
3. Retrieve the `TABLE_ID` column of the `SQL_TABLES` dictionary table and determine the table name (view table name) that corresponds to the table number (view table number).

   For details about how to retrieve the `SQL_TABLES` dictionary table, see the manual *HiRDB Version 8 UAP Development Guide*.

8.7 In the event of a shortage of locked resources management tables

Executor: HiRDB administrator

When there are too few locked resources management tables, HiRDB outputs the KFPS00443-I message and locked resources management table information. The locked resources management table information is output to a directory at the unit where the shortage occurred ($PDDIR/spool/pdlckinf). The HiRDB administrator must reference this information and take appropriate action to resolve the problem.

(1) Names of locked resources management table data files

Locked resources management table information is output to a file each time a shortage of locked resources management tables occurs. The name of this output file is output-date-and-time.mem, where mem is a file descriptor. In the event of a shortage of locked resources management tables occurring at 09:16:02 on October 3, the name of the output file would be Oct3091602.mem. This file name is displayed in the KFPS00447-I message.

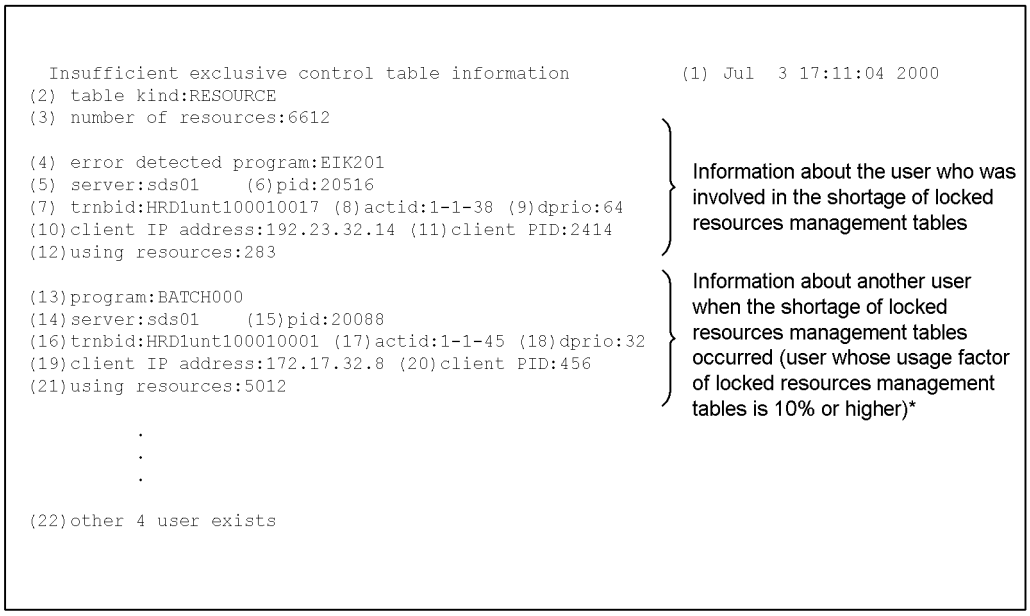
Deleting unneeded locked resources management table data files

HiRDB does not delete locked resources management table data files. Unneeded files must be deleted by the HiRDB administrator (using the OS's rm command, etc.). For details on the rm command, see the applicable OS manual.

(2) Locked resources management table information

Figure 8-12 shows the locked resources management table information that is output.

Figure 8-12: Locked resources management table information that is output



* This information is displayed as many times as there are affected users.

Explanation

1. Date and time the shortage of locked resources management tables was detected (*mmm dd hh:mm:ss yyyy*).
2. Type of affected locked resources management tables:

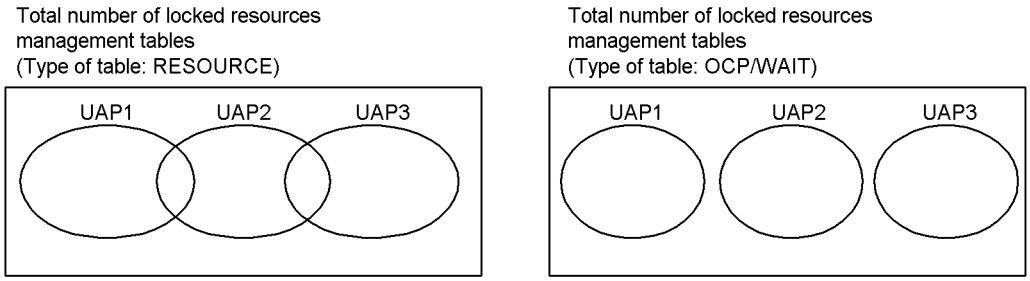
   - RESOURCE

   Tables used for managing resource names. These tables are shared among multiple users. Therefore, the sum of the tables used by all users may exceed the actual number of tables.

   - OCP/WAIT

   Tables used for managing shared or wait status. These tables are not shared among multiple users.

   The following shows the difference between `RESOURCE` and `OCP/WAIT`:



3. Total number of available locked resources management tables.

   Information about the user who was involved in the shortage of locked resources management tables:

4. UAP identification information[1]
5. Server name
6. Process ID
7. Transaction identifier
8. User identification number
9. Deadlock priority value
10. Client's IP address
11. Client's process ID
12. Number of tables currently used

    Information about another user when the shortage of locked resources management tables occurred (user whose usage factor of locked resources management tables is 10% or higher):

13. UAP identification information[1]
14. Server name
15. Process ID
16. Transaction identifier
17. User identification number
18. Deadlock priority value
19. Client's IP address[2]
20. Client's process ID[2]

21. Number of tables currently used

    Information about other users when the shortage of locked resources management tables occurred (users whose usage factor of locked resources management tables is less than 10%):

22. Number of users whose usage factor of locked resources management tables is less than 10%

[1] Displays in 30 bytes the name of the client UAP that was connected.

This is the `PROGRAM` information that is output by the `pdls -d prc` or the `pdls -d trn` command. This information is not displayed for some utilities, in which case `********` is displayed. If the transaction is being recovered by restart processing, `Rerun` is displayed.

[2] This information is not displayed when the HiRDB version of the client library linked to the connected client UAP is 4.0 04-00 or earlier. This information may not be displayed for a back-end server or dictionary server, in which case `********` is displayed as the client IP address and `0` is displayed as the process ID.

(3) Key items to be checked

Following are the key items that should be checked in the locked resources management table information:

- 3. Total number of available locked resources management tables
- 21. Number of tables currently used

The HiRDB administrator can determine from this information the UAPs using the locked resources management tables and their table usage factors. The number of locked resources management tables used is equal to the number of lock requests issued by UAPs. Therefore, a UAP for which the `21`. *Number of tables currently used* value is too high may have issued too many lock requests. The number of lock requests issued by a UAP depends on the SQL; for the number of lock requests for each SQL (the estimated number of locked resources), see the manual *HiRDB Version 8 System Definition*.

(4) Action to be taken

(a) When there is a UAP issuing too many lock requests

A UAP that is issuing too many lock requests should be modified so that it does not issue so many lock requests.

The `PDLOCKLIMIT` operand in the client environment definition can also be used to set a maximum number of lock requests that can be issued by any UAP.

(b) When there is no UAP issuing too many lock requests

It may be that too few locked resources management tables are available. The

following actions can be taken in this case:

When there is not enough shared memory space available at the unit (shared memory cannot be increased)

Do not execute concurrently many UAPs that use locked resources management tables.

When there is enough shared memory space at the unit (shared memory can be increased)

Modify the HiRDB system definition. The operand to be modified depends on the type of server. The server at which there are too few locked resources management tables can be determined from the `5`. *Server name* information in the locked resources management table information.

When a shortage of locked resources management tables has occurred at the front-end server, the value of the `pd_fes_lck_pool_size` operand in the front-end server definition should be increased.

When a shortage of locked resources management tables has occurred at a server other than the front-end server, the value of the `pd_lck_pool_size` operand in the corresponding server definition should be increased.

Note

Before the HiRDB system definition is modified, the entire HiRDB or the entire unit whose definition is to be modified must be terminated normally, then the HiRDB system definition can be modified. If only the applicable server is terminated with the `pdstop -s`, modifications made to the definition will not be effective.

8.8 Monitoring UAP status (skipped effective synchronization point dump monitoring facility)

When a UAP updates a database continuously because of an infinite loop, many overwrite disabled system log files are created because the synchronization point dumps cannot be validated. Once all the system log files have been placed in overwrite disabled status, HiRDB terminates abnormally.

If HiRDB is terminated abnormally or forcibly while more than half of all system log files are in overwrite disabled status, HiRDB cannot be restarted because there are too few system log files for rollback processing. In this case, new system log files must be added in order to restart HiRDB. This also results in a longer restart processing time.

To avoid such problems, HiRDB provides the skipped effective synchronization point dumps monitoring facility.

(1) Skipped effective synchronization point dumps monitoring facility

If a UAP experiences an infinite loop, synchronization point dump validation processing may not be executed consecutively (synchronization point dump validation processing may be skipped). When the number of consecutively skipped validations reaches a set value, the offending transaction is stopped forcibly and rollback processing is performed. This capability is provided by the *skipped effective synchronization point dumps monitoring facility*. To use this facility, specify the `pd_spd_syncpoint_skip_limit` operand.

(2) Value to be specified in the pd_spd_syncpoint_skip_limit operand

Normally, the `pd_spd_syncpoint_skip_limit` operand is set to `0`. When its value is `0`, HiRDB calculates automatically the maximum number of times skipping is permitted. However, if system instability occurs with `0` specified, one of the methods shown in (3) below can be used to calculate an appropriate value for this operand.

When any one of the following conditions is satisfied, the value calculated by either of the methods described in (3) will provide a more precise result than the value HiRDB calculates:

- There are five or fewer generations of system log files that can be used as current files.
- Transactions that take a long time to process are being executed concurrently.
- Data replication transaction processing takes a long time to complete on the HiRDB being updated (when linked with HiRDB Datareplicator).

Also in the following cases you should not use automatic calculation, but should calculate a value using one of the methods described in (3):

- The KFPS02101-I message is issued.
- HiRDB terminated abnormally with abort code Psspc01.

(3) Calculation methods

The following two calculation methods are provided; use one of them to calculate an appropriate value to use:

- Method based on the byte count of the output system logs
- Method based on the byte count of all system logs

Specify in the pd_spd_syncpoint_skip_limit operand a value that is slightly smaller than the value obtained by these calculations. The value specified in this operand takes effect following the first synchronization point dump validation performed after the next HiRDB startup (or restart).

(a) Method based on the byte count of the output system logs

Use the following formula to obtain the value:

Formula

$$\{(\uparrow a \div b \uparrow \div c) \div d\} - 1$$

a:

Sum (in bytes) of the system log information output by the transaction that updates the largest volume of database data and the system log information output by other transactions that are executing concurrently. For details about how to obtain the byte count of system logs, see the manual *HiRDB Version 8 Installation and Design Guide*.

■ When HiRDB Datareplicator is being used

When data replication transaction processing on the HiRDB being updated takes a long time, this monitoring facility may roll back the data replication transaction on the HiRDB being updated. Therefore, you must also add the byte count of system logs output by data replication transaction processing. Add the value obtained by the following formula:

byte-count-of-system-logs-output-by-data-replication-transaction-processing = Σ (*byte-count-of-system-logs-output-by-the-transaction-that-updates-the-largest-amount-of-database-data*)

Σ represents the log volume output by transactions as specified in the cmtintvl operands (trncmtintvl and tblcmtintvl) of the HiRDB Datareplicator replication environment definitions.

b:

System log file record length. You can obtain the record length with the `pdlogls` command.

c:

Average number of records in a system log block. Normally, this is roughly $3 \times 4096 \div b$. You can use the following formula to obtain a precise value:

$\uparrow$ *average-length-of-system-log-output-block* $\div b \uparrow$

You can obtain the average length of the system log output blocks from the system activity statistical information produced by the statistics analysis utility (`OUTPUT BLOCK LENGTH`).

d:

Value of the first parameter of the `pd_log_sdinterval` operand (which specifies the synchronization point dump acquisition interval in terms of the volume of system logs that are output).

(b) Method based on the byte count of all system logs

Use the following formula to obtain the value:

Formula

$\{(a \times b \times c) \div d\} \div e$

a:

Number of system log files that can placed in swappable target status while HiRDB is running.

b:

Number of records in a system log file. If the number of records differs between files, use the average number of records.

c:

Ratio of skipped synchronization point dump validations. Use the ratio of the number of files placed in overwrite disabled status to the total number of system log files.

- For a HiRDB/Single Server, use a value of 0.333 or less. If the number of guaranteed-valid generations is 2, use a value of 0.167 or less.
- For a back-end server, use a value of 0.333 or less. If the number of guaranteed-valid generations is 2, use a value of 0.167 or less.
- For the dictionary server, use a value of approximately 0.5.
- For a front-end server, use a value of approximately 0.7.

d:

Normally, this is roughly 3 × 4096 ÷ *f*. You can use the following formula to obtain a more precise value:

↑ *average-length-of-system-log-output-block* ÷ *f* ↑

You can obtain the average length of the system log output blocks from the system activity statistical information produced by the statistics analysis utility (`OUTPUT BLOCK LENGTH`).

e:

Value of the first parameter of the `pd_log_sdinterval` operand (which specifies the synchronization point dump acquisition interval in terms of the volume of system logs that are output).

f:

System log file record length. You can obtain the record length with the `pdlogls` command.

(4) When the value of the pd_spd_syncpoint_skip_limit operand is not appropriate

If the specified value is too large, it may not be possible to overwrite any of the system log files. If this happens, HiRDB terminates abnormally and cannot be restarted unless new system log files are added.

If the specified value is too small, there may be an increase in the number of transactions that are rolled back forcibly.

(5) When not to use the skipped effective synchronization point dumps monitoring facility

1. During batch processing that involves updating of a large amount of data and the amount of system log information that is output before the commit statement is issued is more than one-third of the total size of all system log files.
2. When the sum of the amount of system log information output by the transaction that updates the largest amount of database data and the amount of system log information output by transactions that execute concurrently is more than one-third of the total size of all system log files.

For details on obtaining the amount of system log information, see the manual *HiRDB Version 8 Installation and Design Guide*.

(6) Transactions that are not rolled back

Even if the number of consecutively skipped synchronization dump points exceeds the value specified in the `pd_spd_syncpoint_skip_limit` operand, the following transactions are not rolled back:

- Transactions that are already being rolled back.
- Transactions waiting for a phase 2 commit completion instruction from OpenTP1.
- Transactions generated by a utility.

(7) Notes

If you execute the `pdlogswap` command several times consecutively while HiRDB is running, the number of system log files that can be used as primary files decreases. This tends to increase the probability that a unit will terminate abnormally, due to there being an insufficient number of system log files.

(8) If HiRDB Datareplicator is being used

If data replication transaction processing on the HiRDB being updated (target HiRDB) takes a long time, the skipped effective synchronization point dumps monitoring facility may roll back the data replication transaction. If this happens, the target HiRDB issues the `KFPS00993-I` message (`REQUEST= abnormal_tran_end`), and the instance of the HiRDB Datareplicator on the target HiRDB side issues the `KFRB03007-W` and `KFRB03013-I` messages. The following shows how to handle this situation:

Procedure

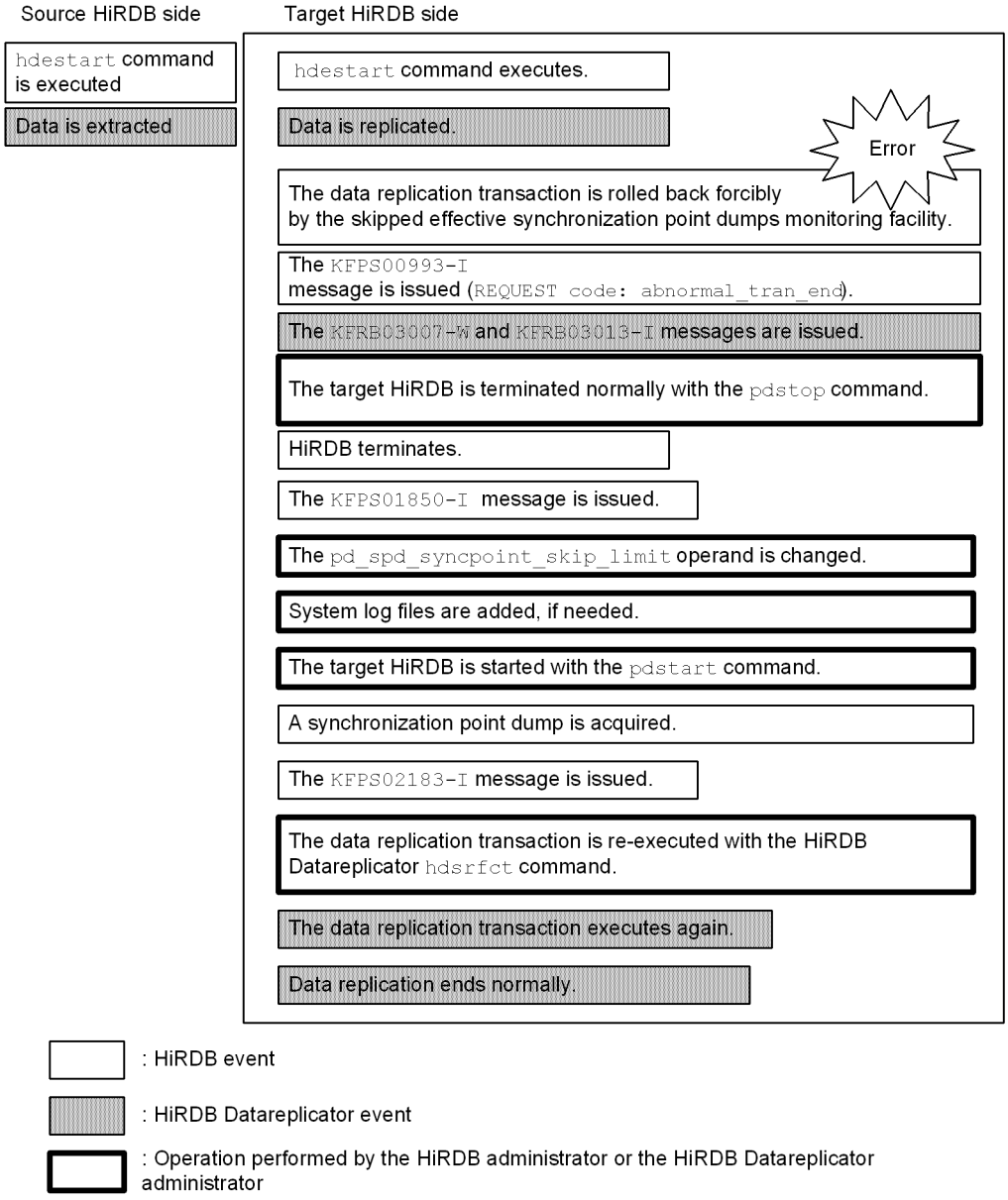
1. Use the `pdstop` command to terminate the target HiRDB normally.*
2. Change the value of the `pd_spd_syncpoint_skip_limit` operand. For details about an appropriate value to specify, see *(3)(a) Method based on the byte count of the output system logs*.
3. Determine whether the number of system log file generations satisfies the following condition; if it does not, add enough system log files to satisfy this condition:

   value-of-ps_spd_syncpoint_skip_limit-operand-after-change ≤ *number-of-system-log-file-generations* ÷ 3
4. Use the `pdstart` command to start the target HiRDB normally.
5. Use the `hdsrfctl` command of the HiRDB Datareplicator on the target HiRDB side to re-execute the data reflection transaction.

* When you use the system reconfiguration command (`pdchgconf` command), you do not need to restart HiRDB normally, because the `pdchgconf` command allows you to modify HiRDB system definitions while HiRDB is running. Note that HiRDB Advanced High Availability is required in order to use this command. For details about modifying HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

Figure 8-13 shows the operational flow when a data replication transaction is rolled back forcibly by the skipped effective synchronization point dumps monitoring facility.

Figure 8-13: Operational flow when a data replication transaction is forcibly rolled back by the skipped effective synchronization point dumps monitoring facility



8.9 Output of warning information about the time required for SQL execution (SQL runtime warning output facility)

If SQL execution time exceeds either of the limits listed below, warning information about that SQL can be output.

- A percentage of the client maximum wait time (value of PDCWAITTIME operand) that has been set
- Amount of elapsed time that constitutes trigger for output of a warning to a file

In the following explanations, PDCWAITTIME refers to the client maximum wait time, and a PDCWAITTIME timeout is an exceedance of the client maximum wait time.

8.9.1 Overview of the SQL runtime warning output facility

After SQL execution, HiRDB determines the SQL execution time. If this amount of time exceeds a preset elapsed time for output of warning information, the following warning information about that SQL is output; this facility is called the *SQL runtime warning output facility*:

- SQL runtime warning information file
- Warning message (KFPA20009-W message)

(1) Reasons for using the SQL runtime warning output facility

The SQL runtime warning output facility can be used for the following purposes:

- To detect in advance the possibility of a PDCWAITTIME timeout occurring for a UAP whose response time to HiRDB server processes has increased, due to an increase in the volume of data
- To collect for use as tuning data information on SQL code whose response wait time meets or exceeds a specific value

(2) Setting the elapsed time basis for output of warning information

The amount of time on the basis of which warning information is to be output is called the *elapsed time basis for output of warning information*. When SQL execution time exceeds the elapsed time basis for output of warning information, warning information is output. The elapsed time basis for output of warning information is determined as either of the following:

- A percentage of the value specified in the PDCWAITTIME operand
- An elapsed time trigger for output of a warning to a file; you can specify a very precise output trigger time value (in milliseconds, for example).

(3) SQL statements monitored by the SQL runtime warning output facility

The SQL runtime warning output facility monitors all SQL statements except for `CONNECT` statements.

(4) Actions when warning information is output

When information warning that a `PDCWAITTIME` timeout may occur is output, you should take the following actions based on the output information:

1. Determine if lockout has occurred.
2. Determine if a network failure has occurred.
3. Tune the SQL code.
4. Increase the value specified in the `PDCWAITTIME` operand.
5. Check if the execution time of the SQL code has increased because of an increase in the number of data transactions.

(5) Conditions under which warning information is output

When the SQL runtime warning output facility is being used, warning information may be output, even when the execution time of SQL code is less than the set time. Warning information (message only) may also be output, even when the SQL runtime warning output facility is not being used. Table 8-7 describes the conditions under which warning information is output by the SQL runtime warning output facility.

Table 8-7: Conditions under which warning information is output by the SQL runtime warning output facility

Condition		Warning information that is output	
		SQL runtime warning information file	KFPA20009-W message
When the SQL runtime warning output facility is being used	The SQL execution time meets or exceeds the set time.	Yes	Yes
	Server process is terminated forcibly due to a `PDCWAITTIME` timeout.	Part	Part
	Server process is terminated forcibly for some other reason.	Part	Part

Condition		Warning information that is output	
		SQL runtime warning information file	KFPA20009-W message
When the SQL runtime warning output facility is not being used	The SQL execution time meets or exceeds the set time.	No	No
	Server process is terminated forcibly due to a PDCWAITTIME timeout.	No	Part
	Server process is terminated forcibly for some other reason.	No	Part

Legend:

Yes: The indicated information is output.

Part: The indicated information is partially output.

In addition, the SQL runtime warning information file or the KFPA20009-W message may not be output depending on the timing of the forced termination.

Note, also, that you can suppress re-output of the SQL runtime information file and the KFPA20009-W message with the pd_dump_suppress_watch_time operand.

No: The indicated information is not output.

Note

When SQL execution time exceeds the value in the PDCWAITTIME operand, the server process is terminated forcibly regardless of whether the SQL runtime warning output facility is being used.

(6) Relationship between PDCWAITTIME and the SQL runtime warning output facility

Figures 8-14 and 8-15 illustrate the relationship between PDCWAITTIME and the SQL runtime warning output facility.

Figure 8-14: Relationship between PDCWAITTIME and the SQL runtime warning output facility (1 of 2)

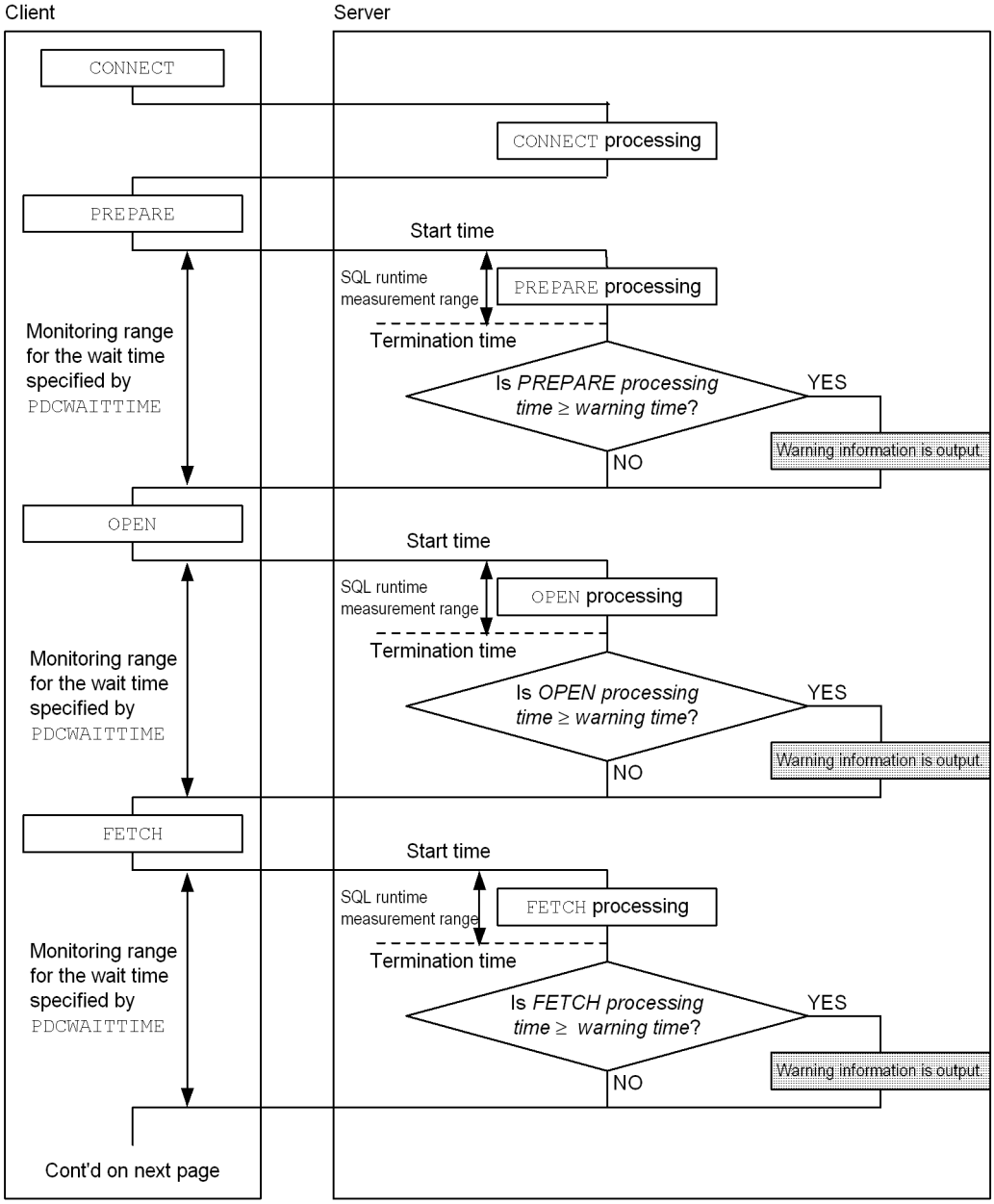
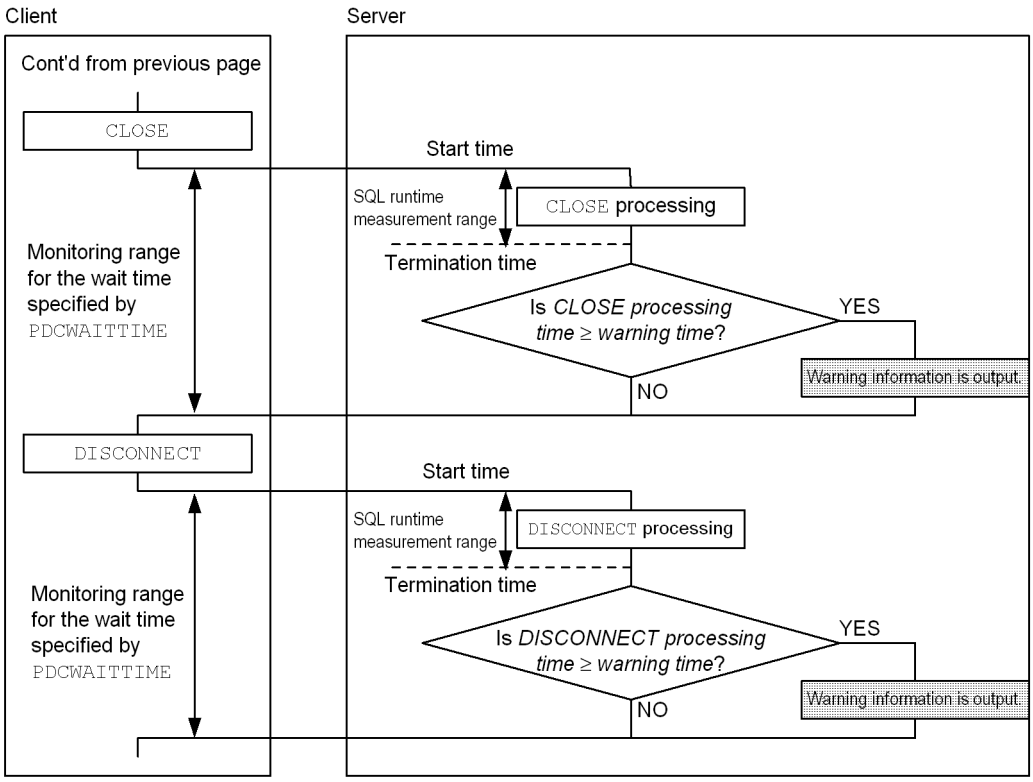


Figure 8-15: Relationship between PDCWAITTIME and the SQL runtime warning output facility (2 of 2)



Explanation

- The range of measuring SQL execution time is from the time a request sent from a client is received by the server until the execution result in response to the request is returned.
- When the SQL execution result is returned to the client, HiRDB determines the SQL execution time. If the SQL execution time equals or exceeds the set time, the warning information is output.

8.9.2 Using the SQL runtime warning output facility

To use the SQL runtime warning output facility, you must specify the following:

- PDCWAITTIME operand
- Percentage of the value specified in the PDCWAITTIME operand or the amount of elapsed time that constitutes the trigger for output of warning information to a file

- Output directory for the SQL runtime warning information files
- Maximum size of an SQL runtime warning information file

(1) Specifying the PDCWAITTIME operand

Specify the client maximum wait time in the PDCWAITTIME operand of the client environment definitions. If you specify 0 (or nothing) in the PDCWAITTIME operand, the SQL runtime warning output facility is not applied to SQL statements executed from that HiRDB client.

For details about the PDCWAITTIME operand, see the manual *HiRDB Version 8 UAP Development Guide*.

(2) Specifying a percentage of the value specified in the PDCWAITTIME operand or an amount of elapsed time that is to constitute a trigger for output of a warning to a file

Use the following operands to specify the conditions under which warning information is to be output:

- PDCWAITTIMEWRNPNT operand (client environment definition)
- pd_cwaittime_wrn_pnt operand (HiRDB system definition)

(a) Specifying a percentage of the value specified in the PDCWAITTIME operand

Specify in the PDCWAITTIMEWRNPNT and pd_cwaittime_wrn_pnt operands a percentage (between 0 and less than 100) of the value specified in the PDCWAITTIME operand. For example, if you specify 90 (%) in the PDCWAITTIMEWRNPNT and pd_cwaittime_wrn_pnt operands when 100 (seconds) is specified in the PDCWAITTIME operand, HiRDB monitors the SQL execution time once the SQL begins executing. If the SQL execution time is determined to be 90 or more seconds, warning information is output.

You can specify an unsigned integer or an unsigned decimal number (a number that includes a decimal point and up to six decimal places).

(b) Specifying an amount of elapsed time that is to constitute a trigger for output of warning information to a file

Specify in the above operands a value between 0 and less than the value specified in the PDCWAITTIME operand as the amount of elapsed time that is to constitute the trigger for output of SQL runtime warning information.

If the value specified in the PDCWAITTIMEWRNPNT operand is greater than the value specified in the PDCWAITTIME operand, an error occurs during a connection request.

Note:

- The PDCWAITTIMEWRNPNT operand specification takes precedence over specification of the pd_cwaittime_wrn_pnt operand.
- If 0 is specified in the PDCWAITTIMEWRNPNT operand, the SQL runtime warning output facility does not monitor SQL statements executed from that HiRDB client.

Reference note:

HiRDB determines the amount of elapsed time for output of warning information from the following formulas based on the values in the above operands:

- When an unsigned integer is used to specify a percentage of the PDCWAITTIME operand:

  elapsed-time-basis-for-output-of-warning-information = MAX($\downarrow$ (*a* × *b*) ÷ 100 $\downarrow$, 1)

  a: Value of the PDCWAITTIME operand

  b: Value of the PDCWAITTIMEWRNPNT or pd_cwaittime_wrn_pnt operand (whichever was applied, in accordance with the precedence level of these operands)

- When an unsigned decimal number is used to specify a percentage of the PDCWAITTIME operand:

  elapsed-time-basis-for-output-of-warning-information = (*a* × *b*) ÷ 100

  a: Value of the PDCWAITTIME operand

  b: Value of the PDCWAITTIMEWRNPNT or pd_cwaittime_wrn_pnt operand (whichever was applied, in accordance with the precedence level of these operands)

  Note that up to six decimal places are valid for the elapsed time basis for output of warning information. The seventh and subsequent decimal places are truncated.

- When an amount of elapsed time that is to constitute a trigger for output of warning information to a file is specified:

  elapsed-time-basis-for-output-of-warning-information = *absolute-value-of-time-that-constitutes-trigger-for-output-of-warning-information-to-file*

  The accuracy of the timer used by HiRDB in a server depends on the platform. Therefore, if the timer's accuracy is low, warning information may not be output in some cases, even though the actual SQL execution time exceeded the elapsed time basis for output of warning information.

(3) Specifying the output directory for the SQL runtime warning information files

Specify the output directory for the SQL runtime warning information files in the pd_cwaittime_report_dir operand. Two files are created in this directory, pdcwwrn1 and pdcwwrn2.

Do not specify this operand if you do not wish to have SQL runtime warning information files output. If omitted, only the KFPA20009-W message will be issued.

Note that if the OS detects an error, such as a file system error or that the user does not have write privileges for the directory or file, no warning will be output to the SQL runtime warning information file. In such a case, execution processing of the SQL code will continue.

(4) Specifying the maximum size of an SQL runtime warning information file

Specify the pd_cwaittime_report_size operand to change the maximum size for SQL runtime warning information files. The value specified in this operand is the size of one SQL runtime warning information file. When you specify this value, take into account that two SQL runtime warning information files are created. For example, if you specify 10,000, then two files, each with a maximum size of 10,000 bytes, will be created in the directory.

Remarks

- When the amount of data output to the first file exceeds the value set in this operand, output will switch to the other file. If the amount of data output to the second file exceeds this value, output will switch back to the first file. Output will continue to switch between the two files in this manner. Each time output is switched to a file that has already been used, the existing information in it is overwritten.
- If the amount of SQL runtime warning information that is output in a single output session exceeds the maximum file size, some of the information will be lost. Such SQL runtime warning information is output only to the point at which the file becomes full. When this happens, a hash mark (#) is output at the end of the SQL runtime warning information.

8.9.3 Information output to the SQL runtime warning information file

(1) Viewing the SQL runtime warning information file

You can view SQL runtime warning information with any text editor or software capable of opening a text file.

Note that, in a HiRDB/Parallel Server environment, the warning information is output to the server machine running the front-end server to which the UAP that issued the offending SQL is connected.

Remarks

- To determine which SQL runtime warning information file is the current file, use an OS command (ls command, for instance) to check the update dates and times of the files. The file with the most recent update date and time is the one that is being used currently.

- The output target file after HiRDB starts is the one with the most recent update date and time.
- SQL runtime warning information is written to the file starting at the previous ending position, which means that warning information is displayed in the file chronologically.
- Because the current SQL runtime warning information file is closed when an SQL statement has executed, you can use OS commands to back up or view the file while SQL code is not executing, without worrying about interfering with the file while it is being written to. Even when SQL code is being executed, you can manipulate the other file (the one not being used) without having to worry about destroying the file that is being written to.

(2) Output format of SQL runtime warning information

The following shows the output format of the SQL runtime warning information (when an unsigned integer is used to specify a percentage of the PDCWAITTIME operand):

```
** SQL CWAITTIME WARNING INFORMATION 07-01    2002/07/04 14:32:22 **
  REASON(01)
  CWAITIME(600) CWAITTIME_WRN_PNT(70) CWAITTIME_WRN_TIME(420)

* UAP INFORMATION *
  UAP_NAME(userprog1) CLTPID(408)
  IP_ADDR(196.12.42.146) SERVICE_NAME(service1)
  USERID(hiuser01) START_TIME(2002/07/03 20:24:42)

* SERVER INFORMATION *
  HOST(host03) PORT(1146)
  SVRNAME(fes1) SVRPID(905)

* SQL INFORMATION *
  OPTIMIZE_LEVEL(132768) ADDITIONAL_OPTIMIZE_LEVEL(3)
  ISOLATION_LEVEL(2)
  SQLOBJ_SIZE(2608) SQLCOUNT(1)

CNCTNO     SQL-       OP   SEC  SQL   SQL   START-TIME          END-TIME EXEC-
           COUNTER    CODE NO   CODE  WARN                             TIME
---------- ---------- ---- ---- ----- ----- ------------------- -------- -----
        10         10 AUI2 1890     0 -0000 2002/07/04 14:32:22 14:39:30   428

* SQL MESSAGE *
  "*" [*]

* SQL STATEMENT *
  DELETE FROM ZAIKO WHERE ZNO=1
```

The following shows the output format of the SQL runtime warning information (when an unsigned decimal number is used to specify a percentage of the PDCWAITTIME operand or an amount of elapsed time that is to constitute a trigger for

output of warning information to a file is specified):

```
** SQL CWAITTIME WARNING INFORMATION 07-01    2002/07/04 14:32:22.100000 **
  REASON(01)
  CWAITIME(600) CWAITTIME_WRN_PNT(70.001000) CWAITTIME_WRN_TIME(420.006000)

* UAP INFORMATION *
  UAP_NAME(userprog1) CLTPID(408)
  IP_ADDR(196.12.42.146) SERVICE_NAME(service1)
  USERID(hiuser01) START_TIME(2002/07/03 20:24:42)

* SERVER INFORMATION *
  HOST(host03) PORT(1146)
  SVRNAME(fes1) SVRPID(905)

* SQL INFORMATION *
  OPTIMIZE_LEVEL(132768) ADDITIONAL_OPTIMIZE_LEVEL(3)
  ISOLATION_LEVEL(2)
  SQLOBJ_SIZE(2608) SQLCOUNT(1)

CNCTNO     SQL-       OP   SEC  SQL   SQL
           COUNTER    CODE NO   CODE  WARN
---------- ---------- ---- ---- ----- -----
        10         10 AUI2 1890     0 -0000

START-TIME                 END-TIME        EXEC-TIME

-------------------------- -------------- ------------
2002/07/04 14:32:22.122222 14:39:30.822223   428.700001

* SQL MESSAGE *
  "*" [*]

* SQL STATEMENT *
  DELETE FROM ZAIKO WHERE ZNO=1
```

Table 8-8 explains the SQL runtime warning information that is output.

Table 8-8: Description of SQL runtime warning information that is output

Output information	Header name	Description	Maximum number of characters (bytes)	Output?	
				Cond. 1	Cond. 2
HiRDB version	SQL CWAITTIME WARNING INFORMATION	HiRDB version, in the format *VV-RR-ZZ*. If there is no *ZZ*, it is not output.	8	Y	Y
Output time		Displays the time at which the warning was written into the SQL runtime warning information file, in one of the following formats: • *YYYY/MM/DD hh:mm:ss* • *YYYY/MM/DD hh:mm:ss.uuuuuu* (*where uuuuuu* is microseconds)	19 or 26	Y	Y
Reason code	REASON	Reason the warning was output to the SQL runtime warning information file: • 00: The server process was terminated forcibly because the UAP was terminated forcibly. • 01: The SQL execution time exceeded the set time.	2	Y	Y
Value of PDCWAITTIME operand	CWAITIME	Value (in seconds) set for the PDCWAITTIME operand in the client environment definitions.	5	Y	Y
Percentage specified in operand, or amount of elapsed time	CWAITTIME_ WRN_PNT	Displays the percentage (%) specified in the PDCWAITTIMEWRNPNT operand or the pd_cwaittime_wrn_pnt operand, or amount of time (seconds). The value that is output is the one that was applied, in accordance with the precedence level of the operands.	2 or 12	Y	Y
Elapsed time basis for output of warning information	CWAITTIME_ WRN_TIME	Amount of time (seconds) on the basis of which warning information was output.[1]	5 or 12	Y	Y

Output information	Header name	Description	Maximum number of characters (bytes)	Output?	
				Cond. 1	Cond. 2
Name of UAP	UAP_NAME	UAP name specified in the PDCLTAPNAME operand of the client environment definitions.	30	Y	Y
Process number	CLTPID	Client process number.	10	Y	Y
IP address	IP_ADDR	IP address of the client that executed the UAP.	15	Y	Y
Service name	SERVICE_ NAME	Depending on the type of UAP, the service name is indicated as follows: • For an OpenTP1 UAP In the case of a service in which an OpenTP1 SUP (Service Utilization Program) has issued a request to an SPP (Service Provider Program), or a service in which TP1/ Message Control is issuing a request to an MHP (Message Handling Program), the name of the service is output. In any other case, an asterisk (*) is output. • For other than an OpenTP1 UAP An asterisk (*) is output.	31	S	S
Authorization identifier	USERID	Name of the connected user.	8	Y	Y
UAP start time	START_TIME	Time execution of the UAP started, in the format *YYYY/MM/DD hh:mm:ss*.	19	Y	Y
Host name	HOST	Name of the host on which the server process is running.	30	Y	Y
Port number	PORT	Communications port number used by the server process.	5	Y	S

Output information	Header name	Description	Maximum number of characters (bytes)	Output?	
				Cond. 1	Cond. 2
Server name	SVRNAME	Name of the server. For a HiRDB/Single Server, the name of the single server is output; for a HiRDB/Parallel Server, the name of the front-end server is output.	8	Y	Y
Process number	SVRPID	Process number of the server process.	10	Y	Y
SQL optimization option	OPTIMIZE_LEVEL	Value (in decimal) of the SQL optimization option. If this value cannot be obtained, an asterisk (*) is output.	10	Y	Y
SQL extension optimizing option	ADDITIONAL_OPTIMIZE_LEVEL	Value (in decimal) of the SQL extension optimizing option. If this value cannot be obtained, an asterisk (*) is output.	10	Y	Y
Data guarantee level	ISOLATION_LEVEL	Value set as the data guarantee level. If this value cannot be obtained, an asterisk (*) is output.	10	Y	Y
Size of SQL object	SQLOBJ_SIZE	Size (in bytes) of the SQL object. If this value cannot be obtained, an asterisk (*) is output.	10	S	N
SQL processed lines count	SQLCOUNT	Number of lines processed by the SQL code (the number of lines read by a SELECT statement, for example). If this value cannot be obtained, an asterisk (*) is output. For details about what is output, see the manual *HiRDB Version 8 SQL Reference*.	10	S	N
Connection sequence number	CNCTNO	Sequence number that increments each time a server receives a CONNECT request.	10	Y	Y

Output information	Header name	Description	Maximum number of characters (bytes)	Output?	
				Cond. 1	Cond. 2
SQL counter	SQL-COUNTER	Sequence number that increments each time an SQL statement is received. If this value cannot be obtained, an asterisk (*) is output.	10	Y	N
Operation code	OP CODE	Operation code that corresponds to the SQL code. If this value cannot be obtained, an asterisk (*) is output.	4	Y	N
Section number	SEC NO	Section number that corresponds to the SQL code. If this value cannot be obtained, an asterisk (*) is output.	4	S	N
SQLCODE	SQL CODE	SQLCODE resulting from execution of the SQL statement. If this value cannot be obtained, an asterisk (*) is output.	5	Y	N
Warning information	SQL WARN	Warning information (in hexadecimal).[2] If this value cannot be obtained, an asterisk (*) is output.	5	Y	N
SQL start time	START-TIME	Displays the date and time the SQL execution request was received from the client, in one of the following formats: • *YYYY/MM/DD hh:mm:ss YYYY/MM/DD* • *hh:mm:ss.uuuuuu* (*where uuuuuu* is microseconds) If this value cannot be obtained, an asterisk (*) is output.	19 or 26	Y	S

Output information	Header name	Description	Maximum number of characters (bytes)	Output?	
				Cond. 1	Cond. 2
SQL end time	END-TIME	Displays the end time of processing in response to the request received from the client, in one of the following formats: • *hh*:*mm*:*ss* • *hh*:*mm*:*ss*.*uuuuuu* (*where uuuuuu* is microseconds) If the server process was terminated forcibly, this value indicates the time at which processing ended.	8 or 15	Y	Y
SQL execution time	EXEC-TIME	Amount of time required to process the request received from the client (in seconds). If the server process was terminated forcibly, this value indicates the amount of time until processing ended. If this value cannot be obtained, an asterisk (*) is output.	5 or 12	Y	S
SQL message	SQL MESSAGE	Message output while the SQL code was executing. If this value cannot be obtained, an asterisk (*) is output.	254	S	N
		Information in square brackets is system maintenance information. If this value cannot be obtained, an asterisk (*) is output.	21	Y	S
SQL statement	SQL STATEMENT	SQL statement. If a comment has been inserted in the SQL statement, or if an SQL optimization option has been specified, this information is also included in the output. If this value cannot be obtained, an asterisk (*) is output.	2,000,000	S	S

Legend:

Cond. 1: The SQL execution time reached the set time.

Cond. 2: The server process was terminated forcibly before SQL execution reached the set time.

Y: Always output

S: Sometimes output

N: Never output

[1] The following formula is used to obtain the time basis for output of warning information:

- When an unsigned integer is used to specify a percentage of the PDCWAITTIME operand:

elapsed-time-basis-for-output-of-warning-information = MAX($\downarrow$ ($a \times b$) $\div$ 100 $\downarrow$, 1)

a: Value of PDCWAITTIME operand

b: Value of PDCWAITTIMEWRNPNT operand or pd_cwaittime_wrn_pnt operand (whichever was applied, in accordance with the precedence level of these operands)

- When an unsigned decimal number is used to specify a percentage of the PDCWAITTIME operand:

a: Value of the PDCWAITTIME operand

b: Value of the PDCWAITTIMEWRNPNT operand or pd_cwaittime_wrn_pnt operand (whichever was applied, in accordance with the precedence level of these operands)

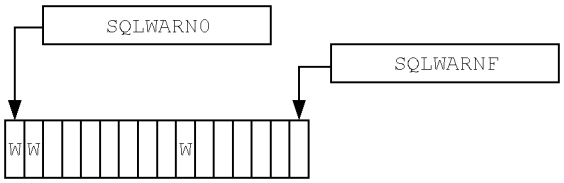
Note that up to six decimal places are valid for the elapsed time basis for output of warning information. The seventh and subsequent decimal places are truncated.

- When the amount of elapsed time that becomes a trigger for output of warning information to a file is specified:

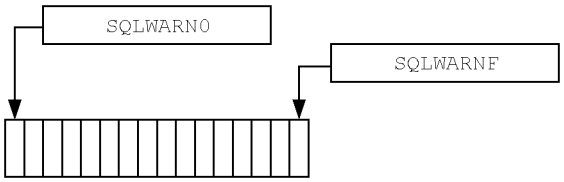
elapsed-time-basis-for-output-of-warning-information = *absolute-value-of-time-that-becomes-trigger-for-output-of-warning-information-to-file*

[2] Starting from the left side, one bit is allocated for each of 16 warning items (SQLWARN0 to SQLWARNF). A 1 is set for a warning item whose warning flag is set, and a 0 is set for a warning item whose warning flag is not set. From this, a 16-bit hexadecimal number is obtained. This 16-bit hexadecimal value is then output as a 4-digit hexadecimal number. If one or more warning flags are set, W is prefixed to the 4-digit hexadecimal number. If no warning flags are set, a hyphen (-) is prefixed to the hexadecimal number.

Example 1: If the warning information is as shown in the following, `WC040` is output:



Example 2: If the warning information is a shown in the following, `-0000` is output:



8.9.4 Output of the KFPA20009-W message

The `KFPA20009-W` message is output to the message log file and the syslogfile.

Normally, the `KFPA20009-W` message (`reason code = 01`) is not re-issued at the same unit within a five-minute period.

However, it may be issued more than once within five minutes if the SQL runtime warning information output conditions are satisfied simultaneously by multiple server processes.

8.9.5 Notes

1. Even when it is not producing warning information, the SQL runtime warning output facility increases the number of system calls issued because the facility still must obtain SQL start times and execution times.
2. Depending on the exact time at which SQL runtime warning information is output, both of the conditions for output of a SQL runtime warning information file explained in *8.9.1(5) Conditions under which warning information is output* may be satisfied. This means that SQL runtime warning information may be output twice for the same SQL statement.
3. A small period of time elapses from when the client issues a processing request until the SQL runtime warning output facility begins measuring the SQL execution time, and from when the facility has finished measuring the SQL execution time until its execution results arrive at the client, due to output processing, communications processing, and other factors handled by the facility. If this processing time takes too long due to the network or I/O load, the following behaviors may occur:

- Even though the client wait time satisfied the conditions for output of a SQL runtime warning information file, no SQL runtime warning information file was output.
- Although the server process was terminated forcibly due to a `PDCWAITTIME` timeout, the SQL execution time that was output to the SQL runtime warning information file does not satisfy the conditions for warning information to be output.

8.10 Monitoring the execution time of UAPs and utilities (reducing the effects of nonresponding programs)

This section explains how to minimize the effects when an error occurs that causes a UAP or utility to stop responding.

When an error (such as a communications error, an intermittent failure such as a power flicker, or a disk error) occurs during execution of an overnight batch job or other process, causing a UAP or a utility to stop responding, execution of other UAPs and utilities may be affected adversely or even stopped. In the worst case, the effects may extend to online operations on the following business day. To minimize the adverse effects of UAPs or utilities that are not responding, you can specify the following operands:

- `PDCWAITTIME` operand of the client environment definitions
- `pdf_utl_exec_time` operand of the system command definitions

The `PDCWAITTIME` operand is used to monitor UAP execution time. If a UAP or utility does not terminate once the time specified in this operand has elapsed, that UAP or utility is terminated forcibly. This minimizes adverse effects on other UAPs or utilities. In these operands, we recommend you specify a time that assumes a high probability of such errors occurring.

For details about the `PDCWAITTIME` operand of the client environment definitions, see the manual *HiRDB Version 8 UAP Development Guide*.

8.11 Monitoring resource utilization factors

A warning message can be output when a resource utilization factor shown in Table 8-9 reaches a specified value. Whether or not the warning messages are to be output is specified in system common definition operands. These operands can also be used to specify output trigger values for the warning messages. For example, a warning message can be output when a resource's utilization factor reaches 90%.

Table 8-9: Resources whose utilization factors can be monitored

Operand specified*	Monitored resource	Output warning message
pd_max_users_wrn_pnt	Maximum number of concurrent connections, as specified in the pd_max_users operand	KFPS05123-W
pd_max_access_tables_wrn_pnt	Number of base table that can access as specified in the pd_max_access_tables operand	
pd_max_rdarea_no_wrn_pnt	Maximum number of RDAREAs, as specified in the pd_max_rdarea_no operand	
pd_max_file_no_wrn_pnt	Maximum number of HiRDB files comprising an RDAREA, as specified in the pd_max_file_no operand	
pdwork_wrn_pnt	HiRDB file system areas for work table files, as specified in the pdwork operand	
pd_max_list_users_wrn_pnt	Maximum number of users who can create lists, as specified in pd_max_list_users	
pd_max_list_count_wrn_pnt	Maximum number of lists that can be created by a user, as specified in pd_max_list_count	
pd_aud_file_wrn_pnt	Number of audit trail files that cannot be swapped	
pd_rdarea_list_no_wrn_pnt	Number of lists created in a server	KFPH22023-W

* When pd_watch_resource = AUTO is specified, the warning message is output when a resource utilization factor shown in Table 8-9 exceeds 80%. In this case, the operands shown in Table 8-9 need not be specified. To change message output triggers (to values other than 80%), use these operands to specify the appropriate output trigger values.

8.12 Monitoring the status of server processes (message queue monitoring facility)

A server on which server processing is down may experience degraded responses to UAPs, and even system freezes. This section explains how to use the message queue monitoring facility to monitor for server processes that are down.

Server processing down means that the server process has entered a state in which it is unable to perform any processing at all, due to CPU load-induced degradation of processing performance, I/O delay arising from an I/O error, or some other extreme condition.

(1) Overview of message queue monitoring facility

HiRDB's server process allocation processing uses a message queue. When server processing is down, messages can no longer be read from the message queue. Under HiRDB, if messages cannot be read from the message queue once a specified amount of time has been reached (which is called the *message queue monitoring time*), a warning message (`KFPS00888-W` message) and an error message (`KFPS00889-E` message) are issued. This capability is provided by the *message queue monitoring facility*. When these messages are issued, server processing may have gone down.

The message queue monitoring time is normally 600 seconds (10 minutes). You can change this time with the `pd_queue_watch_time` operand.

(2) Action to be taken when the warning message is issued

When the warning message is issued, the possibility exists that server processing has gone down, so you should take one of the following actions:

- Restart the unit
- Cancel transactions

(a) Restarting the unit

You can restore the server process that went down by restarting the unit in which it was running. Normally, when the message queue monitoring time is exceeded, HiRDB terminates forcibly the unit whose server processing has gone down.

If you do not want the unit to be terminated forcibly, specify `continue` in the `pd_queue_watch_timeover_action` operand.

(b) Cancelling transactions

If you do not take the action described in (a) above (including when you cannot), use the `pdcancel` command to stop the transactions that are executing on the server whose processing is down. If there were no transactions, use the OS's `kill` command to terminate the server that stopped responding. Afterwards, identify the cause of the

server process no-response and take appropriate action.

(3) Taking steps to prevent server processing from going down

Table 8-10 shows the causes of message queue stagnation and the corrective measures for the server whose message queue is being monitored.

Table 8-10: Causes of message queue stagnation and corrective measures

Cause	Server process being monitored				Corrective measure
	FES	BES	DS	SDS	
Messages cannot be read from the message queue because of a high CPU load.	Y	Y	Y	Y	Check the cause of the CPU overload and take appropriate corrective action.
Messages cannot be read from the message queue because an I/O error is delaying input/output.	Y	Y	Y	Y	Check the cause of the I/O error and take appropriate corrective action.
If the number of concurrent connection requests exceeds the value of the `pd_max_users` operand, the number of processes available for reading messages from the message queue is insufficient (this tends to occur when the high-speed connection facility is used).	Y	—	—	Y	Either reduce the number of concurrent connection requests or connect normally without using the high-speed connection facility. Alternatively, increase the value of the `pd_max_users` operand.

Cause	Server process being monitored				Corrective measure
	FES	BES	DS	SDS	
The number of processes available for reading messages from the message queue is insufficient when the number of active back-end servers or dictionary servers is smaller than the number of active front-end servers or utility servers (this tends to occur in a multiple front-end server environment).	—	Y	Y	—	Ensure that the correct values are specified in the `pd_max_bes_process`, `pd_max_dic_process`, and `pd_max_users` operands. Alternatively, reduce the number of connections.

Legend:

Y: Applicable

—: Not applicable

Reference note:

The number of HiRDB server processes that can be active is restricted by the following operands:

- `pd_max_server_process`

  If the number of active servers in a unit is large, carefully estimate the value to be specified for this operand. Also, if the standby-less system switchover facility is used, the estimate must take system switchover into account.

- `pd_max_bes_process`

  If multiple front-end servers or the standby-less system switchover (1:1) facility is used, carefully estimate the value to be specified for this operand.

- `pd_max_dic_process`

  If multiple front-end servers are used, carefully estimate the value to be specified for this operand.

- `pd_ha_max_server_process`

  If the standby-less system switchover (effects distributed) facility is used, carefully estimate the value to be specified for this operand.

- `pd_max_users`

  If the number of concurrent connections is large, an appropriate value must be specified.

(4) Remarks

You can use the `pdls -d scd` command to determine the time at which the last message was read from the message queue.

8.13 Monitoring the number of times server processes terminate abnormally (abnormal termination monitoring facility)

If server processes terminate abnormally often, servers may not be able to accept new services. However, because HiRDB itself does not usually terminate abnormally when a server process does, frequent server process abnormal terminations could bring online operations to an effective halt. To prevent this from occurring, the *abnormal termination monitoring facility* has been made available.

(1) Overview of the abnormal termination monitoring facility

If the number of times that a server process is terminated abnormally in a specified amount of time reaches the value specified in the `pd_down_watch_proc` operand, HiRDB (or the associated unit for a HiRDB/Parallel Server) also terminates abnormally. This capability is provided by the abnormal termination monitoring facility.

We recommend that you use this facility in conjunction with the system switchover facility. This way, if HiRDB terminates abnormally because server processes have terminated abnormally more than the specified number of times, the system will be switched over quickly. If this monitoring facility is not used, HiRDB does not terminate abnormally, and the system is not switched over.

Even if you do not use this facility, you can restart HiRDB, which will refresh memory and other resources, leading to improved processing efficiency.

If HiRDB terminates abnormally due to the abnormal termination monitoring facility, the `KFPS-01821-E` and `KFPS00729-E` messages are issued.

(2) Application range of the abnormal termination monitoring facility

This facility monitors processes that have terminated abnormally due to a `PDCWAITTIME` timeout or an abort. For a HiRDB/Single Server, it counts the number of times that single server processes terminate abnormally. For a HiRDB/Parallel Server, it counts the total number of times that front-end, back-end, and dictionary server processes in the unit terminates abnormally. Table 8-11 lists the factors that may cause server processes to terminate abnormally and indicates which of these are counted as an abnormal termination.

Table 8-11: Causes of abnormal termination of server processes and which are counted as an abnormal termination

Cause of abnormal termination of a server process	Counted as an abnormal termination?			
	Single server process	Front-end server process	Dictionary server process	Back-end server process
The value of the PDCWAITTME operand in the client environment definitions was exceeded.	Y	Y	N[1]	N[1]
The pdcancel command was executed.	N	N[2]	N	N
An internal forced termination occurred (HiRDB issued SIGKILL internally to stop the process).	Y[3]	Y[3]	N[1]	N[1]
An abort occurred.	Y	Y	Y	Y
Rollback occurred on an XA-connected UAP.	Y	Y	N	N
An abnormal termination other than the above occurred.	Y	Y	Y	Y

Legend:

Y: Counted as an abnormally terminating process

N: Not counted as an abnormally terminating process

[1] If an error is detected during a transaction branch, any abnormal termination of a front-end server process generated from that transaction branch is counted.

[2] If the pdcancel command is used to terminate forcibly a back-end server process or a dictionary server process, front-end server processes are terminated forcibly internally. In such a case, abnormal termination of the front-end server processes may be counted.

[3] If an error is detected during a global transaction issued by an OLTP system, any abnormal termination of a single server process or a front-end server process generated from that global transaction is counted.

(3) Specifying the abnormal termination monitoring facility

You use the pd_down_watch_proc operand to specify the period over which the number of server process abnormal terminations is to be monitored and the maximum number of times that server processes are to be allowed to terminate abnormally.

Example: `pd_down_watch_proc = 1000, 60`

In this case, the number of times server processes terminate abnormally is monitored in 60-second intervals. If the number of times server processes terminate abnormally in any 60-second interval exceeds 1000, HiRDB terminates abnormally.

(4) Notes

- The `KFPS01820-E` message is issued when a server process terminates abnormally. The `KFPS01820-E` message is also issued when the `pdcancel` command is used to terminate a server process abnormally; however, in this case, the abnormal termination is not counted.
- Use of a mutual system switchover configuration may actually cause traffic to increase, negating any benefits from this facility. The reason for this is because, if system switchover executes, multiple instances of HiRDB will become active on a single server machine. When you use the abnormal termination monitoring facility, we recommend that you restart HiRDB on the same system as the instance of HiRDB that terminated abnormally.

8.14 Monitoring the memory size of server processes (facility for monitoring the memory size of server processes)

This section explains the use of the facility for monitoring the memory size of server processes.

This facility need not be used in Linux.

(1) Overview

When the amount of work memory being used by a server process exceeds a given value, the process is terminated at the time shown in Table 8-12. This is called the *facility for monitoring the memory size of server processes*.

Table 8-12: Time of termination of a server process by the facility for monitoring the memory size of server processes

Server type	Process name	Process termination time
Single server	`pdsds`	At UAP disconnection[1]
Front-end server	`pdfes`	
Dictionary server	`pddic`	At transaction completion[2]
Back-end serverpdbes	`pdbes`	

[1] For purposes of this facility, UAP disconnection means any of the following:

- When the UAP terminates
- When the UAP issues a DISCONNECT statement
- When an OpenTP1 user server process that uses the HiRDB XA connection client library terminates
- When a transaction being executed in the user server process is completed (committed or rolled back) when transaction is specified in the `trn_rm_open_close_scope` operand in an OpenTP1 user server that uses the HiRDB XA connection client library
- During data commitment by HiRDB Datareplicator, when no commitment request is received before a certain amount of time (value specified in the target system's `discintvl` definition parameter) elapses after the end of the commitment information queue file has been detected

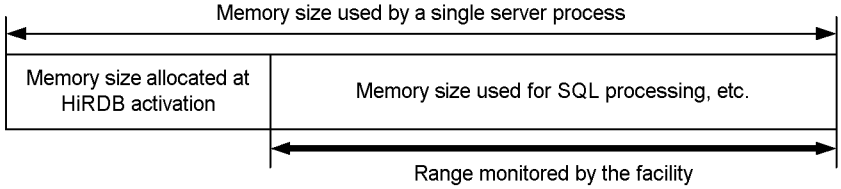
[2] For purposes of this facility, transaction completion means any of the following:

- When a UAP disconnection as explained in footnote 1 above occurs

- When the UAP issues a `COMMIT` or `ROLLBACK` statement
- When the UAP is rolled back internally by an SQL error
- When a transaction being executed in the user server process is completed (committed or rolled back) in an OpenTP1 user server that uses the HiRDB XA connection client library

The amount of memory being used by SQL processing, etc., is monitored by this facility. The amount of memory allocated at the time of HiRDB activation is not monitored. Figure 8-16 shows the monitoring range of this facility.

Figure 8-16: Monitoring range of facility for monitoring the memory size of server processes



(2) Advantages

The facility for monitoring the memory size of server processes addresses the following problem:

- The amount of memory used by server-resident processes in given SQL processing has become large, resulting in system memory shortages.

HiRDB releases unneeded memory. However, even when a program releases memory, the OS uses the memory management function in the applicable process to retain the memory area. Consequently, a process size that has grown large by using a large area even once never shrinks. Resident processes especially continue to impact the system. Because the facility for monitoring the memory size of server processes can terminate even resident processes, the memory shortage problem can be prevented.

This facility does not affect execution of jobs currently being used.

(3) Application standard

You should use this facility when the amount of memory used by HiRDB server processes becomes large, resulting in memory shortages.

(4) Specification of the facility for monitoring the memory size of server processes

To use the facility for monitoring the memory size of server processes, specify the maximum amount of memory to be used by a server process in the `pd_svr_castoff_size` operand in the server definition.

(5) Notes

1. If the pd_work_buff_mode operand is omitted or if pool is specified, the value of the pd_work_buff_size operand will be included in the memory size allocated when HiRDB starts. Consequently, the value of the pd_work_buff_size operand will not be monitored by this facility. Therefore, this facility may not have much effect if a large value is specified in the pd_work_buff_size operand.
2. In the cases shown in Table 8-13, the facility for monitoring the memory size of server processes will not have any effect because server processes that use large amounts of memory cannot be terminated.

   Table 8-13: Cases in which facility for monitoring the memory size of server processes is not effective

Condition	Ineffective cases
Single server or front-end server	• A UAP connected to a server process that uses a large amount of memory does not issue a DISCONNECT statement (including when HiRDB SQL Executor does not disconnect from HiRDB). • Many OpenTP1 user server processes that do not use the HiRDB XA connection client library do not issue DISCONNECT statements, increasing the amount of memory used by the connected server processes. • Many OpenTP1 user server processes remain that use the HiRDB XA connection client library and for which process is specified in the trn_rm_open_close_scope operand, increasing the amount of memory used by the connected server processes. (In this case, the facility for monitoring the memory size of server processes is applied when the user server processes terminate and are disconnected.)
Dictionary server or back-end server	• A UAP connected to a server process that uses a large amount of memory does not issue a COMMIT or ROLLBACK statement (including when HiRDB SQL Executor does not issue a COMMIT or ROLLBACK statement). • Many OpenTP1 user server processes that do not use the HiRDB XA connection client library do not issue a COMMIT or ROLLBACK statement, increasing the memory size of the connected server processes. • Many transactions being executed in the user server processes in an OpenTP1 user server that use the HiRDB XA connection client library are uncompleted (committed or rolled back), increasing the amount of memory used by the connected server processes. (In this case, the facility for monitoring the memory size of server processes is applied when the transactions are completed.)

Chapter

9. Modifying the System Operating Environment

This chapter explains the procedures for modifying the system operating environment.

This chapter contains the following sections:

9.1 Modifying HiRDB system definitions

Executor: HiRDB administrator

This section explains how to modify HiRDB system definitions (other than UAP environment definitions; for details about modifying UAP environment definitions, see the manual *HiRDB Version 8 System Definition*).

(1) Modification procedure

To modify the HiRDB execution environment, you must modify HiRDB system definitions. The following shows the procedure used to modify HiRDB system definitions. In this procedure, `$PDDIR/conf` indicates the directory that stores the unit control information definition file, while `$PDCONFPATH` indicates the directory that stores all other HiRDB system definition files.

Procedure

1. Create subdirectories in the `$PDDIR/conf` and `$PDCONFPATH` directories. For this example, the subdirectories are named `work`.
2. Copy the unit control information definition file into the `$PDDIR/conf/work` directory. Copy the other HiRDB system definition files into the `$PDCONFPATH/work` directory.
3. Modify the HiRDB system definitions that you copied into the `$PDDIR/conf/work` and `$PDCONFPATH/work` directories.
4. Use the `pdconfchk -d work` command to check the HiRDB system definitions in the `$PDDIR/conf/work` and `$PDCONFPATH/work` directories. If errors are detected, correct each erroneous HiRDB system definition, and then execute the `pdconfchk` command again.
5. Use the `pdstop` command to terminate HiRDB normally.
6. Use the `pdlogunld` command to unload any system log files that are in unload wait status.
7. Copy any HiRDB system definitions files that you modified in step 3 into the `$PDDIR/conf` or `$PDCONFPATH` directory, replacing the current HiRDB system definition files.
8. If you changed the value specified in either of the following operands, use the `pdloginit` command to initialize the system log files:
   - `pd_log_dual`
   - `pdstart`
9. Use the `pdstart` command to start HiRDB normally.

(2) Notes

1. In the case of a HiRDB/Parallel Server, you must create subdirectories in the `$PDDIR/conf` and `$PDCONFPATH` directories in each unit and check the HiRDB system definitions in all of the units.
2. HiRDB system definitions that are being used by HiRDB must not be modified during operation. If modifications or deletions are made in such a situation, operation of the HiRDB cannot be guaranteed.
3. When HiRDB is terminated with planned, forced, or abnormal termination, some of the HiRDB system definition operands cannot be modified; for the operands that cannot be modified, see the manual *HiRDB Version 8 System Definition*.
4. After you have modified the HiRDB system definitions, back up the files in the `$PDDIR/conf` directory. To be prepared for the possibility of an error on the disk storing the HiRDB directory, you must back up the files in the HiRDB directory (files in `$PDDIR/conf`). You will need such a backup to recover the HiRDB directory if an error does occur. If the `$PDCONFPATH` directory is also in the HiRDB directory, back it up as well.
5. If you use the system reconfiguration command (`pdchgconf` command), you do not need to terminate HiRDB normally, because the `pdchgconf` command enables you to change HiRDB system definitions while HiRDB is running. Note that HiRDB Advanced High Availability is required in order to use this command. For details about changing HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.
6. If you are using the standby-less system switchover facility, do not modify HiRDB system definitions in the normal BES unit until you have used the `pdstop -u` command to terminate normally both the normal BES unit and the alternate BES unit. After you have modified HiRDB system definitions, copy the unit control information definition file and the back-end server definition files from the normal BES unit to the alternate BES unit. For details, see *25.5.3(2) Standby-less system switchover (1:1) facility*.

(3) Notes on the HiRDB/Parallel Server

1. When a system common definition is modified, the same changes must be made in the system common definition at every server machine.
2. If any unit terminates abnormally during normal or planned termination processing, no HiRDB system definition modifications should be made before the next startup. If modifications are made, HiRDB startup will probably fail (if HiRDB does start, it will not operate correctly).

(4) When linked to HiRDB Datareplicator

Before any of the following operands are added, modified, or deleted, HiRDB

Datareplicator must be terminated:

- `pd_log_dual`
- `pd_log_max_data_size`
- `pdlogadfg -d sys`
- `pdlogadpf -d sys`

HiRDB Datareplicator can be started again after the changes have been made. If these operands are added, modified, or deleted while HiRDB Datareplicator is running, data extraction by HiRDB Datareplicator may fail.

9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)

Executor: HiRDB administrator

This section explains how to modify HiRDB system definitions with the system reconfiguration command (`pdchgconf` command). The system reconfiguration command enables you to modify HiRDB system definitions while HiRDB is running, which eliminates the need to terminate HiRDB normally. To use this command, however, HiRDB Advanced High Availability must be installed.

9.2.1 Modification procedure

The following procedures show how to modify HiRDB system definitions with the system reconfiguration command.

Procedure

1. Create a `$PDDIR/conf/chgconf` directory.
2. Copy the current HiRDB system definition files into the directory that you created in step 1.
3. Modify the HiRDB system definitions in the `$PDDIR/conf/chgconf` directory.
4. Use the `pdconfchk -d` command to check the HiRDB system definitions in the `$PDDIR/conf/chgconf` directory. If errors are detected, correct each erroneous HiRDB system definition, and then execute the `pdconfchk` command again.
5. Use the `pdchgconf` command to replace the current HiRDB system definitions with the HiRDB system definitions that you modified. When the `pdchgconf` command is entered, the HiRDB system definition files currently being used (the unchanged files) are moved into the `$PDDIR/conf/backconf` directory, and the modified HiRDB system definition files in the `$PDDIR/conf/chgconf` directory are copied into the `$PDDIR/conf` directory.

After you execute this command, we recommend that you check that the execution results are correct. For details about checking command execution results, see the manual *HiRDB Version 8 Command Reference*.

Notes

- Assign HiRDB administrator write and access privileges for the `$PDDIR/conf`, `$PDDIR/conf/chgconf`, and `$PDDIR/conf/backconf` directories, and for all files under these directories.

- If a transaction or a utility remains active continuously for 15 minutes after the pdchgconf command has been entered, the command terminates abnormally.

9.2.2 Notes on changing operand specification values

(1) Operands whose specification values cannot be changed

The specification values of the following operands cannot be changed:

- pd_system_id
- pd_master_file_name

If the values specified in these operands are changed, the database must be initialized. Consequently, the system reconfiguration command should not be used to change these HiRDB system definitions.

(2) Operands that cannot be deleted

The following operands cannot be changed:

- pdlogadfg -d sys and pdlogadpf -d sys (only operands associated with system log files that are current or overwrite disabled cannot be deleted)
- pdlogadfg -d spd and pdlogadpf -d spd (only operands associated with synchronization point dump files that are being written or are overwrite disabled cannot be deleted)
- pd_syssts_file_name_1 to 7 (only operands associated with status files for the primary unit cannot be deleted)
- pd_sts_file_name_1 to 7 (only operands associated with status files for the primary server cannot be deleted)

(3) Invalid operands

Specification of operands related to reduced activation has no effect.

(4) Operands whose specification values affect system parameters

Changing the values specified in some operands requires changing values specified in operating system parameters as well. Examples of such operands are pd_max_users and SHMMAX. For a change to an operating system parameter to take effect, the machine must be restarted. In such a case, we recommend that you do not use the system reconfiguration command to modify HiRDB system definitions. Instead, terminate HiRDB normally, and then make the changes.

(5) Note on changing the pd_rpl_init_start operand

You must pay special attention if you use the pdrplstart or pdrplstop command to change the execution status of the HiRDB Datareplicator data extraction facility. Whether or not you will be able to execute the system reconfiguration command may

depend on the value of the pd_rpl_init_start operand.

Table 9-1 shows the executability of the system reconfiguration command depending on the pd_rpl_init_start operand's value and the data extraction facility's status.

Table 9-1: Executability of the system reconfiguration command depending on the pd_rpl_init_start operand value and the data extraction facility status

Value of pd_rpl_init_start before change	Execution status of the HiRDB Datareplicator data extraction facility	Value of pd_rpl_init_start after change	Executability of the system reconfiguration command
Y	Running	Y	Yes
		N (default)	No
	Stopped (by means of the pdrplstop command)	Y	No
		N (default)	Yes
N (default)	Stopped	Y	No
		N (default)	Yes
	Running (by means of the pdrplstart command)	Y	Yes
		N (default)	No

Legend:

Yes: The pd_rpl_init_start operand can be changed.

No: The pd_rpl_init_start operand cannot be changed.

(6) Operands required by prerequisite products

To set up or remove optional program products or plug-ins, you must stop HiRDB. This means that you may not be able to use the system reconfiguration command to delete operands or change specification values required by prerequisite products, such as optional program products and plug-ins.

9.2.3 Notes on executing the system reconfiguration command

Note on system log files (important)

When you execute the system reconfiguration command, system log files are swapped. Therefore, before you execute the pdchgconf command, check that swappable target files are available. If you execute the pdchgconf command when no swappable target files are available, HiRDB terminates abnormally. If this happens, prepare a swappable target file, and restart HiRDB with the pdstart command.

Environments in which the system reconfiguration command cannot be executed

The system reconfiguration command cannot be executed when any one of the following conditions applies:

- HiRDB was started with the `pdstart -r` command.
- A unit or server is stopped (including in reduced activation).*
- An ongoing network communications failure has occurred between units.*
- An updatable online reorganization session is executing.
- The `pdrplstop` command is executing.

* This is applicable to a HiRDB/Parallel Server only.

Reference note:

When a recovery-unnecessary front-end server is used, the environments in which the system reconfiguration command cannot be executed differ. For details, see *9.2.5(3) Relationship with a recovery-unnecessary front-end server (HiRDB/Parallel Server only)*.

Restrictions and limitations on executing the system configuration command

- Do not execute any other command or utility while the system reconfiguration command is executing. If you do, an error message stating that HiRDB has stopped may be issued.
- Cursors cannot be held when the system reconfiguration command is executed. Therefore, do not execute the command while a UAP that uses a holdable cursor is executing.
- The `UNTIL DISCONNECT` lock specification cannot be retained when the system reconfiguration command is executed. Therefore, do not execute the command while a UAP that uses the `LOCK` statement of the `UNTIL DISCONNECT` specification is executing.
- Responses to UAPs are slower than normal while the system reconfiguration command is executing.
- If you execute the system reconfiguration command while using a HiRDB client of version 07-00 or earlier, the HiRDB server is disconnected from that HiRDB client. Therefore, if you are using HiRDB clients of version 07-00 or earlier, be careful to execute the system reconfiguration command only when doing so will not interfere with normal operations.

Notes on linkage with HiRDB Datareplicator

If you add or delete a unit or server while HiRDB is linked to HiRDB

Datareplicator, you must rebuild the extraction environment of HiRDB Datareplicator. Furthermore, if a HiRDB transaction adds, updates, or deletes the extracted database while the extraction environment of HiRDB Datareplicator is being rebuilt, a mismatch with the target database occurs. For this reason, do not execute the system reconfiguration command while the extraction environment of HiRDB Datareplicator is being rebuilt.

9.2.4 HiRDB status after the system reconfiguration command has executed

After the system reconfiguration command has executed, HiRDB is in the same status it would be when it restarts normally. This means that the following events occur:

1. System log files are swapped.
2. Message log files are swapped. To retain messages in the message log file, back up the message log file before you execute the system reconfiguration command.
3. Global buffers assigned with the database structure modification utility become invalid.
4. Global buffers modified dynamically by the `pdbufmod` command become invalid.
5. The time count of the cleanup interval for troubleshooting information specified with the `pd_spool_cleanup_interval` operand resets to `0`.
6. If `normal` or `force` (default) is specified in the `pd_spool_cleanup` operand, troubleshooting information that has already been output is deleted.
7. Any statistical information collection change made with the `pdstbegin` or `pdstend` command becomes invalid, and the values specified in the `pd_statistics` and `pdstbegin` operands take effect.
8. Any resident process count change made with the `pdchprc` command becomes invalid, and the specification in the HiRDB system definitions takes effect.
9. If the `pd_db_io_error_action=unitdown` specification becomes invalid due to recurrence of an I/O error after HiRDB (or a unit in the case of a HiRDB/Parallel Server) terminates abnormally, the `unitdown` specification takes effect.
10. Lists used for narrowed searches are lost.

9.2.5 Relationship with other facilities

(1) Relationship with HiRDB Datareplicator (HiRDB/Parallel Server only)

Because HiRDB Datareplicator definitions and other values are updated when a unit or back-end server configuration is modified, you must stop HiRDB Datareplicator before performing such a change. The following shows the procedure for modifying the configuration of a unit or back-end server.

Procedure

1. Use the `pdrplstop` command to disconnect HiRDB Datareplicator.
2. Use the HiRDB Datareplicator `hdestop` command to stop the data extraction facility.
3. If `Y` is specified in the `pd_rpl_init_start` operand, change it to `N`.
4. Change the configuration of the unit or back-end server, and execute the system reconfiguration command. For details about changing the configuration of a unit or back-end server, see *11. Modifying the System Configuration*.
5. Reconfigure the extraction environment of HiRDB Datareplicator.
6. Use the HiRDB Datareplicator `hdestart` command to start the data extraction facility.
7. Use the `pdrplstart` command to connect HiRDB Datareplicator.
8. If you changed `pd_rpl_init_start` to `N` in step 3, change `N` back to `Y`, and then execute the system reconfiguration command.

(2) Relationship with system switchover facilities

(a) Standby system switchover facility

Execute the system reconfiguration command only when the primary system is the same as the running system. In addition, use the system reconfiguration command only to change HiRDB system definitions on the primary (running) system. Change HiRDB system definitions on the secondary system by copying the HiRDB system definition files from the primary system to the secondary system. The following procedures show how to modify HiRDB system definitions when the standby system switchover facility is being used.

Procedure 1: When your cluster software is HA Monitor and the standby system switchover facility is in the server mode:

1. Execute the system reconfiguration command on the primary system. This causes the secondary system HiRDB to stop automatically.
2. After the system reconfiguration command processing finishes, copy the

HiRDB system definition files from the primary system to the secondary system.

3. On the secondary system, execute the `pdstart` command (`pdstart -q` for a HiRDB/Parallel Server) to start the secondary system HiRDB.

Procedure 2: When Hitachi HA Toolkit Extension is being used:

1. Stop the secondary system HiRDB.
2. Execute the system reconfiguration command on the primary system.
3. After the system reconfiguration command processing finishes, copy the HiRDB system definition files from the primary system to the secondary system.
4. On the secondary system, execute the `pdstart` command (`pdstart -q` for a HiRDB/Parallel Server) to start the secondary system HiRDB.

Procedure 3: When the standby switchover facility is in the monitor mode:

1. Execute the system reconfiguration command on the primary system.
2. After the system reconfiguration command processing finishes, copy the HiRDB system definition files from the primary system to the secondary system.

(b) Standby-less system switchover facility

The following procedures show how to modify HiRDB system definitions when the standby-less system switchover facility is being used. Note that the system reconfiguration command cannot be used when the system has alternated (while the alternate BES is the active system).

Procedure 1: When the cluster software is HA Monitor:

1. Copy the following HiRDB system definitions into the `$PDDIR/conf/chgconf` directory in the alternate BES unit:
   - Back-end server definitions for the normal BES
   - Unit control information definitions for the normal BES unit
2. Execute the system reconfiguration command.

Procedure 2: When Hitachi HA Toolkit Extension is being used:

1. Copy the following HiRDB system definitions into the `$PDDIR/conf/chgconf` directory in the alternate BES unit:
   - Back-end server definitions for the normal BES
   - Unit control information definitions for the normal BES unit
2. Use the `pdstop -q -c` command to release the wait status of the alternate

portion.

3. Execute the system reconfiguration command. Note that executing the system reconfiguration command automatically places the alternate portion whose wait status was released in step 2 back into wait status.

(c) Notes (for both standby and standby-less system switchover facilities)

Note 1 applies when the standby system switchover facility is being used, and note 2 applies when the standby-less switchover facility is being used. Note 3 applies for both facilities.

1. Because the HiRDB system definitions on the primary system do not match the definitions on the secondary system while the system reconfiguration command is executing, the system cannot be switched over by means of a planned system switchover or as a result of a server failure.
2. Because the back-end server definitions and the unit control information definitions on the normal BES unit do not match the equivalent definitions on the alternate BES unit while the system reconfiguration command is executing, the system cannot be switched by means of a planned system switchover or as a result of a server failure.
3. If all of the following conditions apply, set up a shell that does not reposition IP addresses before you execute the system reconfiguration command:
   - The system is configured as a HiRDB/Parallel Server.
   - The cluster software is HA monitor.
   - The system switchover facility is in the server mode.
   - The configuration is set up for inheriting IP addresses.

(3) Relationship with a recovery-unnecessary front-end server (HiRDB/Parallel Server only)

(a) Conditions under which the system reconfiguration command can be executed

When a recovery-unnecessary front-end server is used, the environments in which the system reconfiguration command can be executed differ. If any of the following conditions is satisfied, the system reconfiguration command cannot be executed:

- The `pdstart -r` command was used to start HiRDB.
- Some units or servers have been shut down (including reduced activation).*
- A communication error has occurred between units in the network.
- Updatable online reorganization is being executed.
- The `pdrplstop` command is executing.

- The pd_mode_conf operand is set to AUTO.
- The front-end server in a stopped unit ceases to be a recovery-unnecessary front-end server after a system definition change.

* Not including a recovery-unnecessary front-end server.

Hint:

The system reconfiguration command can be executed even if the recovery-unnecessary front-end server is not active. However, all units other than the recovery-unnecessary front-end server must be active. In addition, all servers in the active units must also be active.

(b) Note

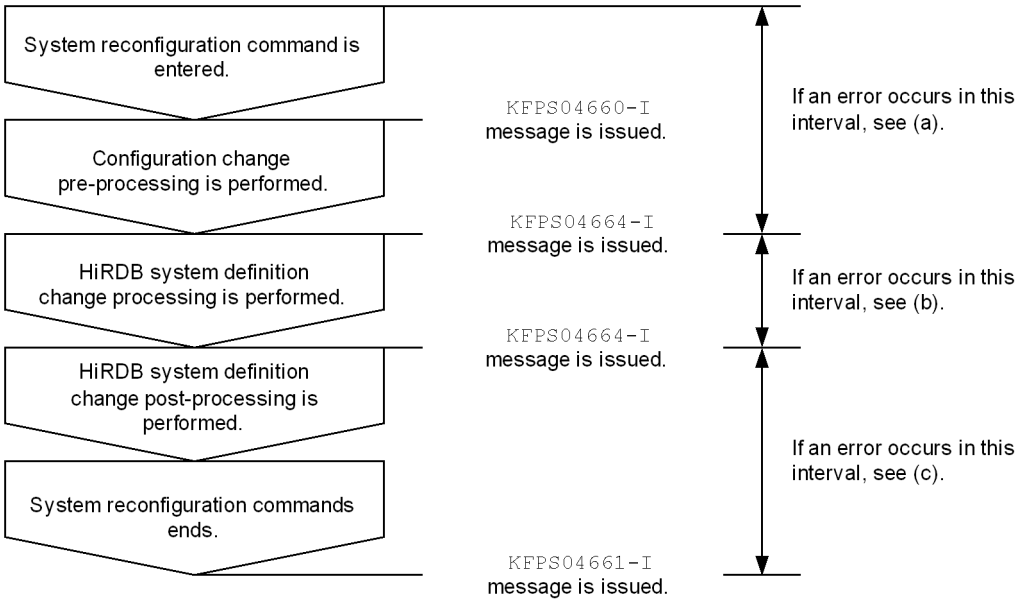
Executing the system reconfiguration command also reconfigures the HiRDB system definition of units that are stopped (recovery-unnecessary front-end server units). During this process, stopped units are started in order to modify their HiRDB system definition.

9.2.6 Actions to take when an error occurs

(1) If an error occurs when the system reconfiguration command is executed

Figure 9-1 shows the actions to take if an error occurs when the system reconfiguration is executed.

Figure 9-1: Action to take if an error occurs when the system reconfiguration command is executed



(a) If an error occurs during the configuration change pre-processing stage

If an error occurs during this interval, determine the cause of the error based on the message that is issued. In addition, use the `pdls` command to check if HiRDB has stopped. If it has stopped, use the `pdstart` command to start HiRDB. In this case, HiRDB starts using the unchanged HiRDB system definitions.

A message or an abort code may be issued accompanying a HiRDB termination processing failure. In this case, take action based on the message or abort code that is issued.

(b) If an error occurs during the HiRDB system definition change processing stage

If an error occurs during this interval, check the usage status of the disk containing the HiRDB directory; also check the access privileges to the HiRDB system definition files. In addition, use the `pdls` command to check if HiRDB has stopped. If it has stopped, copy the unchanged HiRDB system definition files in the `$PDDIR/conf/backconf` directory into the `$PDDIR/conf` directory and recover the system. Then use the `pdstart` command to start HiRDB.

If a unit or server has been moved, check its connection to the disk.

(c) If an error occurs during the HiRDB system definition change post-processing stage

If an error occurs during this interval, the system reconfiguration command automatically restores the unchanged HiRDB system definitions. Determine the cause of the error based on the message that is issued. Then check the HiRDB system definitions in the `$PDDIR/conf` directory.

In addition, use the `pdls` command to check that HiRDB is running. If it is not running, use the `pdstart` command to start HiRDB. To start HiRDB using the changed HiRDB system definitions, copy the HiRDB system definition files in the `$PDDIR/conf/chgconf` directory into `$PDDIR/conf`, and then start HiRDB.

A message or an abort code may be issued accompanying a HiRDB startup processing failure. If this happens, there may be a problem with the changed HiRDB system definitions or environment, so take the following actions:

- Use the `pdconfchk` command to check the changed HiRDB system definitions.
- Based on the message, secure the resources needed to run the modified system configuration.
- Note that the system reconfiguration command cannot be used if an operating system parameter needs to be changed.

If a unit or server has been moved, the system reconfiguration command may not be able to restore the unchanged HiRDB system definitions. Check the connections to the disk. To start HiRDB using the unchanged HiRDB system definitions, copy the HiRDB system definitions from the `$PDDIR/conf/backconf` directory. To start HiRDB using the changed HiRDB system definitions, copy the HiRDB system definitions from the `$PDDIR/conf/chgconf` directory. Then use the `pdstart` command to start HiRDB.

(2) If HiRDB cannot be started using the changed HiRDB system definitions

If HiRDB cannot be started using the changed HiRDB system definitions, the unchanged definitions are restored automatically, and another attempt is made to start HiRDB. If HiRDB does not start using the unchanged HiRDB system definitions either, take action based on the error messages that are issued.

(3) If the KFPS04665-W message is output

If the `KFPS04665-W` message is output, take action in accordance with the following procedure:

Procedure

1. Identify the unit indicated by the `KFPS04665-W` message.
2. Execute the `pdls -d svr` command to check the unit's status. Check the status in the execution results and determine whether `STOP(A)` is indicated

for any unit.

3. If there is a unit with `STOP(A)`, forcibly terminate that unit with the `pdstop -z` command.
4. Replace the HiRDB system definition files under `$PDDIR/conf` and `$PDCONFPATH` for the unit indicated in the `KFPS04665-W` message with the modified HiRDB system definition files.
5. Start the unit with the `pdstart -u` command.

9.3 Adding, modifying, and deleting global buffers while HiRDB is running (dynamic updating of global buffers)

Executor: HiRDB administrator

This section explains how to use the `pdbufmod` command to add, modify, and delete global buffers while HiRDB is running.

9.3.1 Overview of dynamic updating of global buffers

You use the `pdbufmod` command to add, modify, and delete global buffers while HiRDB is running. This is called *dynamic updating of global buffers*. For example, you can perform global updating of global buffers for the following purposes:

- To assign a global buffer to an RDAREA that has been added
- To modify a global buffer already assigned to an RDAREA
- To modify global buffer definitions based on the results of tuning

Note that HiRDB Advanced High Availability must be installed in order to perform dynamic updating of global buffers.

(1) Limitations on the validity of dynamic updating of global buffers

Global buffers that have been updated dynamically become invalid when HiRDB terminates normally or with a planned termination. Therefore, while HiRDB is stopped, you must use the `pdbuffer` operand to define any global buffers that were updated dynamically. Note also that when you use the system reconfiguration command (`pdchgconf` command), you can change values specified in the `pdbuffer` operand while HiRDB is running. For details about using the system reconfiguration command to change HiRDB system definitions, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

In addition, before terminating HiRDB normally or with a planned termination, you should save the execution results of the `pdbufls -k def` command (global buffer definition information). You can then refer to this information to update the `pdbuffer` operand.

(2) Before performing the dynamic update operation

Before updating global buffers dynamically, you must first complete the operations described in the following subsections.

(a) Specify HiRDB system definitions

1. Specify `Y` in the `pd_dbbuff_modify` operand.
2. If `fixed` is specified in the `pd_dbbuff_attribute` operand (default value in the 32-bit mode), the shared memory used by a global buffer that is added or

modified by this function is fixed in real memory. Therefore, consider carefully the size of the real memory before adding or modifying global buffers. If the size of the real memory is inadequate, specify `free` in the `pd_dbbuff_attribute` operand.

3. Re-estimate the value specified in the `SHMMAX` operand.

(b) Perform memory-related preparations

1. When this function is used, the space needed for the shared memory resources listed below increases. To accommodate this increase, you must re-estimate the shared memory size requirements. For details about estimating shared memory size requirements, see the manual *HiRDB Version 8 Installation and Design Guide*.
   - Shared memory used by global buffers
   - Shared memory used by a single server
   - Shared memory used by the dictionary server
   - Shared memory used by back-end servers
2. The space required by status files increases. To accommodate this increase, you must re-estimate the status file size requirements. For details about estimating status file size requirements, see the manual *HiRDB Version 8 Installation and Design Guide*.
3. Re-estimate the `shmmax`, `shmmni`, and `shmseg` operating system parameters (on Solaris, the `shminfo_shmmax`, `shminfo_shmmin`, and `shminfo_shmseg` parameters; on Linux, the `SHMMAX`, `SHMMIN`, and `SHMSEG` parameters). For details about estimating operating system parameters, see the manual *HiRDB Version 8 Installation and Design Guide*.

(3) Notes

(a) Relationship with other facilities

You cannot perform dynamic updating of global buffers while you are using the rapid system switchover facility or the standby-less system switchover facility. To use these facilities, execute the system reconfiguration command (`pdchgconf` command) to change the values specified in the `pdbuffer` operand so that global buffers are assigned. Use of the `pdchgconf` command enables you to modify HiRDB system definitions while HiRDB is running. However, HiRDB Advanced High Availability must be installed in order to use this command. For details about changing HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

(b) Exclusion control

While dynamic updating of global buffers is being performed (while the `pdbufmod`

command is executing), the associated RDAREAs are locked by placing them in exclusion mode (EX). This means that transactions that attempt to access these RDAREAs are placed in wait status.

(c) Maximum number of global buffers

To set the maximum number of global buffers, use the pd_max_add_dbbuff_no and pd_max_add_dbbuff_shm_no operands.

You can determine the number of global buffers by counting the number of global buffers displayed in the execution results of the pdbufls -k def command. For a HiRDB/Parallel Server, count the number of global buffers associated with the server name (header name: SVID).

(d) Maximum number of shared memory segments used for global buffers

When you update a global buffer dynamically (when a global buffer is added or when a global buffer modification increases memory requirements), a new shared memory segment is secured, and the dynamically updated global buffer is assigned to this shared memory segment. To set the shared memory segments, use the pd_max_add_dbbuff_no and pd_max_add_dbbuff_shm_no operands. For a HiRDB/Parallel Server, set the value for each server. You can determine the number of shared memory segments by counting the number of shared memory segments displayed in the execution results of the pdbufls -k mem command. For a HiRDB/Parallel Server, count the number of shared memory segments associated with the server name (header name: SHM-OWNER).

If the number of shared memory segments exceeds the maximum, update the pdbuffer operand with the definition information for the dynamically updated global buffers. Then, starting HiRDB normally will reduce the number of shared memory segments.

Note that if the size of the shared memory used by a dynamically updated global buffer is greater than the value in the SHMMAX operand, the shared memory that is being used by the global buffer will consist of more than one shared memory segment. You can use the following formula to obtain the number of shared memory segments that have been allocated:

↑ *size-of-shared-memory-used-by-the-dynamically-updated-global-buffer* ÷ *value-in-SHMMAX-operand* ↑

For details about shared memory used by global buffers, see the manual *HiRDB Version 8 Installation and Design Guide*.

9.3.2 Application examples

This section provides application examples illustrating how to use the pdbufmod command to add, modify, and delete global buffers.

(1) Example 1: Adding a global buffer

In this example, a global buffer (gbuf01) is added, and an RDAREA (RDAREA1) added for that global buffer is assigned.

```
pdbufmod -k add -a gbuf01 -r RDAREA1 -n 1000
```

Explanation

-k add: Specifies addition of a global buffer.

-a: Specifies a name for the global buffer being added.

-r: Specifies an RDAREA to assign.

-n: Specifies the number of buffer sectors for the global buffer.

(2) Example 2: Adding a global buffer and changing an RDAREA's global buffer assignment

In this example, a global buffer (gbuf02) is added. At the same time, the global buffer to which an RDAREA (RDAREA1) is assigned (the RDAREA global buffer assignment) is changed from gbuf01 to gbuf02.

```
pdbufmod -k add -a gbuf02 -r RDAREA1 -n 1000
```

Explanation

-k add: Specifies addition of a global buffer.

-a: Specifies a name for the global buffer to be added.

-r: Specifies an RDAREA to assign.

-n: Specifies the number of buffer sectors for the global buffer.

(3) Example 3: Setting an RDAREA global buffer assignment

In this example, an added RDAREA (RDAREA1) is assigned to an existing global buffer (gbuf01).

```
pdbufmod -k add -a gbuf01 -r RDAREA1
```

Explanation

-k add: Specifies assignment of an RDAREA to an existing global buffer.

-a: Specifies the name of the target global buffer.

-r: Specifies the RDAREA to be assigned.

(4) Example 4: Changing an RDAREA global buffer assignment

In this example, the global buffer assignment of an RDAREA (RDAREA1) is changed from gbuf01 to gbuf02. Both gbuf01 and gbuf02 are existing global buffers.

```
pdbufmod -k add -a gbuf02 -r RDAREA1
```

Explanation

-k add: Specifies assignment of an RDAREA to an existing global buffer.

-a: Specifies the name of the target global buffer.

-r: Specifies the RDAREA to be assigned.

(5) Example 5: Deleting a global buffer

In this example, a global buffer (gbuf01) to which no RDAREA is assigned is deleted.

```
pdbufmod -k del -a gbuf01
```

Explanation

-k del: Specifies deletion of a global buffer.

-a: Specifies the name of the global buffer to be deleted.

(6) Example 6: Deleting a global buffer to which an RDAREA is assigned

In this example, a global buffer (gbuf01) to which an RDAREA (RDAREA1) is assigned is deleted.

```
pdhold -r RDAREA1 -c                          1
pdbufmod -k del -a gbuf01                     2
```

Explanation

1. The RDAREA is placed in shutdown status. Note, however, that this operation is not needed if gbuf01 is an index global buffer.
2. The global buffer is deleted.

(7) Example 7: Separating an RDAREA from a global buffer

In this example, an RDAREA (RDAREA1) is separated from a global buffer (gbuf01).

```
pdhold -r RDAREA1 -c                              1
pdbufmod -k del -r RDAREA1                        2
```

Explanation

1. The RDAREA is placed in shutdown status.
2. The RDAREA is separated from the global buffer.

(8) Example 8: Changing a global buffer definition

In this example, the number of buffer sectors comprising a global buffer (`gbuf01`) is changed from 1,000 to 2,000.

```
pdbufmod -k upd -a gbuf01 -n 2000
```

Explanation

`-k upd`: Specifies a change in a global buffer definition.

`-a`: Specifies the name of the global buffer whose definition is to be changed.

`-n`: Specifies the number of buffer sectors after the change.

The `pdbufmod` command can be used to change the following items:

- Number of buffer sectors
- Buffer size
- Maximum number of concurrently executing prefetch processes
- Maximum number of concurrent input pages
- Updated input page rate during deferred write trigger

9.4 Changing the number of server processes

This section explains the procedures for changing the number of server processes. The following topics are covered:

- Operands for specifying the number of server processes
- Procedures for changing the number of server processes
- Examples of operation when the number of server processes is changed

(1) Operands for specifying the number of server processes

(a) Operands for specifying the maximum number of active processes

The maximum number of server processes that can be started by HiRDB (maximum number of active processes) is determined by the HiRDB system definition operands shown in Table 9-2.

Table 9-2: Operands for specifying the maximum number of active processes

Type of server	Definition type	Operand
Single server	System common definition	`pd_max_users`
Front-end server	System common definition	`pd_max_users`
Dictionary server	Dictionary server definition or server common definition	`pd_max_dic_process`[1]
Back-end server	Back-end server definition or server common definition	`pd_max_bes_process`[2]

Notes

1. If there are multiple front-end servers, processing requests are sent from those servers to the dictionary server and back-end servers. This may result in a concentration of the number of processes that exceeds the value specifiable in the `pd_max_users` operand in the system common definition. The resulting concentration of processes is the product of the value specified in the `pd_max_users` operand × number of front-end servers). Therefore, when multiple front-end servers are used, estimate a maximum number of processes appropriate to the concentration level of processing must be determined, and then the `pd_max_dic_process` and `pd_max_bes_process` operands can be specified.
2. Even when only one front-end server is used, if more operations related to RDAREAs or global buffers (`pdbufls`, `pddbls`, `pdopen`, `pdclose`, `pdhold`, and `pdrels`) are executed concurrently than the value of

pd_max_users, processing exceeding the value of pd_max_users may be concentrated on the dictionary server. In this case also, the maximum number of processes appropriate to the concentration level of processing should be specified.

[1] When the pd_max_dic_process operand is omitted, the value of the pd_max_users operand in the system common definition is assumed.

[2] When the pd_max_bes_process operand is omitted, the value of the pd_max_users operand in the system common definition is assumed.

(b) Operand for specifying the number of resident processes

The number of server processes generated during HiRDB startup (that is, the number of resident processes) is determined by the value of the pd_process_count operand.

If there will be many processing requests in the system, setting the number of resident processes to a large value will enable all processing to be started immediately. If the number of resident processes is set to a small value and more processing requests are issued than there are resident processes, HiRDB will process the requests by starting as many server processes as possible up to the specified maximum number of active processes. In this case, processing cannot be started until these processes have started. However, when the number of resident processes is set to a larger value, more memory is required in order to keep server processes always active. Therefore, the number of resident processes should be set to an appropriate value taking into account the concentration level of processing and the required memory size.

(2) Procedures for changing the number of server processes

The following two methods are available for changing the number of server processes:

- Changing with the pdchprc command
- Changing with HiRDB system definitions

(a) Changing with the pdchprc command

You can use the pdchprc command to change the maximum number of active processes and the maximum number of resident processes. However, changing the number of processes with the pdchprc command is temporary; such a change remains in effect only until HiRDB is stopped or until the pdchprc command is executed again.

(b) Changing with HiRDB system definitions

You can also change the number of server processes by directly changing the values in the operands listed in Table 9-2 and in the pd_process_count operand. Note, however, that before you can change a value in an operand listed in Table 9-2, you must either terminate HiRDB normally or perform a planned termination. If you change a value in an operand shown in Table 9-2 after HiRDB has been terminated forcibly or

after HiRDB has terminated abnormally, an error will occur when you attempt to restart HiRDB and HiRDB will not restart. This does not apply to changing only the value in the `pd_process_count` operand.

Use of the system reconfiguration command (`pdchgconf` command) enables you to modify HiRDB system definitions while HiRDB is running. Note that HiRDB Advanced High Availability must be installed in order to use this command. For details about changing HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

(3) Examples of operation when the number of server processes is changed

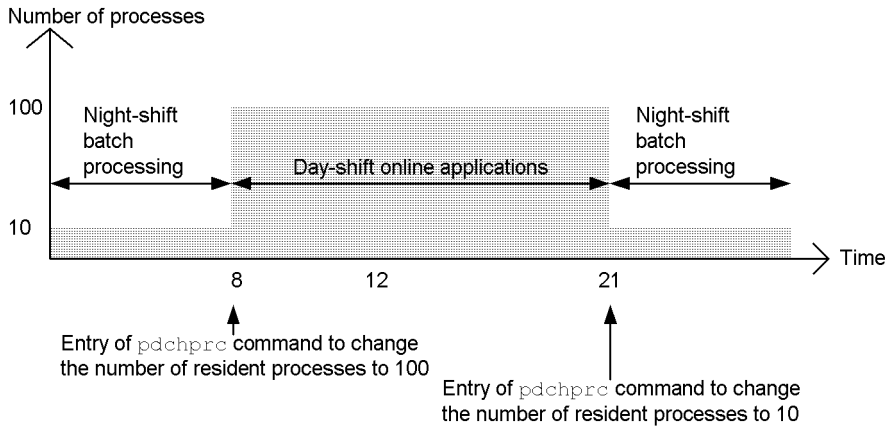
The `pdchprc` command can be used during HiRDB operation to change temporarily the maximum number of active processes and the number of resident processes. Examples are provided below.

(a) Reducing the number of server processes for batch processing

After online applications have terminated, this example reduces temporarily the number of resident server processes during night-shift batch processing. Processing requests are issued concurrently during online application processing, but because there should be no such concurrent processing requests during night-shift batch processing, there is no need to keep many server processes active. It should be sufficient to activate only as many server processes as are required by night-shift batch processing.

Figure 9-2 shows the procedure for reducing the number of server processes for batch processing.

Figure 9-2: Reducing the number of server processes for batch processing



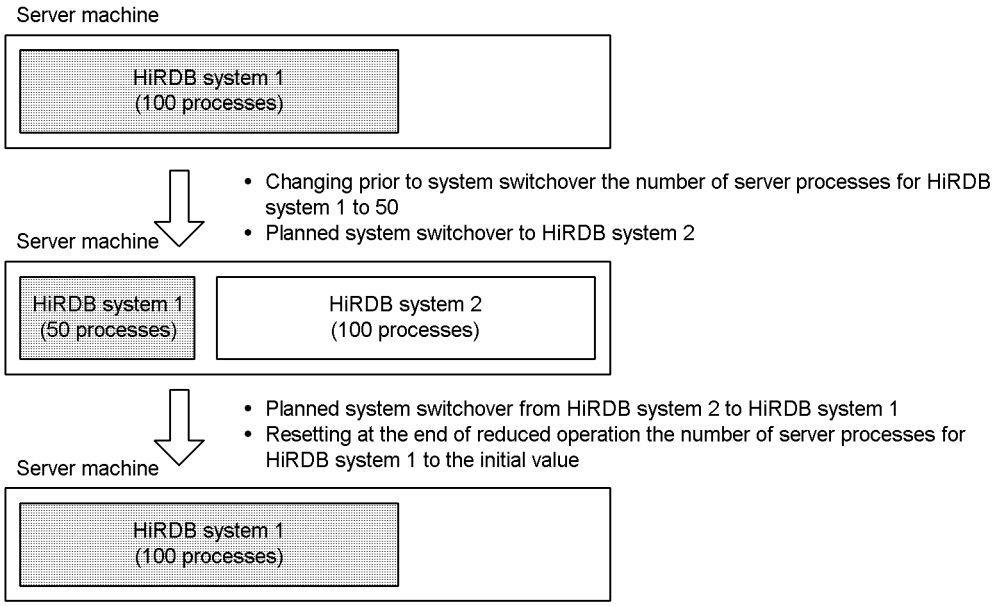
(b) Reduced system operation in a mutual system switchover environment

In a mutual system switchover environment, it is possible after system switchover

occurs for multiple HiRDBs to be running on a single server machine, resulting in a temporary increase in memory usage. An increase in memory usage of this type can be minimized by using the `pdchprc` command to reduce the number of HiRDB server processes.

Figure 9-3 shows the procedure for achieving reduced system operation in a mutual system switching environment.

Figure 9-3: Reduced system operation in a mutual system switching environment



(4) Notes (applicable to HiRDB/Parallel Server only)

Problems may result if the `pdchprc` command is used to set the maximum number of server processes to 0. For example, when the maximum number of server processes is set to 0 for a front-end server that is located in the same unit as the system manager, the following utilities can no longer be executed:

- Database structure modification utility
- Database load utility
- Database reorganization utility
- Database condition analysis utility
- Optimizing information collection utility

9.5 Handling an increase in the number of users

When the number of users increases, the values of the following operands should be increased:

- pd_max_users
- pd_max_server_process

We recommend you keep this point in mind during the discussion that follows. If you omit the pd_max_server_process operand, HiRDB automatically calculates a value for this operand.

(1) Shared memory for HiRDB increases

The size of the shared memory used by HiRDB increases, which may result in shared memory allocation errors. In such a case, the shmget() system call error is reported with the KFPO00113-E message. Appropriate action must be taken on the basis of this message.

- If errno=22, increase the value of the shmmax, shminfo_shmmax, or SHMMAX OS system parameter.
- If errno=12, add more memory or terminate another program.

For details on how to modify the OS system parameters, see the applicable OS manual. When a system parameter is modified, the new value does not take effect until the OS is rebooted.

For information on determining values for shared memory and semaphores, see the manual *HiRDB Version 8 Installation and Design Guide*.

(2) More HiRDB ports are used

As the number of users increases, the number of ports used by HiRDB also increases. If the number of ports becomes inadequate, processing may be interrupted or the communications processing of other programs may be affected adversely. The number of ports assigned automatically by the operating system depends of the OS, so you should check the network-related settings on each server machine. For details about the number of ports that HiRDB uses, see the manual *HiRDB Version 8 Installation and Design Guide*.

If the number of ports assigned automatically by the OS is inadequate, use the pd_registered_port operand to specify a range of port numbers to be used by HiRDB. For details, see *9.7 Specifying a range of port numbers for use in communication processing*.

In addition, if the maximum number of concurrent connections is too large (if the value in the pd_max_users operand is too large), specify 1 in the PDTCPCONOPT operand

in the client environment definition to reduce the number of ports used by UAPs when they connect to HiRDB. For details about the PDTCPCONOPT operand, see the manual *HiRDB Version 8 UAP Development Guide*.

(3) Fewer locked resources can be allocated within a unit

The number of locked resources that can be allocated within a unit is reduced, which may result in output of the KFPS00443-E message indicating a lock error. In such a case, the value of the pd_lck_pool_size operand should be increased in order to allocate a sufficient locked area.

(4) More semaphores and semaphore identifiers are used (applicable to HiRDB/Single Server only)

In the case of a HiRDB/Single Server, more semaphores and semaphore identifiers are used, which means that the semget() system call error is reported with the KFPS01815-E or KFPO00107-E message. Appropriate action must be taken on the basis of the message:

- errno=22

  Increase the value of the semmns, seminfo_semmns, or SEMMNS OS system parameter.

- errno=28

  Increase the value of semmni, seminfo_semmni, or SEMMNI OS system parameter.

In the case of a HiRDB/Parallel Server, an increased number of users has no effect on the number of semaphores or semaphore identifiers.

(5) Synchronization point dump file and status file sizes increase

The synchronization point dump file and status file sizes increase. For information on determining the sizes of synchronization point dump files and status files, see the manual *HiRDB Version 8 Installation and Design Guide*.

(6) Exceeding the number of users authorized at the time of the HiRDB purchase

If the number of users that were authorized at the time HiRDB was purchased is exceeded, HiRDB must be upgraded.

9.6 Accommodating clients that cannot connect to HiRDB (connection frame guarantee facility for client groups)

When the number of clients connected to HiRDB equals the value specified in the `pd_max_users` operand (maximum number of concurrent connections), new clients cannot be connected to HiRDB — even clients who need to execute important applications. In such a case, the *connection frame guarantee facility for client groups* should be used.

(1) Connection frame guarantee facility for client groups

The clients who connect to HiRDB can be assigned to groups, and HiRDB connection frames are guaranteed for each client group. The following are the HiRDB-set client groups (the character strings in parentheses are the names of these client groups):

X/Open XA interface client group (XA)

This is a group for clients who use the X/Open XA interface to access HiRDB. Whether from a PC or a work station, if a client accesses HiRDB via the X/Open XA interface, the client belongs to the X/Open XA interface client group.

Distributed client group (DF)

This is a group for clients who use the distributed database facility to access HiRDB from other nodes. Whether from a PC or a work station, if a client accesses HiRDB using the distributed database facility, the client belongs to the distributed client group.

PC client group (PC)

This is a group for Windows and Linux clients.

WS client group (WS)

This is a group for UNIX clients.

Mainframe client group (MF)

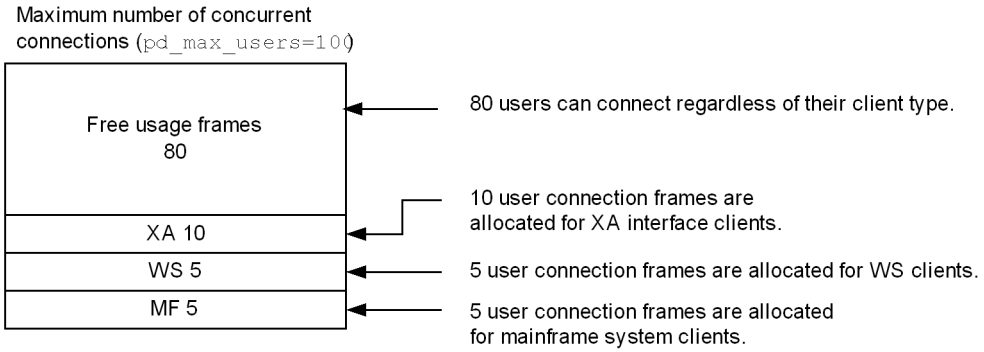
This is a group for VOS3 clients.

Note

The character strings in parentheses designate the client group name.

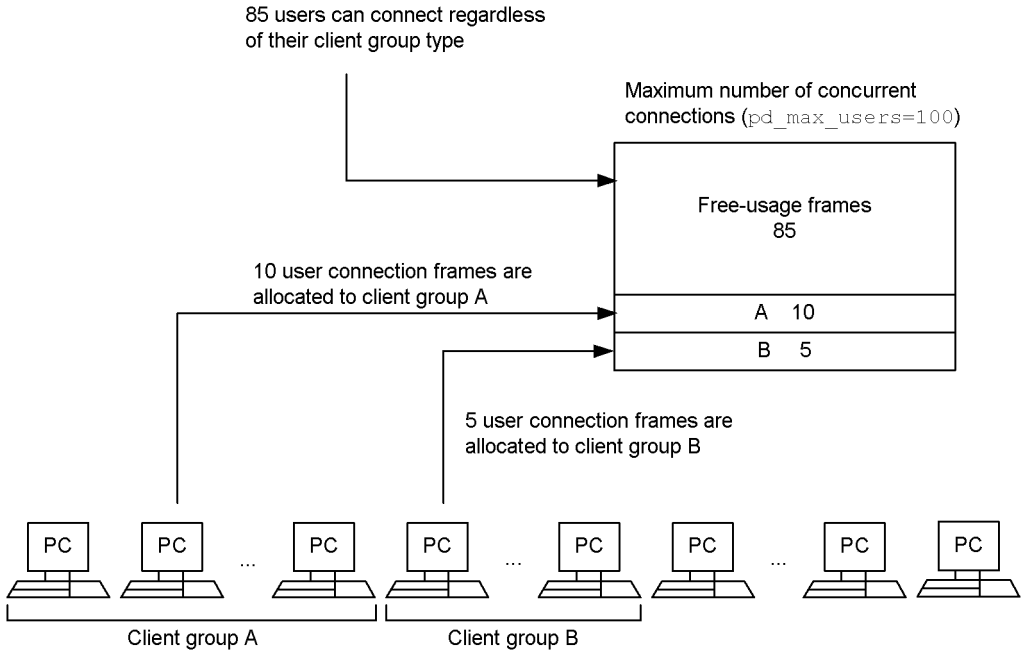
Any client who belongs to a client group for which connection frames have been specified can establish connection to HiRDB until the guaranteed minimum connections count is reached, even though accesses to HiRDB are concentrated among clients belonging to other client groups. Figure 9-4 shows the connection frame guarantee facility for client groups.

Figure 9-4: Connection frame guarantee facility for client groups



(2) User can define a desired client group

The client groups listed above are defined by HiRDB. The user can also define up to 10 client groups:



(3) Environment setup

The following operands must be specified in order to use the connection frame guarantee facility for client groups:

- pdcltgrp operand in the system common definition
- PDCLTGRP operand in the client environment definition*

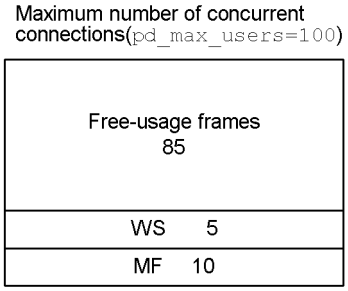
* This operand is used to define a user-specified client group.

The environment setup procedure is explained below by way of examples.

Example 1

Allocate 5 user connection frames for work station clients and 10 user connection frames for mainframe system clients.

Assume that pd_max_users=100 has been specified.

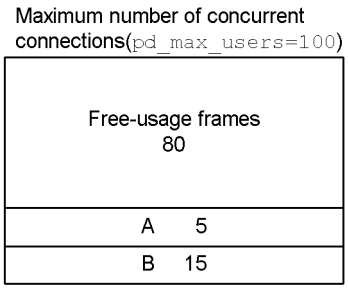


Specifications in the system common definition

```
pdcltgrp -g WS -u 5
pdcltgrp -g MF -u 10
```

Example 2

Set user-specified client groups, and allocate 5 user connection frames to group A and 15 user connection frames to groups B; assume that pd_max_users=100 has been specified:



Specifications in the system common definition

```
pdcltgrp -g A -u 5
pdcltgrp -g B -u 15
```

Client environment definition for group A

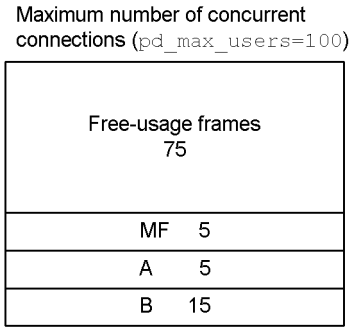
```
PDCLTGRP = A
```

Client environment definition for group B

```
PDCLTGRP = B
```

Example 3

Allocate 5 user connection frames to mainframe clients. Also set user-specified client groups and allocate 5 user connection frames to group A and 15 user connection frames to group B; assume that `pd_max_users=100` has been specified:



Specifications in the system common definition

```
pdcltgrp -g MF -u 5
pdcltgrp -g A -u 5
pdcltgrp -g B -u 15
```

Client environment definition for group A

```
PDCLTGRP = A
```

Client environment definition for group B

```
PDCLTGRP = B
```

Note

If the values in the pdcltgrp operand and the PDCLTGRP operand do not match, the user-specified client group will be ignored.

(4) Notes

1. The number of free-usage frames should not be set too small. If there are too few such frames, a utility or UAP connection error may occur due to an excessive number of connected users.
2. Check that the sum of the guaranteed numbers of connected users specified with the -u option of the pdcltgrp operand does not exceed the value of the pd_max_users operand. If the value of the pd_max_users operand is exceeded, processing of HiRDB startups is discontinued.
3. A utility (except for the database definition utility) cannot belong to a client group.
4. A mainframe client cannot belong to a user-specified client group.
5. To specify an X/Open XA interface client group (XA), the HiRDB client's version must be one of the following:
   - 04-05 (neither 05-00 nor 05-01 supports this specification)
   - 05-02 or later
6. To specify a PC client group (PC) or work station client group (WS), the HiRDB client's version must be the following:
   - 04-00 or later

(5) Relationship to monitoring the resources utilization factor

When the pd_max_users utilization factor is being monitored,* both the pd_max_users operand utilization factor and the free-usage frames utilization factor are monitored. There must be at least 10 free-usage frames.

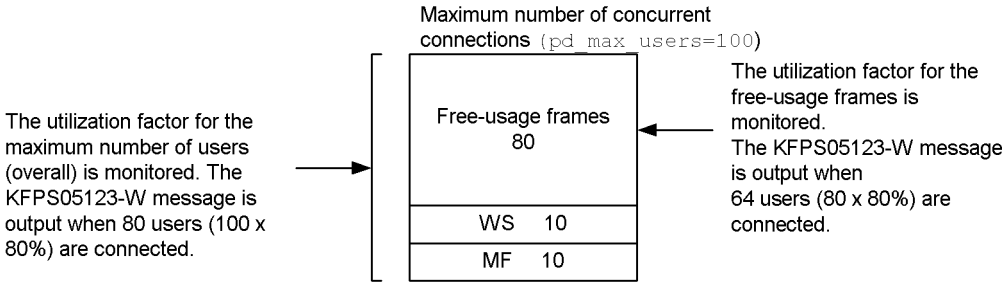
* Applicable when either of the following operands is specified:

- pd_watch_resource = AUTO
- pd_max_users_wrn_pnt

The following example shows when error messages are to be output when the pd_max_users utilization factor is being monitored.

Example

This example specifies that the message is to be output when the pd_max_users utilization factor reaches 80%:



9.7 Specifying a range of port numbers for use in communication processing

The ports to be used for communication between a HiRDB server and a HiRDB client or between HiRDB servers are assigned automatically by the OS. If a large volume of communication occurs, a shortage of ports may interrupt communication processing or may have an adverse effect on the communication processing of other programs. To prevent such problems from occurring, it is possible to specify a range of port numbers to be used by HiRDB for communication processing.

It is not necessary to specify this facility if only a small number of ports will be used for communication processing. This facility is applicable to server-to-server communication, and will not necessarily be applicable to all processes under HiRDB. The port numbers assigned by the OS will be used for communication processing such as the processing of commands.

For the Linux versions of HiRDB, consider whether or not to specify a range of port numbers. For other OS versions of HiRDB, you do not need to specify a range of port numbers.

(1) Number of ports used by HiRDB

For details about the number of ports used by HiRDB, see the manual *HiRDB Version 8 Installation and Design Guide*.

(2) Specifying a port number range

Use the `pd_registered_port` operand to specify a range of port numbers.

(3) Notes

It is important to ensure that the port number specified in the `pd_registered_port` operand do not duplicate any of the following port numbers (if there is a duplication in port numbers, an error may occur and it will not be possible to start HiRDB):

1. Port number specified in the `pd_name_port` operand
2. Port number specified in the `pd_service_port` operand
3. Port number specified in the `-p` option of the `pdunit` operand
4. Port numbers registered in `/etc/services`
5. Port number specified in the `PDCLTRCVPORT` operand in the client environment definition (when the HiRDB client resides in the same server machine as the HiRDB server)
6. Port number being used by another program

If `Y`, `C`, or `W` is specified in the `pd_registered_port_check` operand, HiRDB

checks the port numbers in 4 for any duplications. If `N` is specified in the `pd_registered_port_check` operand, HiRDB will not check port numbers, and the HiRDB administrator must make the check for duplicated port numbers.

Hint:

- In a multi-HiRDB environment, ensure that the range of reserved port numbers for each HiRDB (the range specified in the `pd_registered_port` operand) does not duplicate any other range of port numbers.
- If the system switchover facility is used and multiple units exist on a single server machine, specify the `pd_registered_port` operand of the unit control information definition instead of the system common definition. Ensure that the range of reserved port numbers for each unit does not duplicate any other range of port numbers.pd_registered_port_level.
- You can use operands to specify the target range for the HiRDB reserved port facility. For details, see the manual *HiRDB Version 8 System Definition*.

9.8 Changing the host name

Executor: HiRDB administrator

This section explains the procedure for changing the host name (changing the computer name) of HiRDB while it is active.

Procedure

1. Use the `pdstop` command to terminate HiRDB normally.
2. Change the `pd_mode_conf` operand.
3. Use the `pdlogunld` command to unload system logs.
4. Change the host name.
5. Restart the server machine.
6. Change HiRDB system definitions.
7. Initialize the system files.
8. Change the `pd_mode_conf` operand.
9. Use the `pdstart` command to start HiRDB normally.

The procedure step numbers above correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdstop command to terminate HiRDB normally

```
pdstop
```

(2) Correct the pd_mode_conf operand

If `AUTO` was specified in the `pd_mode_conf` operand, change the specification to `MANUAL2`.

(3) Use the pdlogunld command to unload system logs

Unload the system log files waiting to be unloaded. The status of system log files can be checked with the `pdlogls` command:

```
pdlogunld -d sys -g log01 -o /unld/unldlog01
```

(4) Rename the host

Rename the host (change the computer name).

(5) Reboot the server machine

Reboot the server machine.

In the case of an HiRDB/Parallel Server, reboot all server machines whose host has been renamed.

(6) Change HiRDB system definitions

Change the host name specified in the following operands in the HiRDB system definition:

- `-x` option in `pdunit` operand
- `-x` option in `pdstart` operand
- `pd_hostname` operand

(7) Initialize the system files

Initialize the system log files, synchronization point dump files, and status files (delete them, then create them again). In the case of a HiRDB/Parallel Server, initialize all system files in the unit where the host name was changed.

```
pdlogrm -d sys -f /sysfile01/syslog1a                    1
pdloginit -d sys -f /sysfile01/syslog1a -n 5000          2
pdlogrm -d spd -f /sysfile01/sync1                       3
pdloginit -d spd -f /sysfile01/sync1 -n 5000             4
pdstsrm  -f /sysfile01/usts1a                            5
pdstsinit -f /sysfile01/usts1a -c 5000                   6
pdstsrm  -f /sysfile01/ssts1a                            7
pdstsinit -f /sysfile01/ssts1a -c 5000                   8
```

Explanation

1. Deletes system log files.
2. Re-creates system log files.
3. Deletes synchronization point dump file.
4. Re-creates synchronization point dump files.
5. Deletes unit status files.
6. Re-creates unit status files.
7. Deletes server status files.
8. Re-creates server status files.

(8) Change the pd_mode_conf operand

If the specification of the `pd_mode_conf` operand was changed to `MANUAL2` in step (2), change it back to `AUTO`.

(9) Use the pdstart command to start HiRDB normally

```
pdstart
```

9.9 Changing the deadlock priority value for commands

This section explains how to change the deadlock priority value for the following commands:

- `pdhold -b` (referencing-possible backup hold)
- `pdhold -s` (synchronization hold)
- `pddbchg` (replica status switching of a replica RDAREA)
- `pdorbegin` (database commit after online reorganization)
- `pdorend` (reflection processing after online reorganization)

9.9.1 Deadlock priority value for commands

The deadlock priority value for commands is normally lower than the deadlock priority value for a UAP. Therefore, if deadlock occurs, it is the command that terminates in an error. To prevent commands from terminating in an error, you can set the deadlock priority value for commands to be higher than the deadlock priority value for a UAP. When you increase the command deadlock priority value, the error occurs in the transaction when deadlock occurs.

(1) Commands for which the deadlock priority value can be changed

You can change the deadlock priority value for the following commands:

- `pdhold -b` (referencing-possible backup hold)
- `pdhold -s` (synchronization hold)
- `pddbchg` (replica status switching of a replica RDAREA)
- `pdorbegin` (database commit after online reorganization)
- `pdorend` (reflection processing after online reorganization)

(2) List of deadlock priority values

Table 9-3 lists the deadlock priority values.

Table 9-3: Deadlock priority values

Type and condition				Deadlock priority value
UAP	Value of `PDDLKPRIO` in client environment definition	96		96
		64		64
		32		32
		Omitted	When X/Open XA interface is used	96
			When X/Open XA interface is not used and a distributed server is used for the distributed databases	64
Utility				64
Command	`pdhold -b`, `pdhold -s`, `pddbchg`, `pdorbegin`, `pdorend`			Value of `pd_command_deadlock_priority`*
	Other commands			64

* You can specify in this operand a deadlock priority value of 32, 64, 96, or 120. If this operand is omitted, the deadlock priority value is either 64 or 120 depending on the type of locked resource.

Reference note:

- A smaller deadlock priority value indicates a higher processing priority, and a larger deadlock priority value indicates a lower processing priority.
- If two transactions have the same deadlock priority value, the first one that executes is accorded the higher priority.

9.9.2 Environment assignment

You specify the following operands to change the deadlock priority value for commands:

- `pd_deadlock_priority_use`

  Specifies whether or not the deadlock priority value for commands is to be changed. To change the deadlock priority value, specify `Y`.

- `pd_command_deadlock_priority`

Specifies a deadlock priority value for commands. The values that can be specified are 32, 64, 96, and 120.

9.9.3 Operation method

(1) Procedure for changing the deadlock priority value

The following describes the procedure for changing the deadlock priority value:

Procedure

1. Check the deadlock priority value (value of the `PDDLKPRIO` operand in the client environment definition) of the transaction that may cause deadlock.
2. Specify `Y` in the `pd_deadlock_priority_use` operand.
3. Specify a deadlock priority value for commands in the `pd_command_deadlock_priority` operand.

Hint:

The following are specification guidelines for the `pd_command_deadlock_priority` operand:

- To cause an error in the transaction when deadlock occurs, ensure that the operand specification values satisfy the following condition:

  value of pd_command_deadlock_priority < value of PDDLKPRIO

- To cause an error in the command when deadlock occurs, ensure that the operand specification values satisfy the following condition:

  value of pd_command_deadlock_priority > value of PDDLKPRIO

(2) Operand specification example

An example of specifying operands is provided for the following outcome:

- When deadlock occurs between a transaction of UAP1 and a command, an error is to be caused in the transaction.
- When deadlock occurs between a transaction of UAP2 and a command, an error is to be caused in UAP2.

Example of operand specifications:

Specifying the client environment definition for executing UAP1

```
PDDLKPRIO 32
```

Specifying the client environment definition for executing UAP2

```
PDDLKPRIO 96
```

Specifying the system common definition

```
pd_deadlock_priority_use = Y
pd_command_deadlock_priority = 64
```

Chapter

10. Handling HiRDB File System Areas

This chapter explains the procedures for handling HiRDB file system areas and HiRDB files.

This chapter contains the following sections:

10.1 Obtaining information about a HiRDB file system area

Executor: HiRDB administrator

The commands shown in Table 10-1 can be used to obtain information about a HiRDB file system area.

Table 10-1: Commands used to display information about a HiRDB file system area

Command name	Purpose
`pdfstatfs`	• To obtain the size of the HiRDB file system area • To obtain the amount of unused space in the HiRDB file system area • To obtain the number of HiRDB files created in the HiRDB file system area • To obtain the number of HiRDB files that can still be created in the HiRDB file system area
`pdfls`	• To check the HiRDB files contained in the HiRDB file system area • To obtain the owner name, access privileges, record length, number of records, and update closure date and time of a HiRDB file • To obtain the lock status of a HiRDB file

For details on the information displayed by executing the `pdfstatfs` and `pdfls` commands, see the manual *HiRDB Version 8 Command Reference*.

10.2 Creating (initializing) a HiRDB file system area

This section explains how to create (initialize) a HiRDB file system area.

A character special file or a regular file can be used as a HiRDB file system area by initializing the file with the `pdfmkfs` command. Following is the procedure for creating a HiRDB file system area:

Procedure

1. Link the file name symbolically.
2. Change the owner and access privileges for the HiRDB file system area (applicable to character special files).
3. Initialize the HiRDB file system area.

(1) Link the file name symbolically

Executor: Superuser

It is recommended that the name of a HiRDB file system area be a name linked symbolically by the OS's `ln` command to the entity name of the character special file or regular file, rather than using the entity name itself. This simplifies the following operations:

- *Recovery of the HiRDB file system area onto a different hard disk in the event of a hard disk error*
- *Modification of RDAREA structure*

For information on the `ln` command, refer to the OS manual..

(2) Change the owner and access privileges for the HiRDB file system area (applicable to character special files)

Executor: Superuser

The owner and access privileges for the HiRDB file system area should be changed so that the file system area can be protected from access by unauthorized users.

Table 10-2 lists the owners and access privileges to be specified for a HiRDB file system area.

Table 10-2: Owners and access privileges to be set for HiRDB file system area

Owner or access privileges		Information to be set	Command to be executed*
Owner	User ID	HiRDB administrator	`chown` command
	Group ID	HiRDB group	`chgrp` command

Owner or access privileges		Information to be set	Command to be executed*
Access privileges	Owner	rw (both read and write permitted)	chmod command
	Group	rw (both read and write permitted)	
	Other	— (access not permitted)	

* These are OS commands; see the OS manual.

(3) Initialize the HiRDB file system area

Executor: HiRDB administrator

The pdfmkfs command is used to initialize the character special file or regular file so that it can be used as a HiRDB file system area. For details on HiRDB file system area design guidelines, see the manual *HiRDB Version 8 Installation and Design Guide*.

Following is an example of executing the pdfmkfs command:

Example

```
pdfmkfs -n 25 -l 5 -k DB -e 5 /svr01DB001
```

Explanation

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

This size must not exceed the size of the partition; if it is larger than the partition size, the subsequent physical partition may be damaged.

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

-k: Specifies the usage of the HiRDB file system area:

DB: HiRDB file system area for RDAREAs

SYS: HiRDB file system area for system files

WORK: HiRDB file system area for work table files

UTL: HiRDB file system area for utilities

SVR: HiRDB file system area for all purposes (except for utilities)

-e: Specifies the maximum number of secondary allocations for the HiRDB files.

/svr01DB001: Specifies a name for the HiRDB file system area (character

special file or regular file) that is to be created. If this is a character special file, specify a symbolically linked name.

Duplicating the HiRDB file system area using a mirror disk

If the HiRDB file system area is to be duplicated using a mirror disk, the following should be done:

- 3050RX group or 3500/3xx

  Specify in the `pdfmkfs` command the name of the character special file on the master disk (/dev/rdsk/rdskxxx) or the name linked symbolically to it.

- 3500 series (except 3500/3xx)

  Specify in the `pdfmkfs` command the mirror special file name (/dev/mirror/rdskxxx) or the name linked symbolically to it.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

10.3 Backing up a HiRDB file system area

Executor: HiRDB administrator

The `pdfbkup` command is used to back up a HiRDB file system area. The `pdfbkup` command should be used in the following cases:

- When a HiRDB file system area is to be defragmented
- When attributes, such as the capacity of a HiRDB file system area and the number of files, are to be changed
- When the contents of a HiRDB file system area should be stored in order to prepare for possible errors (backing up a database in units of HiRDB file system areas, rather than in units of RDAREAs)

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

10.4 Restoring a HiRDB file system area

Executor: HiRDB administrator

A HiRDB file system area can be restored from a backup copy created with the `pdfbkup` command. The `pdfrstr` command can be used to restore a HiRDB file system area in the following cases:

- When a HiRDB file system area has been defragmented
- When attributes, such as the capacity of a HiRDB file system area and the number of files, are to be changed
- When it is necessary to restore a HiRDB file system area from a backup copy because of an error*

* To restore a database to the synchronization point immediately before the error, the HiRDB file system area must be restored and then the database recovery utility must be executed to restore the database using the unload log only.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

10.5 Deleting a HiRDB file system area

Executor: HiRDB administrator

(1) Deleting a HiRDB file constituting an RDAREA

The database structure modification utility (`pdmod` command) is used to delete a HiRDB file or to change the size of a HiRDB file that is a constituent of an RDAREA.

There is no need to terminate HiRDB to re-create a HiRDB file deleted by the database structure modification utility.

Notes

- The `pdfrm` command must be used with caution because it deletes forcibly any specified HiRDB file regardless of the RDAREA structure.
- The `pdfrm` command cannot be used during HiRDB operation to delete a HiRDB file in a HiRDB file system area that is in use by HiRDB. The `pdfrm` command should be used only after HiRDB has been terminated.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

(2) Deleting a system file

The operation commands listed in Table 10-3 are used to delete system files.

Table 10-3: Commands used to delete system files

Command	Type of system file
`pdlogrm -d sys`	System log file
`pdlogrm -d spd`	Synchronization point dump file
`pdstsrm`	Status file

Notes

- If a system file cannot be deleted with the applicable command shown in Table 10-3, the reason for the deletion failure should be determined. If it is safe to delete the file, the `pdfrm` command can be used.
- A deleted HiRDB file can be re-created with an operation command without having to terminate HiRDB.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results,

see the manual *HiRDB Version 8 Command Reference.*

Chapter

11. Modifying the System Configuration

This chapter explains how to modify the unit or server configuration of a HiRDB/Parallel Server. It also explains the procedures for migrating from a HiRDB/Single Server to a HiRDB/Parallel Server.

This chapter contains the following sections:

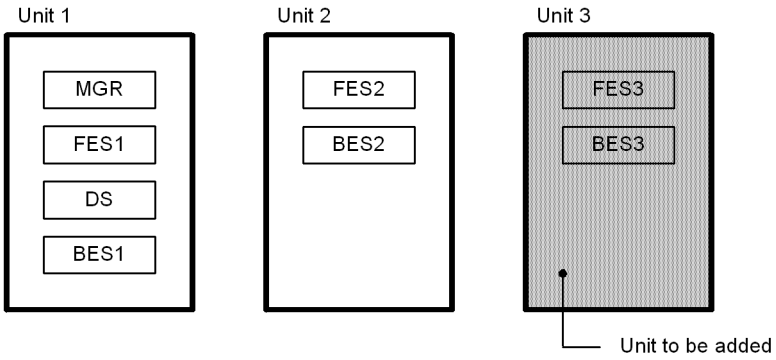
11.1 Adding a unit

Executor: HiRDB administrator

This section explains how to add a unit to a HiRDB/Parallel Server.

11.1.1 Adding a unit while HiRDB is running

Add unit 3 to a HiRDB/Parallel Server. The added unit contains a front-end server (FES3) and a back-end server (BES3). The procedure is detailed below.



Use of the system reconfiguration command (`pdchgconf` command) eliminates the need to terminate HiRDB normally. Note that HiRDB Advanced High Availability must be installed in order to use this command.

(1) Deploy a new server machine

Deploy a new server machine, and install and set up HiRDB. For details about installing and setting up HiRDB, see the manual *HiRDB Version 8 Installation and Design Guide*.

Check that the following are the same as in your existing units:

- HiRDB version
- Addressing mode (32-bit or 64-bit mode)
- POSIX library version installed/not installed
- Character codes
- Installation status of optional program products and related products

(2) Create updated HiRDB system definitions

Use the procedure explained below to create HiRDB system definitions that reflect the

change in the unit configuration. If you plan to add or move RDAREAs as explained in step (6), update the global buffer definitions as well.

Procedure

1. Create a `$PDDIR/conf/chgconf` directory.
2. Copy the current HiRDB system definition files into the directory you created in step 1.
3. Modify the HiRDB system definitions that are in the `$PDDIR/conf/chgconf` directory.
4. Copy the HiRDB definition files in the `$PDDIR/conf/chgconf` directory to the `$PDDIR/conf` directory. Perform the operation in step 4 only for the server machine of the unit to be added.

Note:

Do not change the `pdstart` operand value for a server in the existing units (units 1 and 2). If it is changed, you must take additional actions, such as initializing system files. For details about the actions to be taken when the `pdstart` operand value is changed, see the description of the `pdstart` operand in the manual *HiRDB Version 8 System Definition*.

(3) Use the pdloginit and pdstsinit commands to create the system files required for unit 3

```
pdloginit -d sys -s fes3 -f /sysarea/log01 -n 5000 -D         1
   :
pdloginit -d spd -s fes3 -f /sysarea/sync01 -n 5000 -D        2
   :
pdstsinit -s fes3 -f /sysarea/ssts01 -c 3000 -D               3
   :
pdstsinit -u UNT3 -f /sysarea/usts01 -c 3000 -D               4
   :
```

Explanation

1. Creates system log files for FES3 and BES3.
2. Creates synchronization point dump files for FES3 and BES3.
3. Creates server status files for FES3 and BES3.
4. Creates unit status files for unit 3.

Because these commands are executed in unit 3 rather than in unit 1 (system manager unit), the `-D` option must be specified for the `pdloginit` and `pdstsinit` commands.

(4) Use the pdconfchk command to check the updated HiRDB system definitions

```
pdconfchk -d chgconf
```

Check the HiRDB system definitions in the `$PDDIR/conf/chgconf` directory. If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(5) Use the pdchgconf command to modify the HiRDB system definitions

```
pdchgconf
```

Replace the HiRDB system definitions with the updated HiRDB system definitions.

(6) Use the pdmod command to add or move RDAREAs

If necessary, you can add or move RDAREAs to BES3. For details about adding RDAREAs, see *15.2 Creating an RDAREA (RDAREA addition)*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

If you did not modify the global buffer definitions in step (2), use the `pdbufmod` command to update the global buffers dynamically.

(7) Modify client environment definitions

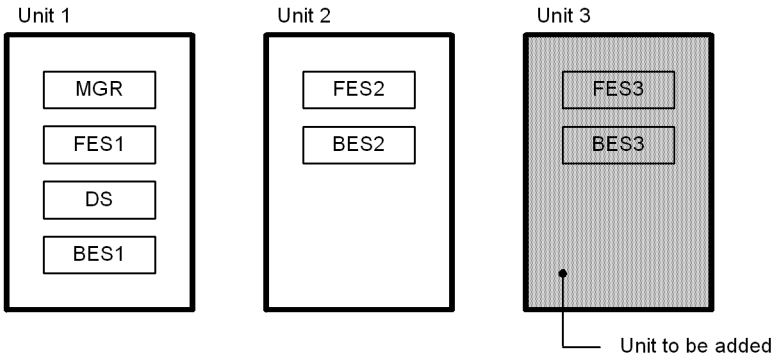
If necessary, specify the following client environment definition operands for the front-end server (FES3) that you added:

- `PDFESHOST`
- `PDSERVICEGRP`
- `PDSERVICEPORT`

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

11.1.2 Adding a unit while HiRDB is stopped

Add unit 3 to a HiRDB/Parallel Server. The added unit contains a front-end server (FES3) and a back-end server (BES3). The procedure is detailed below.



(1) Deploy a new server machine

Deploy a new server machine, and install and set up HiRDB. For details about installing and setting up HiRDB, see the manual *HiRDB Version 8 Installation and Design Guide*.

Check that the following are the same as in your existing units:

- HiRDB version
- Addressing mode (32-bit or 64-bit mode)
- POSIX library version installed/not installed
- Character codes
- Installation status of optional program products and related products

(2) Use the pdstop command to stop HiRDB normally

```
pdstop
```

Check that HiRDB terminates normally.

(3) Modify HiRDB system definitions

Create HiRDB system definitions that reflect the change in the unit configuration. If you plan to add or move RDAREAs as explained in step (7), update the global buffer definitions as well.

Note:

Do not change the `pdstart` operand value for a server in the existing units (units 1 and 2). If it is changed, you must take additional actions, such as initializing system files. For details about the actions to be taken when the `pdstart` operand value is changed, see the description of the `pdstart` operand in the manual *HiRDB Version 8 System Definition*.

(4) Use the pdloginit and pdstsinit commands to create the system files required for unit 3

```
pdloginit -d sys -s fes3 -f /sysarea/log01 -n 5000          1
   :
pdloginit -d spd -s fes3 -f /sysarea/sync01 -n 5000         2
   :
pdstsinit -s fes3 -f /sysarea/ssts01 -c 3000                3
   :
pdstsinit -u UNT3 -f /sysarea/usts01 -c 3000                4
   :
```

Explanation

1. Creates system log files for FES3 and BES3.
2. Creates synchronization point dump files for FES3 and BES3.
3. Creates server status files for FES3 and BES3.
4. Creates unit status files for unit 3.

(5) Use the pdconfchk command to check the HiRDB system definitions

```
pdconfchk
```

If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(6) Use the pdstart command to start HiRDB normally

```
pdstart
```

(7) Use the pdmod command to add or move RDAREAs

If necessary, you can add or move RDAREAs to BES3. For details about adding RDAREAs, see *15.2 Creating an RDAREA (RDAREA addition)*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

(8) Modify client environment definitions

If necessary, specify the following client environment definition operands for the front-end server (FES3) that you added:

- PDFESHOST
- PDSERVICEGRP
- PDSERVICEPORT

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

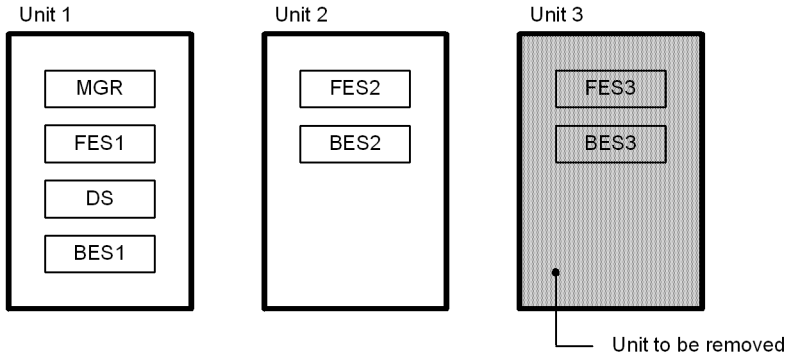
11.2 Removing a unit

Executor: HiRDB administrator

This section explains how to remove a unit from a HiRDB/Parallel Server.

11.2.1 Removing a unit while HiRDB is running

Remove unit 3 from a HiRDB/Parallel Server. The procedure is detailed below.



Use of the system reconfiguration command (`pdchgconf` command) eliminates the need to terminate HiRDB normally. Note that HiRDB Advanced High Availability must be installed in order to use this command.

Notes

- You cannot remove a unit that contains the system manager or dictionary server.
- You cannot remove a unit if there would be no more front-end servers.
- You cannot remove a unit if there would be no more back-end servers.

(1) Use the pdmod command to delete or move the RDAREAs of BES3

For details about deleting RDAREAs, see *15.6 Deleting an RDAREA*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

(2) Modify client environment definitions

Check if the following operands of the client environment definitions are specified; if the front-end server (FES3) that you plan to remove is specified in these operands, change their values as appropriate:

- `PDFESHOST`

- PDSERVICEGRP
- PDSERVICEPORT

(3) Create updated HiRDB system definitions

Use the procedure explained below to create HiRDB system definitions that reflect the change in the unit configuration.

Procedure

1. Create a $PDDIR/conf/chgconf directory.
2. Copy the current HiRDB system definition files into the directory that you created in step 1.
3. Modify the HiRDB system definitions that are in the $PDDIR/conf/chgconf directory.

If you are using the HiRDB External Data Access facility

If a back-end server for connecting to foreign servers is in the unit to be removed, delete the definitions related to foreign servers. If you remove the unit without deleting the foreign server-related definitions, either the configuration change will fail or an error will occur when an attempt is made to access a table or an index stored in an RDAREA of the unit that was removed.

(4) Use the pdconfchk command to check the updated HiRDB system definitions

```
pdconfchk -d chgconf
```

Check the HiRDB system definitions in the $PDDIR/conf/chgconf directory. If errors are detected, correct the HiRDB system definitions, and then execute the pdconfchk command again.

(5) Use the pdchgconf command to modify the HiRDB system definitions

```
pdchgconf
```

Replace the HiRDB system definitions with the updated HiRDB system definitions.

(6) Use the pdlogrm and pdstsrm commands to delete the unit 3 system files

```
pdlogrm -d sys -s fes3 -f /sysarea/log01 -D          1
   :
pdlogrm -d spd -s fes3 -f /sysarea/sync01 -D         2
   :
pdstsrm -s fes3 -f /sysarea/ssts01 -D                3
   :
pdstsrm -u UNT3 -f /sysarea/usts01  -D               4
   :
```

Explanation

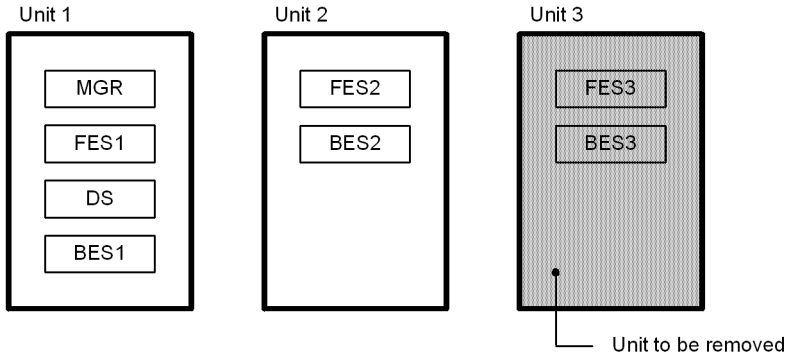
1. Deletes the system log files for FES3 and BES3.
2. Deletes the synchronization point dump files for FES3 and BES3.
3. Deletes the server status files for FES3 and BES3.
4. Deletes the unit status files for unit 3.

   Because these commands are executed in unit 3 rather than in unit 1 (system manager unit), the `-D` option must be specified for the `pdloginit` and `pdstsinit` commands.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

11.2.2 Removing a unit while HiRDB is stopped

Remove unit 3 from a HiRDB/Parallel Server. The procedure is detailed below.



Notes

- Do not remove a unit containing the system manager or dictionary server.

- Do not remove a unit if there would be no more front-end servers.
- Do not remove a unit if there would be no more back-end servers.

(1) Use the pdmod command to delete or move the RDAREAs of BES3

For details about deleting RDAREAs, see *15.6 Deleting an RDAREA*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

(2) Modify client environment definitions

Check if the following operands in the client environment definitions are specified; if the front-end server (FES3) that you plan to remove is specified in these operands, change their values as appropriate:

- PDFESHOST
- PDSERVICEGRP
- PDSERVICEPORT

(3) Use the pdstop command to stop HiRDB normally

```
pdstop
```

Check that HiRDB terminates normally.

(4) Use the pdlogrm and pdstsrm commands to delete the unit 3 system files

```
pdlogrm -d sys -s fes3 -f /sysarea/log01          1
   :
pdlogrm -d spd -s fes3 -f /sysarea/sync01         2
   :
pdstsrm -s fes3 -f /sysarea/ssts01                3
   :
pdstsrm -u UNT3 -f /sysarea/usts01                4
   :
```

Explanation

1. Deletes the system log files for FES3 and BES3.
2. Deletes the synchronization point dump files for FES3 and BES3.
3. Deletes the server status files for FES3 and BES3.
4. Deletes the unit status files for unit 3.

(5) Modify HiRDB system definitions

Create HiRDB system definitions that reflect the change in the unit configuration.

Note when the HiRDB External Data Access facility is used

If a unit to be deleted contains a back-end server for external connection, delete definitions about the external server. If the unit is deleted without deleting the external server-related definitions, the configuration change may fail or an attempt to access a table or index stored in an RDAREA on the unit to be deleted will result in an error.

(6) Use the pdconfchk command to check the HiRDB system definitions

```
pdconfchk
```

If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(7) Use the pdstart command to start HiRDB normally

```
pdstart
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

11.3 Moving a unit

Executor: HiRDB administrator

This section explains how to move a unit within a HiRDB/Parallel Server.

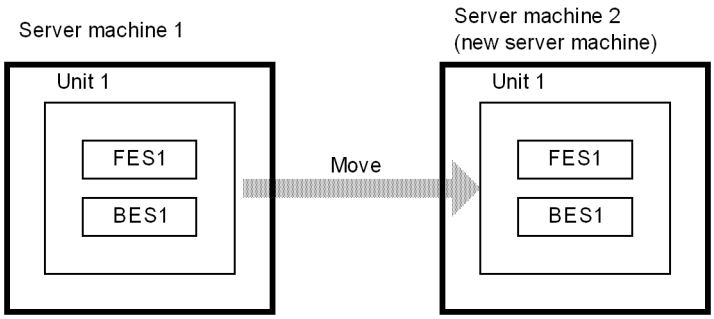
Note:

If a heterogeneous system configuration is employed and a unit containing a back-end server is to be moved, observe the following rule:

- Make sure that the same platform is used at the server machines before and after moving the unit containing the back-end server. Do not move a unit to a different platform.

11.3.1 Moving a unit while HiRDB is running

Move a unit of a HiRDB/Parallel Server. The procedure is detailed below.



Use of the system reconfiguration command (`pdchgconf` command) eliminates the need to terminate HiRDB normally. Note that HiRDB Advanced High Availability must be installed in order to use this command.

Notes

- You cannot move a system manager unit.
- The disk on which you create the system files and the RDAREAs for the unit to be moved must be shared by means of a SAN or other storage network system between both the current server machine and the new server machine.
- You can move a unit only if you are using the HP-UX or AIX version of HiRDB. This is because OS commands must be used during the configuration change to disconnect and connect the shared disk, which means that units can be moved only on operating systems that provide such commands (HP-UX and AIX).

(1) Deploy a new server machine

Deploy a new server machine, and install and set up HiRDB. For details about installing and setting up HiRDB, see the manual *HiRDB Version 8 Installation and Design Guide*.

Check that the following are the same as in your existing units:

- HiRDB version
- Addressing mode (32-bit or 64-bit mode)
- POSIX library version installed/not installed
- Character codes
- Installation status of optional program products and related products

(2) Prepare shells for disconnecting and connecting the shared disk

Prepare the shells from which you will disconnect and connect the shared disk. Execute the shells as a superuser. For safety's sake, do not include in these shells any commands other than commands for disconnecting and connecting the disk.

Shell on current server machine

Create a shell (`$PDDIR/conf/chgconf/diskdiscon.sh`) containing the command for disconnecting the disk storing the system files and RDAREAs from the current server machine. The following shows an example on HP-UX:

`/usr/sbin/vgchange -a n` *device-name*

Shell on new server machine

Create a shell (`$PDDIR/conf/chgconf/diskcon.sh`) containing the command for connecting the disk storing the system files and RDAREAs onto the new server machine. The following shows an example on HP-UX:

`/usr/sbin/vgchange -a y` *device-name*

(3) Create updated HiRDB system definitions

Use the procedure explained below to create HiRDB system definitions that reflect the change in the unit configuration.

Procedure

1. Create a `$PDDIR/conf/chgconf` directory.
2. Copy the current HiRDB system definition files into the directory that you created in step 1.
3. Modify the HiRDB system definitions that are in the `$PDDIR/conf/chgconf` directory.

(4) Use the pdconfchk command to check the updated HiRDB system definitions

```
pdconfchk -d chgconf
```

Check the HiRDB system definitions in the `$PDDIR/conf/chgconf` directory. If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(5) Use the pdchgconf command to modify the HiRDB system definitions

```
pdchgconf
```

Replace the HiRDB system definitions with the updated HiRDB system definitions. At this time, the shells that you created in step (2) for disconnecting and connecting the disk are executed.

Note

If system reconfiguration processing fails, the `pdchgconf` command may end in an error without replacing the HiRDB system definitions. If the `pdchgconf` command ends in an error, check the files in the `$PDDIR/conf` directory.

(6) Modify client environment definitions

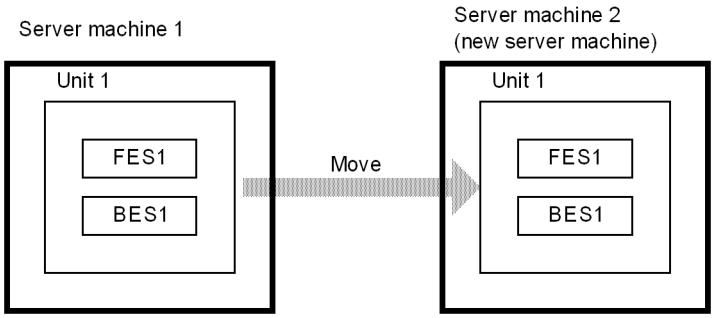
Check if the following operands in the client environment definitions are specified; if the front-end server (FES1) that you moved is specified in these operands, change their values as appropriate:

- `PDFESHOST`
- `PDSERVICEGRP`
- `PDSERVICEPORT`

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

11.3.2 Moving a unit while HiRDB is stopped

Move a unit of a HiRDB/Parallel Server. The procedure is detailed below.



Note

Because moving the system manager unit has far-ranging effects, we recommend that you do not move the system manager unit.

(1) Deploy a new server machine

Deploy a new server machine, and install and set up HiRDB. For details about installing and setting up HiRDB, see the manual *HiRDB Version 8 Installation and Design Guide*.

Check that the following are the same as in your existing units:

- HiRDB version
- Addressing mode (32-bit or 64-bit mode)
- POSIX library version installed/not installed
- Character codes
- Installation status of optional program products and related products

(2) Use the pdrorg command to unload the table data on BES1, one RDAREA at a time

For details about unloading table data on a per-RDAREA basis, see the manual *HiRDB Version 8 Command Reference*.

(3) Use the pdstop command to stop HiRDB normally

```
pdstop
```

Check that HiRDB terminates normally.

(4) Use the pdlogls command to check the status of the system log files in the source unit

```
pdlogls -d sys -s bes1
```

(5) Use the pdlogunld command to unload any system log files in unload wait status

```
pdlogunld -d sys -s bes1 -g log01 -o /unld/unldlog01
```

(6) Use the pdlogrm and pdstsrm commands to delete the system files in the source unit

```
pdlogrm -d sys -s fes1 -f /sysarea/log01          1
   :
pdlogrm -d spd -s fes1 -f /sysarea/sync01         2
   :
pdstsrm -s fes1 -f /sysarea/ssts01               3
   :
pdstsrm -u UNT1 -f /sysarea/usts01               4
   :
```

Explanation

1. Deletes the system log files (for FES1 and BES1) in the source unit.
2. Deletes the synchronization point dump files (for FES1 and BES1) in the source unit.
3. Deletes the server status files (for FES1 and BES1) in the source unit.
4. Deletes the unit control files in the source unit.

(7) Modify HiRDB system definitions

Create HiRDB system definitions that reflect the changes in the unit configuration.

(8) Use the pdloginit and pdstsinit commands to create the system files needed for the target unit

```
pdloginit -d sys -s fes1 -f /sysarea/log01 -n 5000          1
   :
pdloginit -d spd -s fes1 -f /sysarea/sync01 -n 5000         2
   :
pdstsinit -s fes1 -f /sysarea/ssts01 -c 3000                3
   :
pdstsinit -u UNT1 -f /sysarea/usts01 -c 3000                4
   :
```

Explanation

1. Creates system log files for FES1 and BES1.
2. Creates synchronization point dump files for FES1 and BES1.
3. Creates server status files for FES1 and BES1.
4. Creates unit status files for unit 1.

(9) Use the pdconfchk command to check the HiRDB system definitions

```
pdconfchk
```

If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(10) Use the pdstart command to start HiRDB normally

```
pdstart
```

(11) Use the pdcopy command to back up all RDAREAs

For details about making backups, see *6.4 Examples of backup*.

(12) Use the pdmod command to re-initialize the RDAREAs in the target unit

For details about re-initializing RDAREAs, see *15.4 Increasing the size of an RDAREA or modifying its attributes (RDAREA reinitialization)*.

(13) Use the pdrorg command to reload the table data in the target unit, one RDAREA at a time

Use the unload data file that you created in step (2) as input data. For details about reloading on a per-RDAREA basis, see the manual *HiRDB Version 8 Command Reference*.

(14) Use the pdcopy command to back up all RDAREAs

For details about making backups, see *6.4 Examples of backup*.

(15) Modify client environment definitions

Check if the following operands in the client environment definitions are specified; if the front-end server (FES1) that you moved is specified in these operands, change their values as appropriate:

- `PDFESHOST`
- `PDSERVICEGRP`
- `PDSERVICEPORT`

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

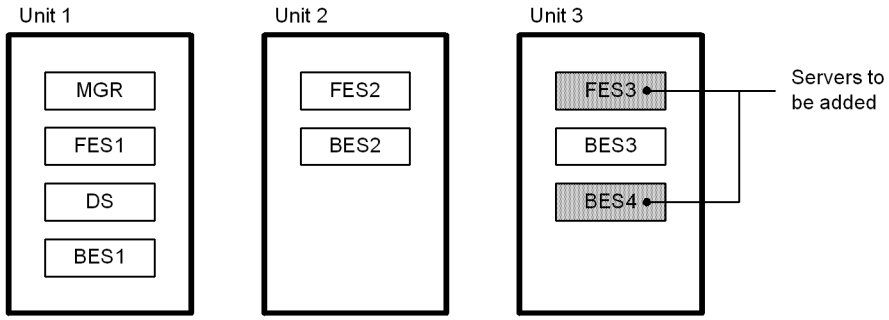
11.4 Adding a server

Executor: HiRDB administrator

This section explains how to add a server to a HiRDB/Parallel Server.

11.4.1 Adding a server while HiRDB is running

Add a front-end server (FES3) and a back-end server (BES3) to unit 3. The procedure is detailed below.



Use of the system reconfiguration command (`pdchgconf` command) eliminates the need to terminate HiRDB normally. Note that HiRDB Advanced High Availability must be installed in order to use this command.

(1) Check the memory requirements for unit 3

To add the servers, you must re-estimate the memory requirements for unit 3. You may also need to modify operating system parameters of the OS. For details about memory requirements and evaluating operating system parameters, see the manual *HiRDB Version 8 Installation and Design Guide*.

Note that you must restart the OS for changes to operating system parameters to take effect, which means that HiRDB must be terminated normally. In this example, we assume that there are no operating system parameters to be changed.

(2) Create updated HiRDB system definitions

Use the procedure explained below to create HiRDB system definitions that reflect the change in the server configuration. If you plan to add or move RDAREAs as explained in step (6), update the global buffer definitions as well.

Procedure

1. Create a `$PDDIR/conf/chgconf` directory.
2. Copy the current HiRDB system definition files into the directory you

created in step 1.

3. Modify the HiRDB system definitions that are in the `$PDDIR/conf/chgconf` directory.

(3) Use the pdloginit and pdstsinit commands to create the system files required for FES3 and BES4

```
pdloginit -d sys -s fes3 -f /sysarea/log01 -n 5000 -D         1
   :
pdloginit -d spd -s fes3 -f /sysarea/sync01 -n 5000 -D        2
   :
pdstsinit -s fes3 -f /sysarea/ssts01 -c 3000 -D               3
   :
```

Explanation

1. Creates system log files for FES3 and BES4.
2. Creates synchronization point dump files for FES3 and BES4.
3. Creates server status files for FES3 and BES4.

Because these commands are executed in unit 3 rather than in unit 1 (system manager unit), the `-D` option must be specified for the `pdloginit` and `pdstsinit` commands.

(4) Use the pdconfchk command to check the updated HiRDB system definitions

```
pdconfchk -d chgconf
```

Check the HiRDB system definitions in the `$PDDIR/conf/chgconf` directory. If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(5) Use the pdchgconf command to modify the HiRDB system definitions

```
pdchgconf
```

Replace the HiRDB system definitions with the updated HiRDB system definitions.

(6) Use the pdmod command to add or move RDAREAs

If necessary, you can add or move RDAREAs to BES4. For details about adding RDAREAs, see *15.2 Creating an RDAREA (RDAREA addition)*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

If you did not modify the global buffer definitions in step (2), use the `pdbufmod` command to update the global buffers dynamically.

(7) Modify client environment definitions

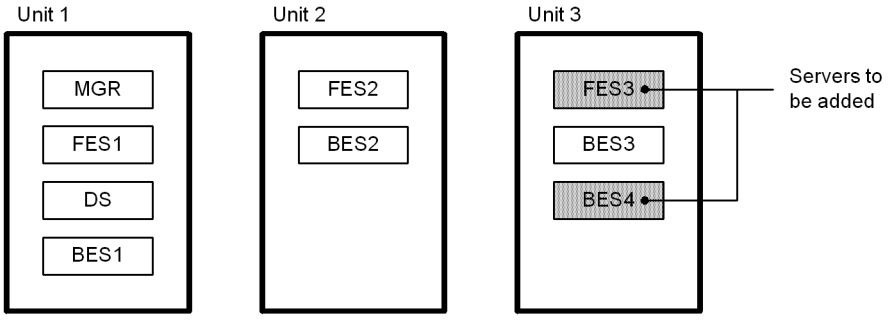
If necessary, specify the following operands in the client environment definitions for the front-end server (FES3) that you added:

- `PDFESHOST`
- `PDSERVICEGRP`
- `PDSERVICEPORT`

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

11.4.2 Adding a server while HiRDB is stopped

Add a front-end server (FES3) and a back-end server (BES4) to unit 3. The procedure is detailed below.



(1) Check the memory requirements for unit 3

To add the servers, you must re-estimate the memory requirements for unit 3. You may also need to modify operating system parameters of the OS. For details about memory requirements and evaluating operating system parameters, see the manual *HiRDB Version 8 Installation and Design Guide*.

(2) Use the pdstop command to stop HiRDB normally

```
pdstop
```

Check that HiRDB terminates normally.

(3) Use the pdlogls command to check the status of the system log files in unit 3

```
pdlogls -d sys -s bes3
```

(4) Use the pdlogunld command to unload any system log files in unload wait status

```
pdlogunld -d sys -s bes3 -g log01 -o /unld/unldlog01
```

If you decide not to unload the system logs, use the `pdcopy` command to back up all RDAREAs. For details about making backups, see *6.4 Examples of backup*.

(5) Use pdlogrm and pdstsrm commands to delete the system files required for unit 3

```
pdlogrm -d sys -s bes3 -f /sysarea/log01          1
   :
pdlogrm -d spd -s bes3 -f /sysarea/sync01         2
   :
pdstsrm -s bes3 -f /sysarea/ssts01                3
   :
pdstsrm -u UNT3 -f /sysarea/usts01                4
   :
```

Explanation

1. Deletes system log files for BES3.
2. Deletes synchronization point dump files for BES3.
3. Deletes server status files for BES3.
4. Deletes unit status files for unit 3.

(6) Modify HiRDB system definitions

Create HiRDB system definitions that reflect the changes in the server configuration. If you plan to add or move RDAREAs in step (10), also modify the global buffer definitions.

(7) Use the pdloginit and pdstsinit commands to create the system files required for unit 3

```
pdloginit -d sys -s fes3 -f /sysarea/log01 -n 5000          1
   :
pdloginit -d spd -s fes3 -f /sysarea/sync01 -n 5000         2
   :
pdstsinit -s fes3 -f /sysarea/ssts01 -c 3000                3
   :
pdstsinit -u UNT3 -f /sysarea/usts01 -c 3000                4
   :
```

Explanation

1. Creates system log files for FES3, BES3, and BES4.
2. Creates synchronization point dump files for FES3, BES3, and BES4.
3. Creates server status files for FES3, BES3, and BES4.
4. Creates unit status files for unit 3.

Because you re-create the status files, the error shutdown status of RDAREAs is not

inherited. If necessary, use the pdhold command to place RDAREAs in shutdown status again.

(8) Use the pdconfchk command to check the HiRDB system definitions

```
pdconfchk
```

If errors are detected, correct the HiRDB system definitions, and then execute the pdconfchk command again.

(9) Use the pdstart command to start HiRDB normally

```
pdstart
```

(10) Use the pdmod command to add or move RDAREAs

If necessary, you can add or move RDAREAs to BES4. For details about adding RDAREAs, see *15.2 Creating an RDAREA (RDAREA addition)*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

(11) Modify client environment definitions

If necessary, specify the following operands in the client environment definitions for the front-end server (FES3) that you added:

- PDFESHOST
- PDSERVICEGRP
- PDSERVICEPORT

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

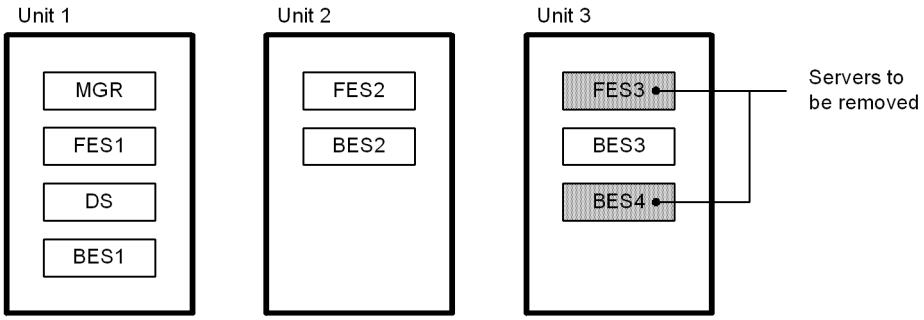
11.5 Removing a server

Executor: HiRDB administrator

This section explains how to remove a server from a HiRDB/Parallel Server.

11.5.1 Removing a server while HiRDB is running

Remove a front-end server (FES3) and a back-end server (BES3) from a HiRDB/Parallel Server. The procedure is detailed below.



Use of the system reconfiguration command (`pdchgconf` command) eliminates the need to terminate HiRDB normally. Note that HiRDB Advanced High Availability must be installed in order to use this command.

Notes

- You cannot remove the system manager or dictionary server.
- You cannot remove a front-end server if there would be no more front-end servers.
- You cannot remove a back-end server if there would be no more back-end servers.

(1) Use the pdmod command to delete or move the RDAREAs of BES4

For details about deleting RDAREAs, see *15.6 Deleting an RDAREA*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

(2) Modify client environment definitions

Check if the following operands in the client environment definitions are specified; if the front-end server (FES3) that you plan to remove is specified in these operands, change their values as appropriate:

- `PDFESHOST`

- PDSERVICEGRP
- PDSERVICEPORT

(3) Create updated HiRDB system definitions

Use the procedure explained below to create HiRDB system definitions that reflect the change in the server configuration.

Procedure

1. Create a $PDDIR/conf/chgconf directory.
2. Copy the current HiRDB system definition files into the directory that you created in step 1.
3. Modify the HiRDB system definitions that are in the $PDDIR/conf/chgconf directory.

If you are using the HiRDB External Data Access facility

If you plan to remove a back-end server for connecting to foreign servers, delete the definitions related to foreign servers. If you remove the server without deleting the foreign server-related definitions, either the configuration change will fail or an error will occur when an attempt is made to access a table or an index stored in an RDAREA of the server that was removed.

(4) Use the pdconfchk command to check the updated HiRDB system definitions

```
pdconfchk -d chgconf
```

Check the HiRDB system definitions in the $PDDIR/conf/chgconf directory. If errors are detected, correct the HiRDB system definitions, and then execute the pdconfchk command again.

(5) Use the pdchgconf command to modify the HiRDB system definitions

```
pdchgconf
```

Replace the HiRDB system definitions with the updated HiRDB system definitions.

(6) Use the pdlogrm and pdstsrm commands to delete the FES3 and BES4 system files

```
pdlogrm -d sys -s fes3 -f /sysarea/log01 -D          1
   :
pdlogrm -d spd -s fes3 -f /sysarea/sync01 -D         2
   :
pdstsrm -s fes3 -f /sysarea/ssts01 -D                3
   :
```

Explanation

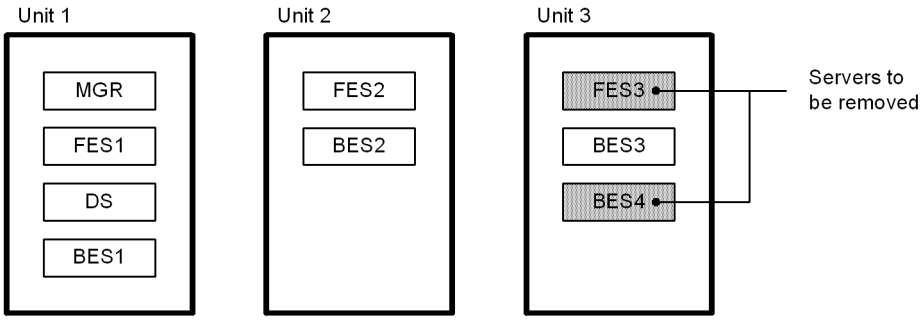
1. Deletes the system log files for FES3 and BES4.
2. Deletes the synchronization point dump files for FES3 and BES4.
3. Deletes the server status files for FES3 and BES4.

Because these commands are executed in unit 3 rather than in unit 1 (system manager unit), the `-D` option must be specified in the `pdloginit` and `pdstsinit` commands.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

11.5.2 Removing a server while HiRDB is stopped

Remove a front-end server (FES3) and a back-end server (BES3) from a HiRDB/ Parallel Server. The procedure is detailed below.



Notes

- Do not remove the system manager or dictionary server manager.
- Do not remove a front-end server if there would be no more front-end servers.

- Do not remove a back-end server if there would be no more back-end servers.

(1) Use the pdmod command to delete or move the RDAREAs of BES4

For details about deleting RDAREAs, see *15.6 Deleting an RDAREA*; for details about moving RDAREAs, see *15.8 Moving an RDAREA (RDAREA migration)*.

(2) Modify client environment definitions

Check if the following operands in the client environment definitions are specified; if the front-end server (FES3) that you plan to remove is specified in these operands, change their values as appropriate:

- PDFESHOST
- PDSERVICEGRP
- PDSERVICEPORT

(3) Use the pdstop command to stop HiRDB normally

```
pdstop
```

Check that HiRDB terminates normally.

(4) Use the pdlogls command to check the status of the system log files in unit 3

```
pdlogls -d sys -s bes3
```

(5) Use the pdlogunld command to unload any system log files in unload wait status

```
pdlogunld -d sys -s bes3 -g log01 -o /unld/unldlog01
```

(6) Use the pdlogrm and pdstsrm commands to delete the unit 3 system files

```
pdlogrm -d sys -s fes3 -f /sysarea/log01            1
   :
pdlogrm -d spd -s fes3 -f /sysarea/sync01           2
   :
pdstsrm -s fes3 -f /sysarea/ssts01                  3
   :
pdstsrm -u UNT3 -f /sysarea/usts01                  4
   :
```

Explanation

1. Deletes the system log files for FES3, BES3, and BES4.
2. Deletes the synchronization point dump files for FES3, BES3, and BES4.
3. Deletes the server status files for FES3, BES3, and BES4.
4. Deletes the unit status files for unit 3.

(7) Change the HiRDB system definition

Create HiRDB system definitions that reflect the changes in the server configuration.

Note when the HiRDB External Data Access facility is used

If a back-end server for external server connection is deleted, delete definitions about the external server. If the unit is deleted without deleting the external server-related definitions, the configuration change may fail or an attempt to access a table or index stored in an RDAREA on the unit to be deleted will result in an error.

(8) Use the pdloginit and pdstsinit commands to create the system files needed for unit 3

```
pdloginit -d sys -s bes3 -f /sysarea/log01 -n 5000          1
   :
pdloginit -d spd -s bes3 -f /sysarea/sync01 -n 5000         2
   :
pdstsinit -s bes3 -f /sysarea/ssts01 -c 3000                3
   :
pdstsinit -u UNT3 -f /sysarea/usts01 -c 3000                4
   :
```

Explanation

1. Creates system log files for BES3.
2. Creates synchronization point dump files for BES3.
3. Creates server status files for BES3.
4. Creates unit status files for unit 3.

Because you re-created the status files, the error shutdown status of the RDAREAs is not inherited. If necessary, use the `pdhold` command to place RDAREAs in shutdown status again.

(9) Use the pdconfchk command to check the HiRDB system definitions

```
pdconfchk
```

If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(10) Use the pdstart command to start HiRDB normally

```
pdstart
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

11.6 Moving a server

Executor: HiRDB administrator

This section explains how to move a server within a HiRDB/Parallel Server.

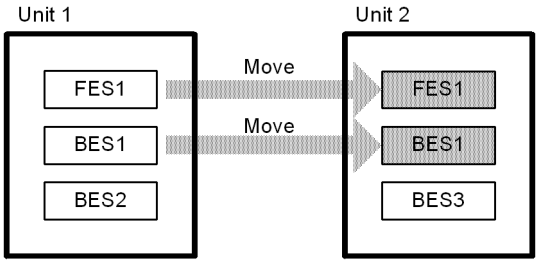
Note:

If a heterogeneous system configuration is employed and a back-end server is to be moved, observe the following rule:

- Make sure that the same platform is used at the server machines before and after moving the back-end server. Do not move a back-end server to a different platform.

11.6.1 Moving a server while HiRDB is running

Move a HiRDB/Parallel Server front-end server (FES1) and back-end server (BES1). The procedure is detailed below.



Use of the system reconfiguration command (`pdchgconf` command) eliminates the need to terminate HiRDB normally. Note that HiRDB Advanced High Availability must be installed in order to use this command.

Notes

- You cannot move a system manager.
- The disk on which you create the system files and the RDAREAs for the server to be moved must be shared by means of a SAN or other storage network system between both the current server machine and the new server machine.
- You can move a server only if you are using the HP-UX or AIX versions of HiRDB. This is because OS commands must be used during the configuration change to disconnect and connect the shared disk, which means that servers can be moved only on operating systems that provide such

commands (HP-UX and AIX).

(1) Determine the memory requirements for unit 2

To move the servers, you must re-estimate the memory requirements for unit 2. You may also need to modify operating system parameters of the OS. For details about memory requirements and evaluating operating system parameters, see the manual *HiRDB Version 8 Installation and Design Guide*.

Note that you must restart the OS for changes to operating system parameters to take effect, which means that HiRDB must be terminated normally. In this example, we assume no operating system parameters are to be changed.

(2) Prepare shells for disconnecting and connecting the shared disk

Prepare the shells from which you will disconnect and connect the shared disk. Execute the shells as a superuser. For safety's sake, do not include in these shells any commands other than commands for disconnecting and connecting the disk.

Shell on current server machine

Create a shell (`$PDDIR/conf/chgconf/diskdiscon.sh`) containing the command for disconnecting the disk storing the system files and RDAREAs from the current server machine. The following shows an example on HP-UX:

`/usr/sbin/vgchange -a n` *device-name*

Shell on new server machine

Create a shell (`$PDDIR/conf/chgconf/diskcon.sh`) containing the command for connecting the disk storing the system files and RDAREAs onto the new server machine. The following shows an example on HP-UX:

`/usr/sbin/vgchange -a y` *device-name*

(3) Create updated HiRDB system definitions

Use the procedure explained below to create HiRDB system definitions that reflect the change in the server configuration.

Procedure

1. Create a `$PDDIR/conf/chgconf` directory.
2. Copy the current HiRDB system definition files into the directory that you created in step 1.
3. Modify the HiRDB system definitions that are in the `$PDDIR/conf/chgconf` directory.

(4) Use the pdconfchk command to check the updated HiRDB system definitions

```
pdconfchk -d chgconf
```

Check the HiRDB system definitions in the `$PDDIR/conf/chgconf` directory. If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(5) Use the pdchgconf command to modify the HiRDB system definitions

```
pdchgconf
```

Replace the HiRDB system definitions with the updated HiRDB system definitions. At this time, the shells that you created in step (1) for disconnecting and connecting the disk are executed.

Note

If system reconfiguration processing fails, the `pdchgconf` command may end in an error without replacing the HiRDB system definitions. If the `pdchgconf` command ends in an error, check the connection state of the disks and the files in the `$PDDIR/conf` directory.

(6) Modify client environment definitions

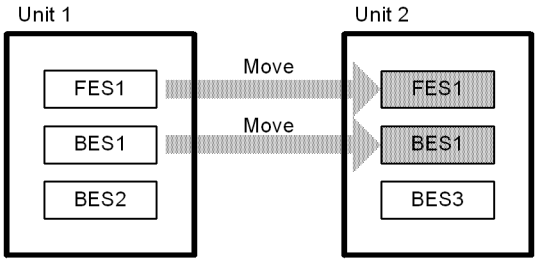
Check if the following operands in the client environment definitions are specified; if the front-end server (FES1) that you moved is specified in these operands, change their values as appropriate:

- `PDFESHOST`
- `PDSERVICEGRP`
- `PDSERVICEPORT`

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

11.6.2 Moving a server while HiRDB is stopped

Move a HiRDB/Parallel Server front-end server (FES1) and back-end server (BES1). The procedure is detailed below.



Note

Because moving the system manager has far-ranging effects, we recommend that you do not move the system manager.

(1) Determine the memory requirements for unit 2

To move the servers, you must re-estimate the memory requirements for unit 2. You may also need to modify operating system parameters of the OS. For details about memory requirements and evaluating operating system parameters, see the manual *HiRDB Version 8 Installation and Design Guide*.

(2) Use the pdrorg command to unload the table data on BES1, one RDAREA at a time

For details about unloading table data on a per-RDAREA basis, see the manual *HiRDB Version 8 Command Reference*.

(3) Use the pdstop command to stop HiRDB normally

```
pdstop
```

Check that HiRDB terminates normally.

(4) Use the pdlogls command to check the status of the system log files in unit 1

```
pdlogls -d sys -s bes1
pdlogls -d sys -s bes2
```

(5) Use the pdlogunld command to unload any system log files in unload wait status

```
pdlogunld -d sys -s bes1 -g log01 -o /unld/unldlog01
pdlogunld -d sys -s bes2 -g log02 -o /unld/unldlog02
```

If you decide not to unload the system logs, use the `pdcopy` command to back up all RDAREAs. For details about making backups, see *6.4 Examples of backup*.

(6) Use the pdlogrm and pdstsrm commands to delete all system files from units 1 and 2

```
pdlogrm -d sys -s fes1 -f /sysarea/log01          1
   :
pdlogrm -d spd -s fes1 -f /sysarea/sync01         2
   :
pdstsrm -s fes1 -f /sysarea/ssts01                3
   :
pdstsrm -u UNT1 -f /sysarea/usts01                4
   :
```

Explanation

1. Deletes the system log files for FES1, and BES1 to BES3.
2. Deletes the synchronization point dump files for FES1, and BES1 to BES3.
3. Deletes the server status files for FES1, and BES1 to BES3.
4. Deletes the unit status files for units 1 and 2.

(7) Change the HiRDB system definition

Create HiRDB system definitions that reflect the change in the server configuration.

(8) Use the pdloginit and pdstsinit commands to create the system files required for units 1 and 2

```
pdloginit -d sys -s fes1 -f /sysarea/log01 -n 5000          1
   :
pdloginit -d spd -s fes1 -f /sysarea/sync01 -n 5000         2
   :
pdstsinit -s fes1 -f /sysarea/ssts01 -c 3000                3
   :
pdstsinit -u UNT1 -f /sysarea/usts01 -c 3000                4
   :
```

Explanation

1. Creates system log files for FES1, and BES1 to BES3.
2. Creates synchronization point dump files for FES1, and BES1 to BES3.
3. Creates server status files for FES1, and BES1 to BES3.
4. Creates unit status files for units 1 and 2.

Because you re-created the status files, the error shutdown status of the RDAREAs is not inherited. If necessary, use the `pdhold` command to place RDAREAs in shutdown status again.

(9) Use the pdconfchk command to check the HiRDB system definitions

```
pdconfchk
```

If errors are detected, correct the HiRDB system definitions, and then execute the `pdconfchk` command again.

(10) Use the pdstart command to start HiRDB normally

```
pdstart
```

(11) Use the pdcopy command to back up all RDAREAs

For details about making backups, see *6.4 Examples of backup*.

(12) Use the pdmod command to re-initialize the RDAREAs on BES1

For details about re-initializing RDAREAs, see *15.4 Increasing the size of an RDAREA or modifying its attributes (RDAREA reinitialization)*.

(13) Use the pdrorg command to reload the table data in the unit that was moved, one RDAREA at a time

Use the unload data file that you created in step (2) as input data. For details about reloading on a per-RDAREA basis, see the manual *HiRDB Version 8 Command Reference*.

(14) Use the pdcopy command to back up all RDAREAs

For details about making backups, see *6.4 Examples of backup*.

(15) Modify client environment definitions

Check if the following operands in the client environment definitions are specified; if the front-end server (FES1) that you moved is specified in these operands, change their values as appropriate:

- `PDFESHOST`

- PDSERVICEGRP
- PDSERVICEPORT

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

11.7 Migrating a HiRDB/Single Server to a HiRDB/Parallel Server

This section explains the procedures for migrating a HiRDB/Single Server to a HiRDB/Parallel Server.

The following items are explained:

- Preparations for migration
- Migration procedure
- Notes on migrating multiple user RDAREAs to different back-end servers

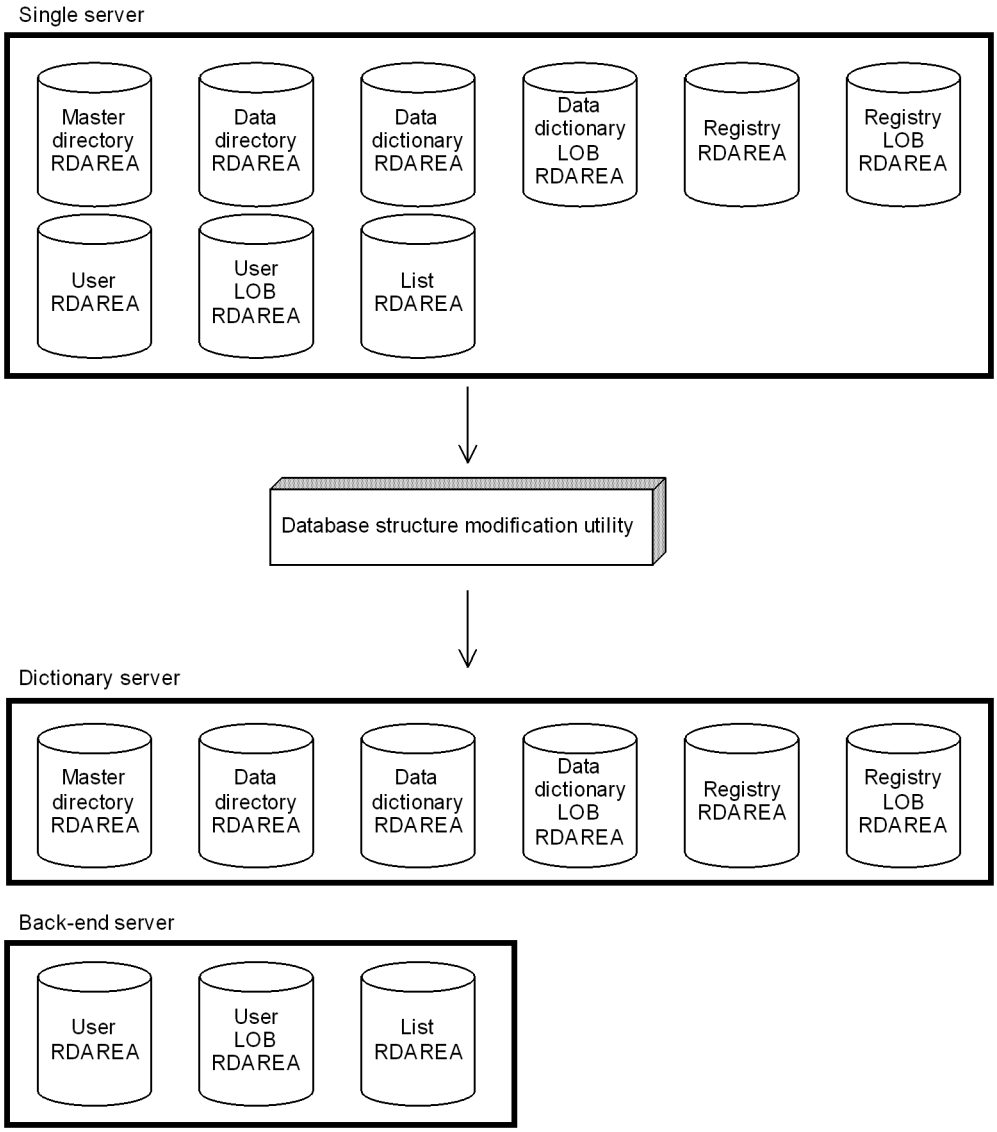
11.7.1 Preparations for migration

(1) Using the database structure modification utility (pdmod command) for migration

The database structure modification utility (`pdmod` command) is used to migrate a HiRDB/Single Server to a HiRDB/Parallel Server. The RDAREAs are migrated from the single server to the dictionary and back-end servers by executing the alter HiRDB mode to parallel statement provided by the database structure modification utility.

Figure 11-1 shows migration of RDAREAs using the database structure modification utility.

Figure 11-1: Migration of RDAREAs using the database structure modification utility



(2) Notes

(a) Backing up data before migration

The `pdfbkup` command should be used to back up the HiRDB files and HiRDB file system areas that contain the following RDAREAs:

- User RDAREAs
- User LOB RDAREAs

(b) Handling of stored procedures and stored functions

When a system is migrated from a HiRDB/Single Server to a HiRDB/Parallel Server, stored procedures and stored functions (stored functions for user-defined functions) created for the HiRDB/Single Server are no longer valid. The stored procedures and stored functions must be re-created with `ALTER ROUTINE` after migration to the HiRDB/Parallel Server has been completed.

(c) Handling of user LOB RDAREAs

User LOB RDAREAs containing LOB data and the user RDAREAs containing the LOB column structure base table for the LOB data must be placed on the same back-end server. Otherwise, the back-end servers may terminate abnormally when the RDAREAs are accessed.

It can be verified that the RDAREAs have been placed on the same back-end server by referencing the `SQL_RDAREA`, `SQL_TABLES`, and `SQL_DIV_COLUMN` dictionary tables after migration has been completed.

(d) Handling of RDAREAs for lists

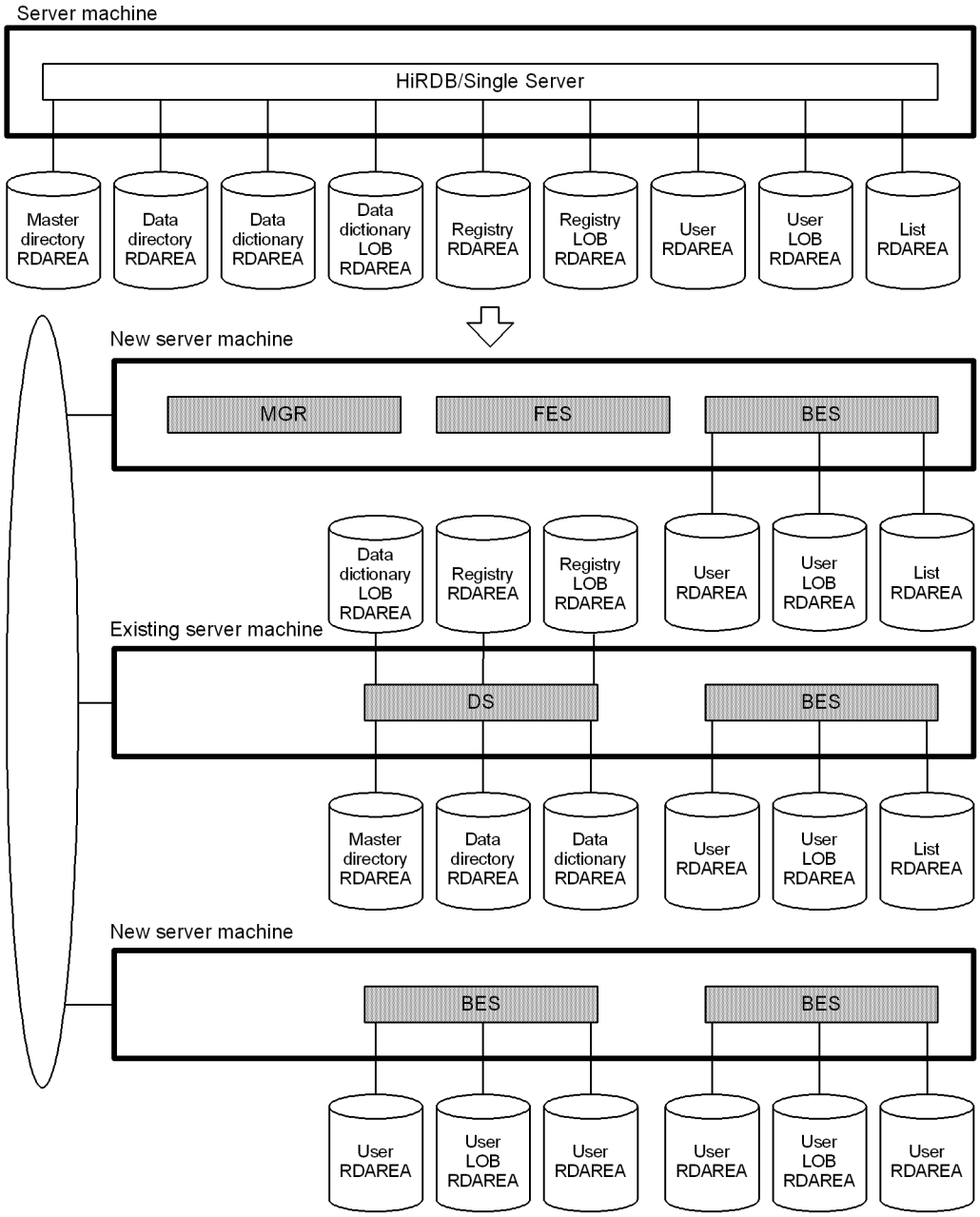
RDAREAs for lists and user LOB RDAREAs containing the base table for the lists must be placed on the same back-end server.

11.7.2 Migration procedure

Executor: HiRDB administrator and superuser

Figure 11-2 shows a model for migration from a HiRDB/Single Server to a HiRDB/Parallel Server.

Figure 11-2: Model for migration from HiRDB/Single Server to HiRDB/Parallel Server



The procedure for migrating from a HiRDB/Single Server to a HiRDB/Parallel Server

is explained below.

(1) Execute the database structure modification utility

The alter HiRDB mode to parallel statement of the database structure modification utility (`pdmod` command) is used to modify the RDAREAs for the HiRDB/Parallel Server. For examples of using this utility, see the manual *HiRDB Version 8 Command Reference*.

(2) Terminate HiRDB normally

The `pdstop` command is entered to terminate the HiRDB/Single Server normally (the termination must be a normal termination).

(3) Set up the HiRDB/Parallel Server environment

The HiRDB/Parallel Server environment is set up. For details on HiRDB/Parallel Server environment setup, see the manual *HiRDB Version 8 Installation and Design Guide*.

The following points should be noted about environment setup:

- A different HiRDB directory must be used than the one for the HiRDB/Single Server.
- The system files for the HiRDB/Single Server must not be used; new system files must be created for the HiRDB/Parallel Server.
- The server machine used by the HiRDB/Single Server (existing server machine) must be defined as the dictionary server.

(4) Migrate the user RDAREAs, user LOB RDAREAs, and list RDAREAs to the other server machine

At this point, the user RDAREAs, user LOB RDAREAs, and list RDAREAs are still at the server machine used for the HiRDB/Single Server (existing server machine). There are two ways to migrate the user RDAREAs, user LOB RDAREAs, and list RDAREAs to the other server machine:

- Using HiRDB operation commands
- Using OS commands

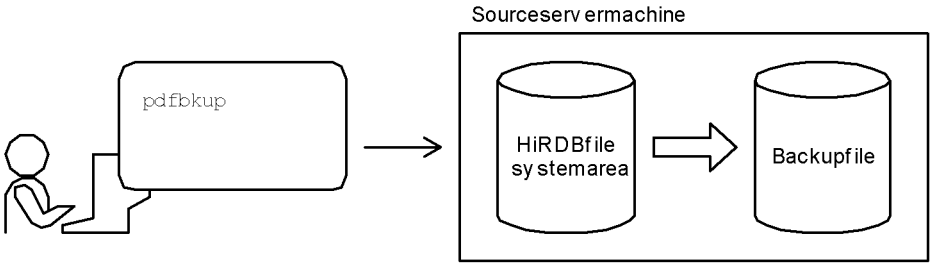
(a) Using HiRDB operation commands

RDAREAs can be migrated in units of HiRDB file system areas or in units of HiRDB files by using HiRDB operation commands.

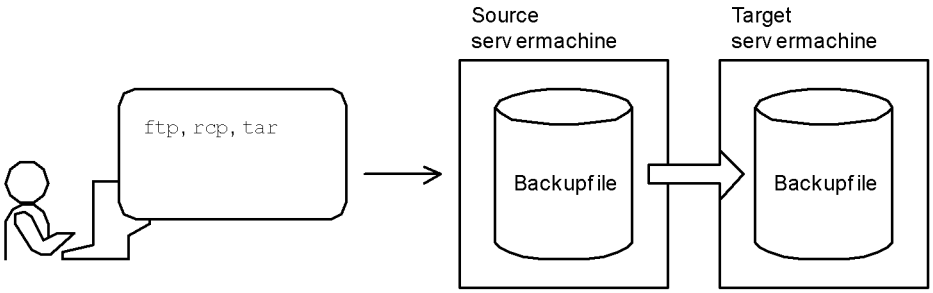
Figure 11-3 shows the procedure for migrating RDAREAs to another server using HiRDB operation commands.

Figure 11-3: Procedure for migrating RDAREAs to another server using HiRDB operation commands

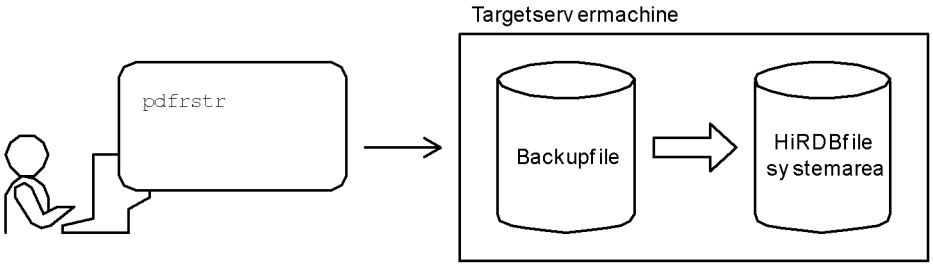
1. Use the pdfbkup command to create a backup of the HiRDB file system area to be moved.



2. Use OS commands (ftp, rcp, tar) to transfer the backup file to the target server machine.



3. Use the pdfrst command to restore the contents of the HiRDB file system area on the target server machine. You can restore as little data as a single HiRDB file at a time.*



* Before performing step 3, confirm that a HiRDB file system area has been created on the target server machine. If it has not, create a HiRDB file system area before performing step 3.

Rules and notes on migration

1. The HiRDB identifier cannot be changed (the value of `pd_system_id` in the system common definition cannot be changed).
2. The HiRDB file system areas must be created in the target server machine under the same names as were used in the source server machine.
3. The lengths of the HiRDB file system areas in the target server machine must be set so as to be equal to or greater than the lengths of the HiRDB file system areas in the source server machine.
4. When a HiRDB file is restored, the HiRDB file system area created in the target server machine must be large enough to store the file.
5. The HiRDB file system areas in the source server machine should not be deleted immediately after migration has been completed. Startup and operation of the HiRDB/Parallel Server should be checked before the HiRDB file system areas are deleted from the source server machine.

(b) Using OS commands

The following OS commands can be used to migrate RDAREAs to another server machine:

- `tar` or `cp` command
- `rcp` command

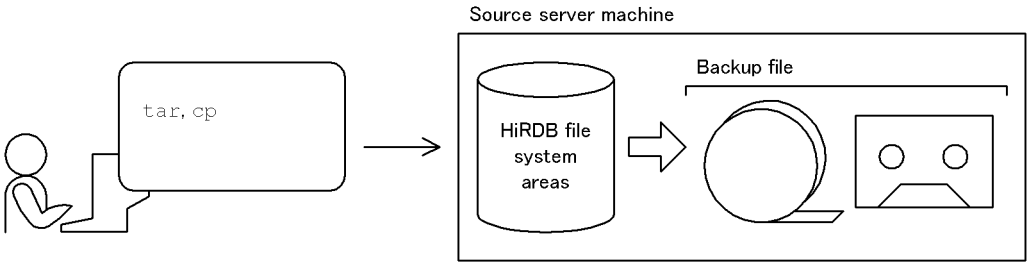
RDAREAs can be migrated in units of HiRDB file system areas by using OS commands.

Using the tar or cp command

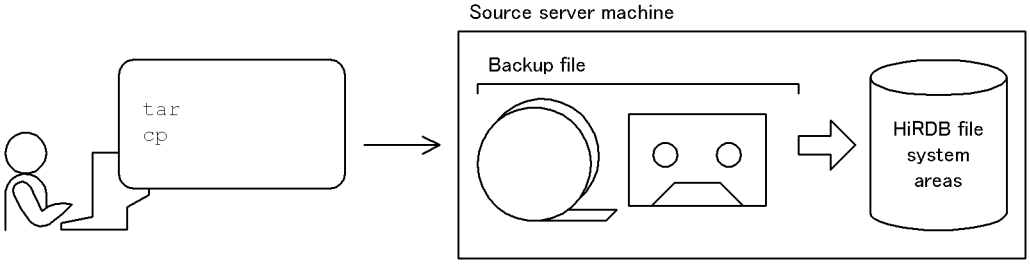
Figure 11-4 shows the procedure for migrating RDAREAs to another server machine using the `tar` or `cp` command.

Figure 11-4: Procedure for migrating RDAREAs to another server machine using the tar or cp command

1. Use the OS commands (tar and cp) to back up the HiRDB file system areas to be migrated. Create the backup file on a medium such as CMT or DAT (magnetic tape, etc.).



2. Use the OS commands (tar and cp) to restore the HiRDB file system areas in the target server machine.



Note: Before executing step 2, a check should be made that the HiRDB file system areas have been created in the target server machine. If they have not been created, they must be created before step 2 is executed.

Notes

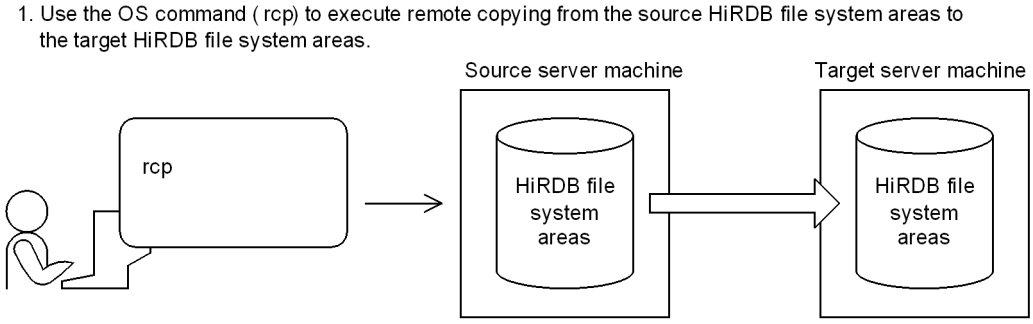
- The HiRDB file system areas must be created in the target server machine under the same names as were used in the source server machine.
- If a file to be migrated is a character special file, the partition size in the target server machine must be set to be equal to or greater than the partition size in the source server machine.
- The HiRDB file system areas in the source server machine should not be deleted immediately after migration has been completed. Startup and operation of the HiRDB/Parallel Server should be checked before the HiRDB file system areas are deleted from the source server machine.

Using the rcp command

Figure 11-5 shows the procedure for migrating RDAREAs to another server

machine using the rcp command.

Figure 11-5: Procedure for migrating RDAREAs to another server machine (using the rcp command)



Note: Before executing step 1, a check should be made that the HiRDB file system areas have been created in the target server machine. If they have not been created, they must be created before step 1 is executed.

Notes

- The HiRDB file system areas must be created in the target server machine under the same names as were used in the source server machine.
- If a file to be migrated is a character special file, the partition size in the target server machine must be set to be equal to or greater than the partition size in the source server machine.
- The HiRDB file system areas in the source server machine should not be deleted immediately after migration has been completed. Startup and operation of the HiRDB/Parallel Server should be checked before the HiRDB file system areas are deleted from the source server machine.

(5) Start HiRDB normally

The pdstart command is entered to start the HiRDB/Parallel Server normally.

(6) Re-create the stored procedures and stored functions

If stored procedures and stored functions (stored functions for user-defined functions) have been created, ALTER ROUTINE must be used to re-create them.

This step is necessary because when a HiRDB/Single Server is migrated to a HiRDB/Parallel Server, the existing stored procedures and stored functions are no longer valid.

11.7.3 Points to be noted about migrating multiple user RDAREAs to different back-end servers

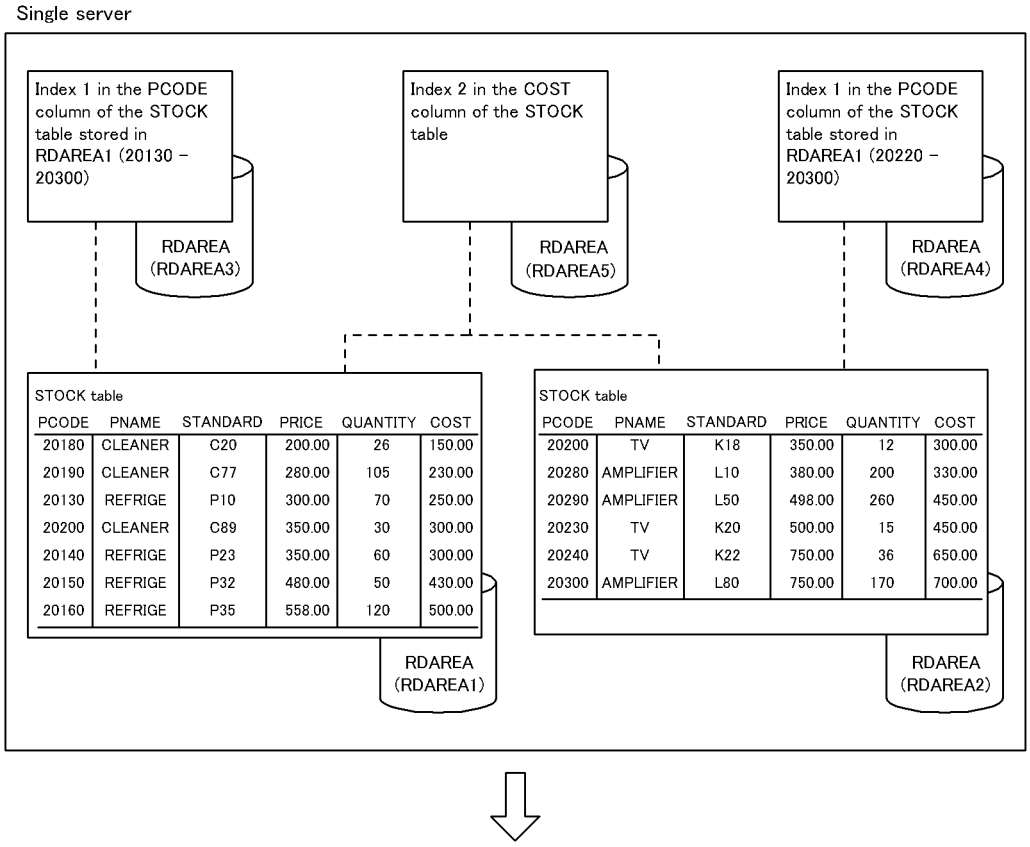
(1) Non-partitioned table

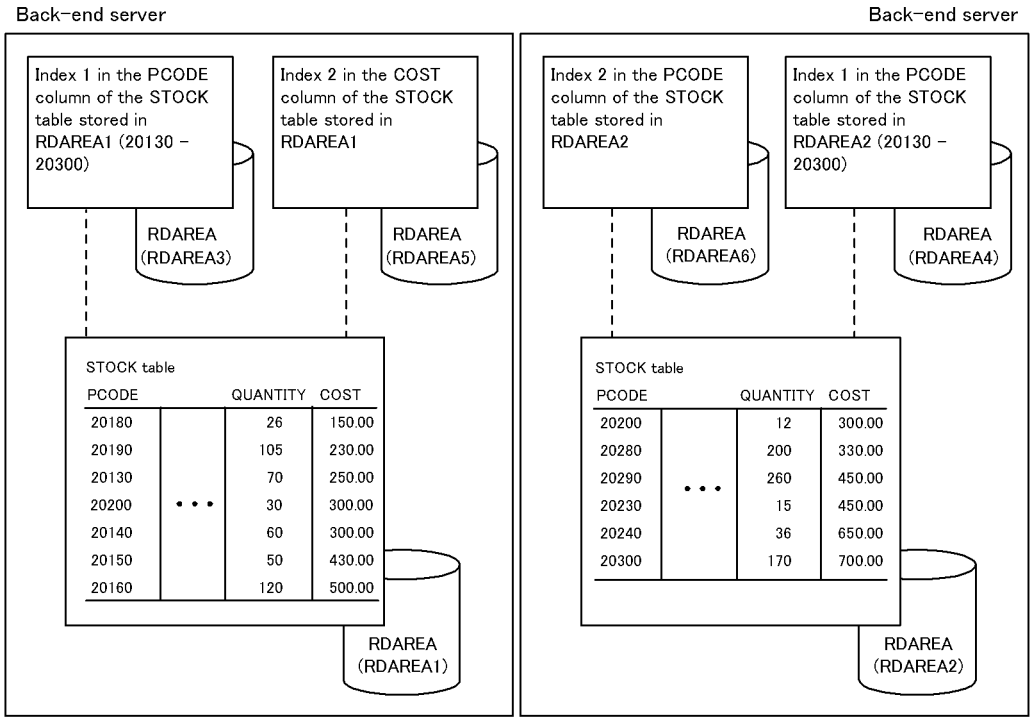
If a table and its index are stored in different RDAREAs, those RDAREAs should be placed in the same back-end server. After executing the database structure modification utility, dictionary tables (`SQL_RDAREAS`, `SQL_TABLES`, and `SQL_INDEXES` tables) should be referenced to check that the RDAREAs have migrated to the same back-end server.

(2) Row-partitioned table

1. If a table and its index are stored in different RDAREAs, those RDAREAs should be placed in the same back-end server. After the database structure modification utility has executed, dictionary tables (`SQL_RDAREAS`, `SQL_TABLES`, `SQL_DIV_TABLE`, `SQL_INDEXES`, and `SQL_DIV_INDEX` tables) should be referenced to check that the RDAREAs have been migrated into the same back-end server.
2. When a non-partitioning index is defined for a table, the table cannot be migrated into multiple back-end servers. To migrate such a table into multiple back-end servers, the non-partitioning index must be deleted before migration; it can then be redefined after the migration. Figure 11-6 shows the procedure for migrating a table for which a non-partitioning index is defined.

Figure 11-6: Procedure for migrating a table with a non-partitioning index defined





Explanation

The STOCK table is row-partitioned into RDAREA1 and RDAREA2. Index 1 is row-partitioned. Index 2 is not row-partitioned. To migrate the STOCK table into multiple back-end servers, delete Index 2, migrate the table, then define Index 2.

11.8 Migrating back-end servers for load balancing

This section explains the procedure for migrating back-end servers in order to balance the workload.

11.8.1 Back-end server load balancing based on a scenario

If, for example, some jobs have peak demand at the end of each month, the workload may become concentrated on specific back-end servers, slowing down the overall job-processing speed. In such a case, you can dynamically modify the positioning of back-end servers according to the fluctuations in their workload, thereby balancing the workload among the individual units.

Note that the back-end server load balancing explained here assumes the following:

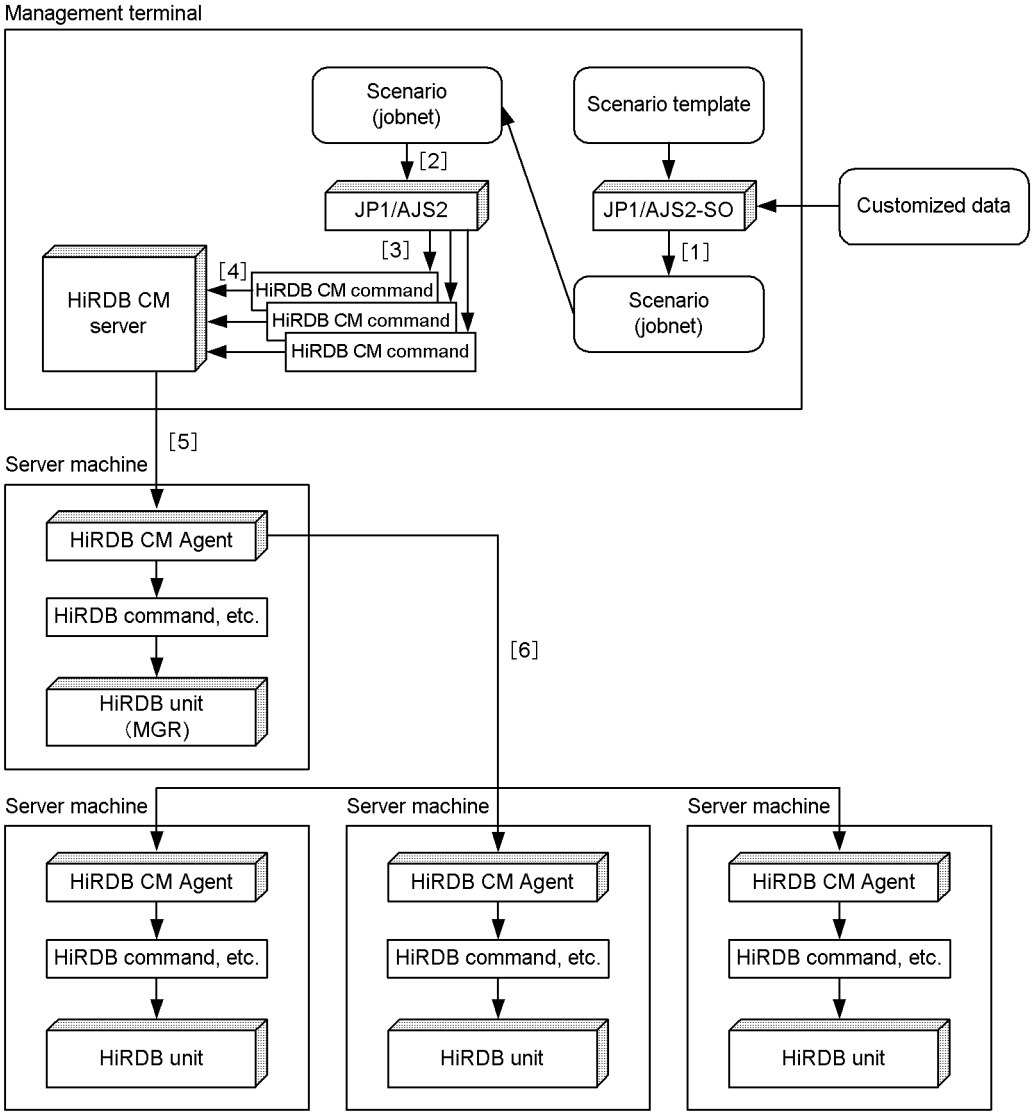
- Use of the standby-less system switchover (effects distributed) facility to migrate back-end servers
- Execution of a scenario created by JP1/AJS2-SO on JP1/AJS2
- Execution of a scenario that uses HiRDB CM

In the case of a HiRDB policy operation, a scenario template is converted to a scenario and executed as shown in Figure 11-7.

Terminology

- Scenario: Object that is formed by a networking scenario job by assigning to it an execution sequence. A scenario is registered into JP1/AJS2 and can then be executed.
- Scenario template: Template for a scenario that consists of procedures only, and in which the environmental information (parameters) necessary for executing the scenario are defined as scenario variables. A scenario template is used as a component for creating a scenario.
- Scenario variable values: User (system)-specific information (including HiRDB identifier, host name, server name, and unit identifier of the migration destination) that is added when a scenario is created from a scenario template.
- Jobnet: Set of jobs to which an execution sequence is assigned by JP1/AJS2. When a jobnet is executed, the jobs constituting the jobnet are executed automatically according to the execution sequence.

Figure 11-7: Using a scenario for back-end server load balancing



Explanation

1. Create a scenario (JP1/AJS2 jobnet) by entering a scenario template and scenario variable values into JP1/AJS2-SO. This step is performed only once the first time the scenario is to be executed.
2. Register the created scenario (jobnet) into JP1/AJS2 and execute it.

3. The jobnet executes HiRDB CM commands.
4. The HiRDB CM commands request processing by the HiRDB CM server.
5. The HiRDB CM server requests processing by the HiRDB CM Agent of the server machine where the system manager is defined.
6. The HiRDB CM Agent of the server machine where the system manager is defined executes HiRDB commands and HA monitor commands, or requests processing by the HiRDB CM Agents of other units.

For details about HiRDB CM, see the *Help* section in HiRDB CM.

11.8.2 Prerequisites and conditions for the target jobs

To balance back-end server workload based on a scenario, the prerequisites and the conditions for the target jobs described below must be satisfied.

Prerequisites

- HiRDB/Parallel Server consisting of multiple units.
- The standby-less system switchover (effects distributed) facility is used (or can be used).

Conditions for the target jobs

- At least two different jobs are executed by a single HiRDB.
- The back-end servers used primarily by the individual jobs (back-end servers that become overloaded) constitute only some of the available back-end servers.
- The back-end servers used primarily by the individual jobs (back-end servers that become overloaded) are mutually exclusive in terms of the processing of the individual jobs.*
- The peak periods (times) of the individual jobs do not overlap.
- The load fluctuations of each job can be predicted or anticipated.
- Transactions of the jobs that use a back-end server to be migrated terminate within short periods (at least there is no transaction that will remain uncompleted for a long time when the back-end server is to be moved).
- While a back-end server is being migrated, transactions that use that back-end server can be queued temporarily or cancelled.

* Because the objective is to balance the load among individual units by migrating back-end servers, the individual jobs need not be completely exclusive.

Reference note:

- A back-end server is migrated when its load is high. Consequently, transactions may be placed on hold temporarily or may be cancelled, thus affecting the jobs adversely.
- The unit to which a back-end server is to be migrated may also have an unexpectedly high processing load. For this reason, the back-end server load may not be balanced even after back-end server migration.

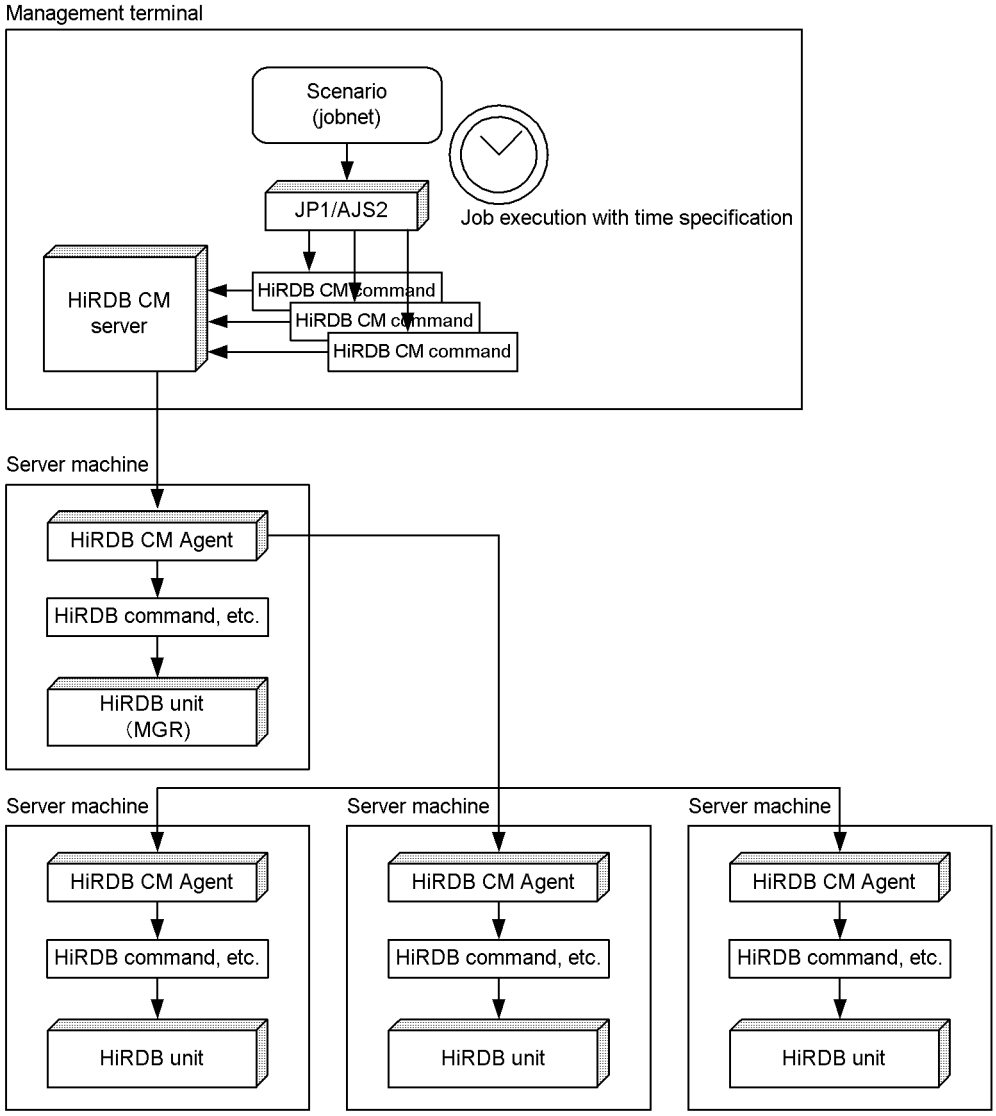
11.8.3 Using a scenario

This section explains how to use a scenario for back-end server load balancing.

(1) Regular execution of a scenario by JP1/AJS2

You can use JP1/AJS2 to execute a scenario by specifying the time of or a time period for back-end server load balancing. Figure 11-8 shows a typical execution of a scenario by JP1/AJS2.

Figure 11-8: Typical execution of a scenario by JP1/AJS2



Characteristics

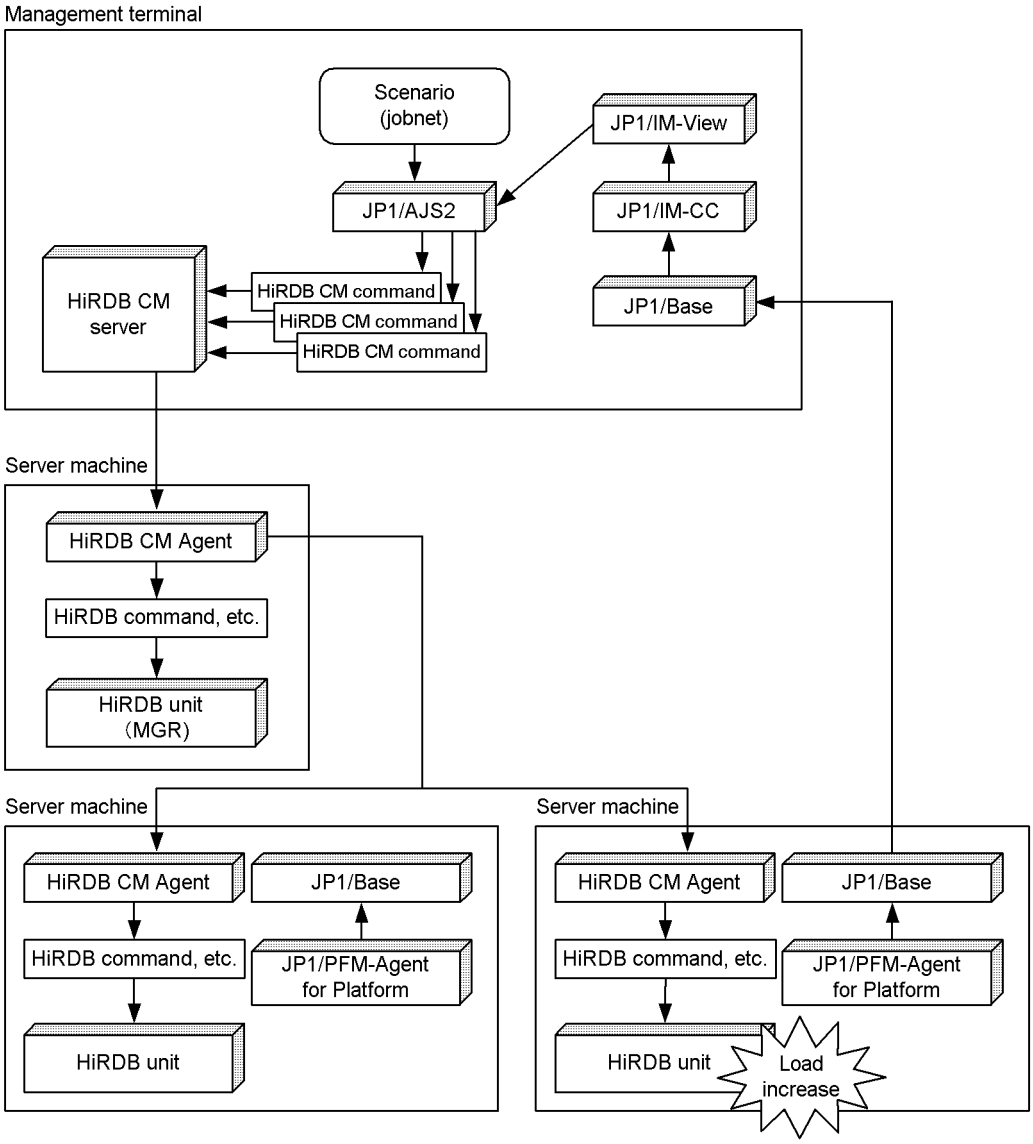
- This method is effective when you can predict the time or time period when the back-end server workload tends to become uneven.
- Because you can specify the time of or time period for migrating a back-end server, it is easy to create an operation plan.

- This method can be used with the minimum product configuration for policy operation.

(2) Monitoring back-end server workload by JP1/PFM and scenario execution based on a user operation

You use JP1/PFM to monitor the workload of back-end servers. When the workload exceeds a preset value, this fact is reported as an event to JP1/IM-CC and displayed in JP1/IM-View. An operator monitors this and uses a GUI operation to execute a scenario. Figure 11-9 shows monitoring of the workload of back-end servers by JP1/PFM and scenario execution based on a user operation.

Figure 11-9: Monitoring the workload of back-end serves by JP1/PFM and scenario execution based on a user operation



Characteristics

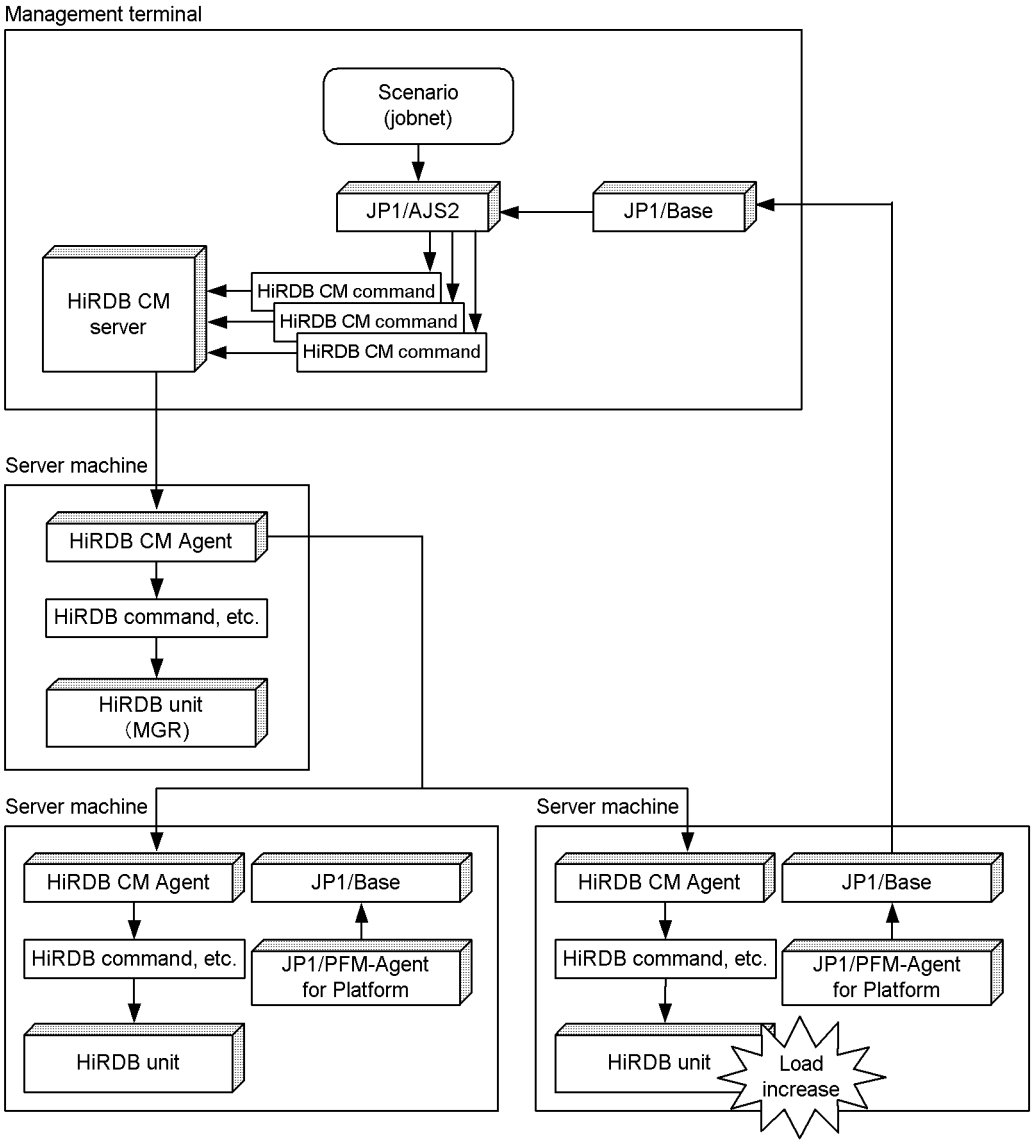
- You can balance back-end server workload dynamically.
- You can change back-end server workload semi-automatically.

- Because a back-end server is migrated after the operator confirms the need for migration, flexible and safe operation is ensured.

(3) Monitoring back-end server workload by JP1/PFM and automatic scenario execution

You use JP1/PFM to monitor the workload of back-end servers. When the workload exceeds a preset value, a scenario is executed automatically by JP1/AJS2. Figure 11-10 shows monitoring of the workload of back-end servers by JP1/PFM and automatic scenario execution.

Figure 11-10: Monitoring of the workload of back-end servers by JP1/PFM and automatic scenario execution



Characteristics

- You can balance back-end server workload dynamically.
- No operator action is required; the back-end server workload is adjusted automatically.

11.8.4 Back-end server configuration examples

This section provides examples of back-end server configurations when load balancing is performed.

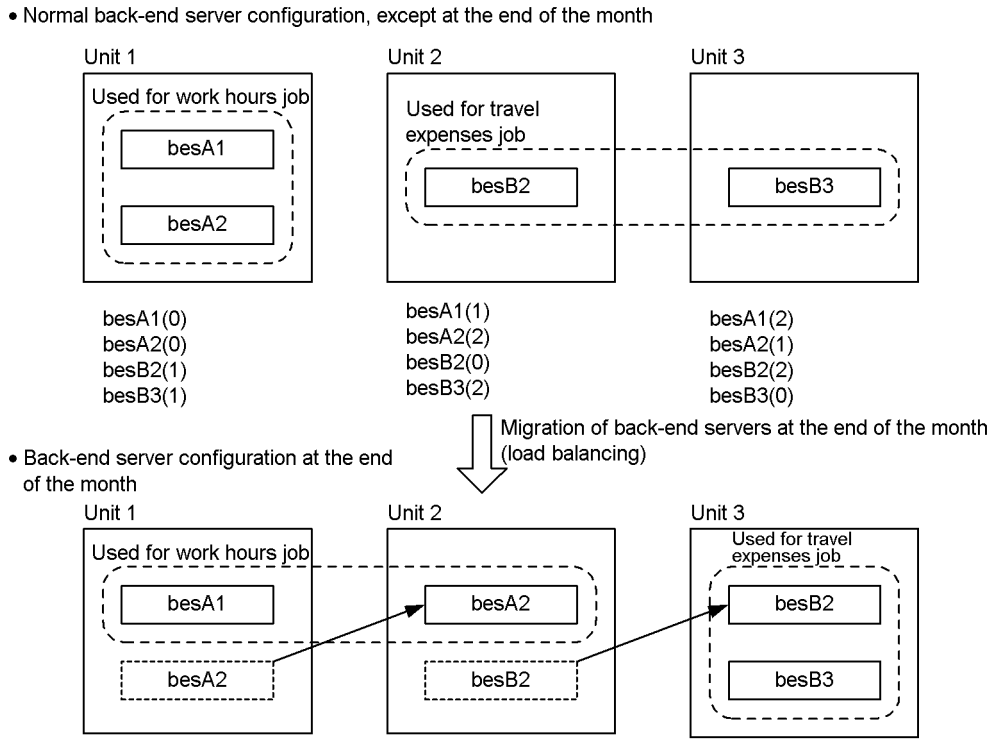
(1) Configuration example 1: Two jobs, three units, four back-end servers

Job assumptions

- The two jobs are a job for keeping track of hours worked and a job for settling business travel expenses.
- Two back-end servers are allocated to each job and are used exclusively for that job.
- During normal operations, the travel expenses job requires higher performance.
- At the end of the month, the workload of the work hours job increases.
- At the end of the month, the work hours job is given higher priority than the travel expenses job.

Figure 11-11 shows the back-end server configuration when load balancing is performed.

Figure 11-11: Back-end server configuration when load balancing is performed for Example 1 (two jobs, three units, four back-end servers)



Note

The number in parentheses indicates the switching priority of each back-end server within the unit. This is the same as the value of the `standbypri` operand of the standby server (`standby` is specified in the `inital` operand) for each server specified in the `servers` definition of the HA monitor for each server machine.

Note that (0) indicates the primary system. In this case, `online` must be specified in the `inital` operand of the `servers` definition for the applicable server machine or server and the `standbypri` operand must not be specified.

Explanation

- At the end of the month when the workload of the work hours job increases, back-end servers are migrated to balance the job load.
- This scenario is achieved by combining two basic scenarios for migrating back-end servers.

- When the priority order shown in the figure is specified, you can prevent the load from becoming unbalanced among the back-end servers even if an error occurs after back-end server migration.

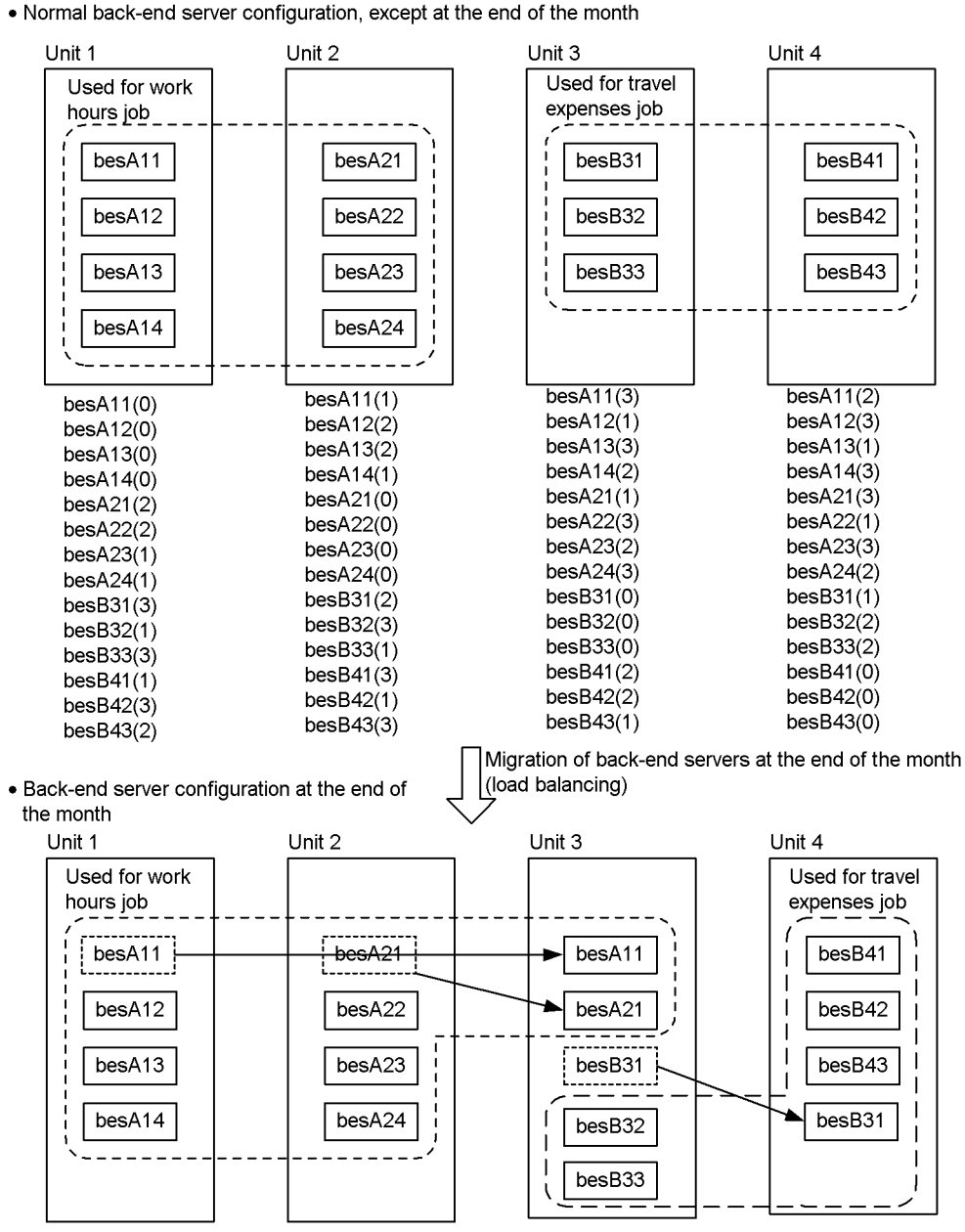
(2) Configuration example 2: Two jobs, four units, 14 back-end servers

Job assumptions

- The two jobs are a job for keeping track of hours worked and a job for settling business travel expenses.
- Eight back-end servers are allocated to the work hours job and six back-end servers are allocated to the travel expenses job, and all back-end servers are used exclusively for their allocated job.
- During normal operations, the two jobs require the same level of performance.
- At the end of the month, the workload of the work hours job increases.
- Even at the end of the month, the performance of the travel expenses job must not be slowed down if at all possible.

Figure 11-12 shows the back-end server configuration when load balancing is performed.

Figure 11-12: Back-end server configuration when load balancing is performed for Example 2 (two jobs, four units, and 14 back-end servers)



Note

The number in parentheses indicates the switching priority of each back-end server within the unit. This is the same as the value of the `standbypri` operand of the standby server (`standby` is specified in the `inital` operand) for each server specified in the `servers` definition of the HA monitor for each server machine.

Note that (0) indicates the primary system. In this case, `online` must be specified in the `inital` operand of the `servers` definition for the applicable server machine or server and the `standbypri` operand must not be specified.

Explanation

- At the end of the month when the workload of the work hours job increases, back-end servers are migrated to balance the job load.
- This scenario is achieved by combining three basic scenarios for migrating back-end servers.

11.8.5 Preparations related to HiRDB

The HiRDB-related preparations explained in this subsection are required.

(1) Applying the standby-less system switchover (effects distributed) facility

You must apply the standby-less system switchover (effects distributed) facility. For details about this facility, see *25. Using the System Switchover Facility*.

(2) Selecting whether to apply the transaction queuing facility

Select whether or not the transaction queuing facility is to be applied. Guidelines on applying this facility are provided below. For details about the transaction queuing facility, see *25.19 Transaction queuing facility*.

(a) Processing method when the transaction queuing facility is used

The processing method when the transaction queuing facility is used is explained below.

Processing method

1. When a back-end server is migrated, startups of new transactions for that back-end server are put on hold.
2. Waits for termination of transactions being processed by the back-end server that is to be migrated. When there are no more transactions being processed by the back-end server to be migrated, the back-end server is migrated.
3. Transactions put on hold in 1 are started.

Note that you can specify the transaction queuing wait time in the `pd_ha_trn_queuing_wait_time` operand. If termination of a back-end server is

delayed for a considerable period because of a transaction that requires a long time to execute, transaction queuing is cancelled when the transaction queuing wait time is exceeded and migration of the back-end server is cancelled. In such a case, the transactions that were placed on hold in 1 are restarted.

(b) Processing method when the transaction queuing facility is not used

A back-end server is migrated without placing startups of new transactions on hold and without waiting for transactions currently executing to terminate.

(c) Advantages and disadvantages of the transaction queuing facility

Table 11-1 shows the advantages and disadvantages of the transaction queuing facility.

Table 11-1: Advantages and disadvantages of the transaction queuing facility

Application of the transaction queuing facility	Advantages	Disadvantages
Applied	Transaction errors can be avoided during back-end server migration (except when back-end server startup takes a long time).	• Transactions can become queued during back-end server migration. • Back-end server migration can take longer than when the transaction queuing facility is not applied. • Back-end server migration may be cancelled if there is a transaction that executes over an extended period.
Not applied	• Transactions are not queued during back-end server migration. • A back-end server can be migrated even when there is a transaction that executes over an extended period. • Back-end server migration takes less time than when the transaction queuing facility is applied.	• A transaction that is executing at the time of back-end server migration terminates with an error. • A transaction that starts during back-end server migration terminates with an error.

Reference note:

Because the transaction queuing facility is also used for the system switchover at the time of system failure, you may want to consider using this facility together with the system switchover facility.

(d) Setting up the transaction queuing facility

To use the transaction queuing facility, you must specify `queuing` in the `pd_ha_transaction` operand.

You must also specify the sum of the times described below as the transaction queuing wait time in the `pd_ha_trn_queuing_wait_time` operand.

- Maximum queuing time for transactions that occur during back-end server migration
- Time in the back-end server migration scenario between forced server termination (which basically is the time at which the KFPS01843-I message is output when the pdstop -s -f command is entered) and completion of server startup (issuance of the KFPS01813-I message)

If the transaction queuing facility is not used, specify error in the pd_ha_transaction operand or omit this operand.

11.8.6 Back-end server load balancing performed by the user

You can use commands to balance the workload of the back-end servers.

(1) Operating procedure

The operating procedure depends on whether an HA monitor or Hitachi HA Toolkit Extension is used. This subsection describes the operating procedure. If you are moving multiple back-end servers, see *(3)(a) Automating the moving of back-end servers*.

(a) When an HA monitor is used

1. Check the status of the server to be moved.

   At the unit where the system manager is located, execute the pdls -d ha -s *server-name* -a command and check the displayed information (standard output) for the following:

   - The back-end server to be moved is active (ONL) at the source unit.
   - The back-end server to be moved is in standby (SBY) or inactive (STP) status at the target unit.

2. Start the transaction queuing facility at the back-end server that is to be moved.

   Start the transaction queuing facility by executing the pdtrnqing -s *server-name* command at the back-end server's source unit. There is no need to perform this operation if queuing was not specified in the pd_ha_transaction operand.

   Make sure that the command terminates normally with return code 0 or with warning with return code 4. Return code 0 means that the transaction queuing facility has started successfully. Return code 4 means that the pd_ha_transaction operand specification resulted in an error. If the return code is 8 or 12, terminate the procedure because a failure has occurred.

3. Forcibly terminate the back-end server that is to be moved.

   Forcibly terminate the back-end server by executing the pdstop -f -s *server-name* command at the unit containing the system manager. Make sure that

the command's return code is `0`. If the return code is not `0`, terminate the procedure because a failure has occurred.

4. Start the back-end server being moved at the target unit.

   At the unit containing the system manager, execute the `pdstart -s` *server-name* `-u` *target-unit-identifier* command. Make sure that the command's return code is `0`. If the return code is not `0`, terminate the procedure because a failure has occurred.

5. Check the status of the back-end server being moved.

   At the unit containing the system manager, execute the `pdls -d ha -s` *server-name* `-a` command to check the displayed information (standard output) and return code for the following:

   - Check that the back-end server is waiting for startup of the running system (`WIT`) in the target unit. If the back-end server is waiting for startup of the running system (`WIT`) in the target unit, proceed to step 6.
   - Check that the back-end server is in active status (`ONL`) at the target unit. Proceed to step 7 only if the back-end server is in active status (`ONL`) at the target unit.
   - If neither of the above is true, cancel the procedure because a failure has occurred.

6. Resume startup of the back-end server at the target unit.

   At the target unit, the super user must execute the `monact` *server-name* command and make sure that the command's return code is `0`. If the return code is not `0`, terminate the procedure because a failure has occurred.

7. Release the transaction queuing facility at the back-end server being moved.

   At the back-end server's target unit, execute the `pdtrnqing -d -s` *server-name* command to release the transaction queuing facility. If `queuing` was omitted in the `pd_ha_transaction` operand, there is no need to perform this operation.

   Check that the command's return code is `0`. If the return code is not `0`, terminate the procedure because a failure has occurred.

(b) When Hitachi HA Toolkit Extension is used

1. Check the status of the server to be moved.

   See 1. in *(a) When an HA monitor is used.*

2. Start the transaction queuing facility at the back-end server that is to be moved.

   See 2. in *(a) When an HA monitor is used.*

3. Use the cluster software system switchover command to move the back-end

server.

Use the cluster software system switchover command to switch over to the system at the target unit.

4. Check the status of the back-end server.

At the unit containing the system manager, execute the `pdls -d ha -s` *server-name* `-a` command to check the displayed information (standard output) and the return code for the following:

- The back-end server is active (`ONL`) at the target unit.

5. Release the transaction queuing facility at the back-end server being moved.

See 7. in *(a) When an HA monitor is used.*

(2) Error handling procedure

If an error occurs while the back-end server is being moved, check the syslogfile and the messages in the standard output and standard error output, and eliminate the cause of the error.

If a failure occurs after the transaction queuing facility has started, execute the following command at the unit containing the system manager in order to release the transaction queuing facility:

```
pdtrnqing -d -f -s server-name
```

When this command is executed, the transaction queuing facility is released. Because this action is a protective measure against illegal operation of the transaction queuing facility as a result of the failure, a processing request occurs for the back-end server that has not been started, which results in an SQL error.

(3) Notes

(a) Automating the moving of back-end servers

To use the transaction queuing facility to move multiple back-end servers, you can automate the procedure as follows:

1. Start the transaction queuing facility
2. Move the back-end servers
3. Release the transaction queuing facility

If you attempt to do this manually, the processing of moving the back-end servers may take too much time and a timeout may occur in the transaction queuing facility, resulting in a transaction error.

(b) Moving multiple back-end servers

If you use the transaction queuing facility to move multiple back-end servers, execute *(1) Operating procedure* for each server. Do not execute this procedure for multiple servers at the same time. If multiple back-end servers are moved concurrently, transactions accessing those back-end servers are queued and they may not be able to terminate. In such a case, the `pdtrnqing` command may result in an error.

Chapter

12. Migrating Resources Between Systems

This chapter explains the procedures for migrating tables and stored procedures to another HiRDB system.

This chapter contains the following sections:

12.1 Migrating a table to another HiRDB system

Executor: HiRDB administrator

The dictionary import/export utility and the database reorganization utility enable a table that is currently in use to be migrated to another HiRDB system. This facility can be used to achieve the following:

- Migrating a database from a test system to the actual system
- Migrating a database created in one HiRDB system to another HiRDB system
- During system reconstruction, storing existing database information and restoring it after the system has been reconstructed

12.1.1 Migrating a table to another HiRDB system

(1) Limitations

- A table for which an abstract data type is defined cannot be migrated to another HiRDB system. Some abstract data types provided by a plug-in can be migrated to another HiRDB system, depending on the plug-in type. To determine whether or not an abstract data type can be migrated, refer to the applicable plug-in manual. To migrate an abstract data type, the same plug-in as is used at the source HiRDB system is required at the target HiRDB system.
- If the character codes used differ between the source system and the target system, you will not be able to use the database reorganization utility to reload table data on the target system. In such a case, instead of reloading, you must use the database load utility to load the data.
- A falsification prevented table cannot be moved to another HiRDB system, nor can its table data be reloaded on the target system.
- You cannot use the database copy utility and database recovery utility to migrate data. Do not attempt to migrate data by restoring a backup made on the migration-source system to the migration-destination system.

(2) Information to be migrated

The following information is migrated:

- Table definition information
- Table data

(a) Procedure for migrating table definition information

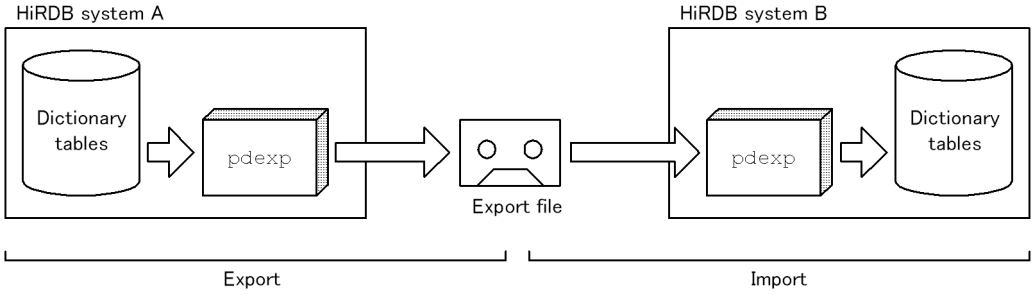
The dictionary import/export utility (`pdexp` command) is used to migrate table definition information in dictionary tables currently in use to another HiRDB system.

The following table definition information is subject to migration:

- Base table definition (applicable only for tables that do not contain abstract data types)
- View table definitions
- Index definitions
- Comment information
- Table alias definitions

Figure 12-1 shows the procedure for migrating table definition information with the dictionary import/export utility.

Figure 12-1: Migration of table definition information with the dictionary import/export utility

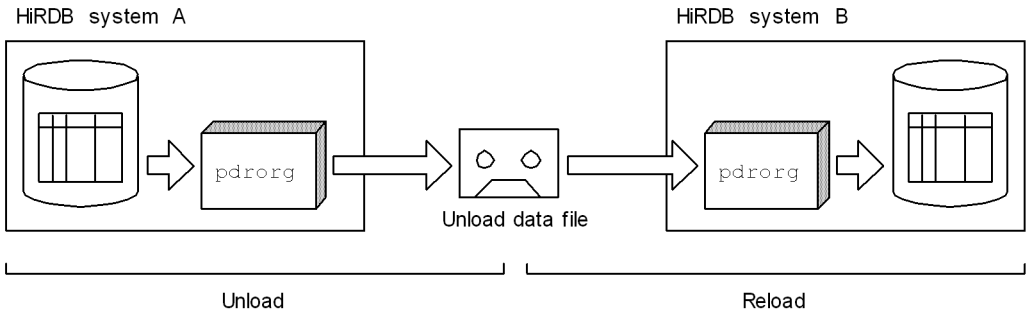


Note: The table definition information remains intact in HiRDB system A.

(b) Procedure for migrating table data

The database reorganization utility (`pdrorg` command) is used to migrate table data that is currently in use to another HiRDB system. Figure 12-2 shows the procedure for migrating table data with the database reorganization utility.

Figure 12-2: Migration of table data with the database reorganization utility



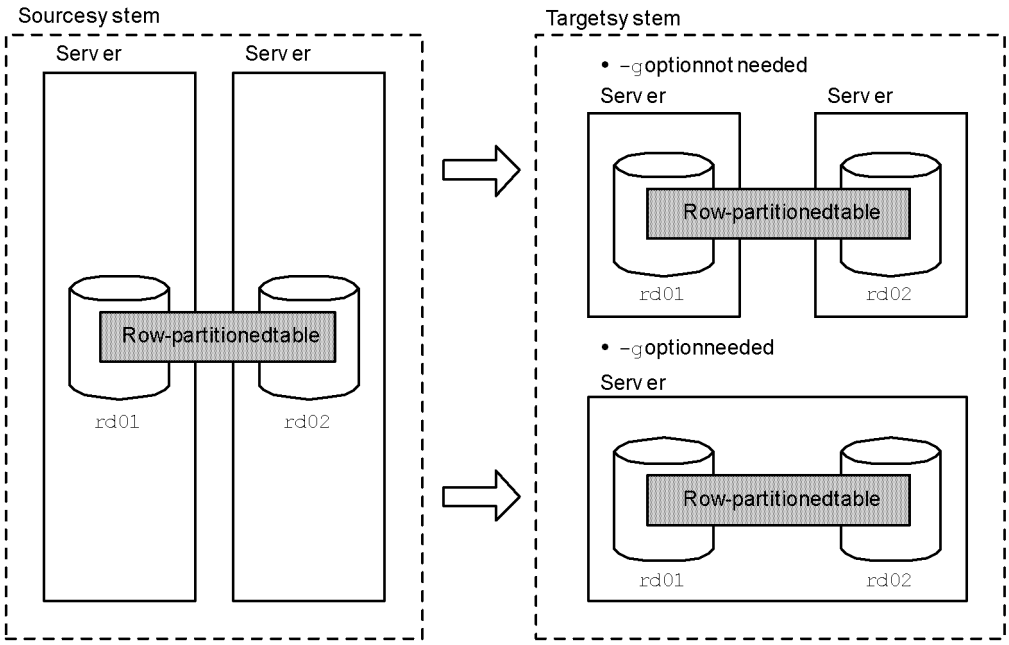
Note 1: Table data remains unchanged in HiRDB system A.
Note 2: In the following cases, the `-j` option must be specified together with the `pdorg` command:

- A LOB column is defined in the table to be moved.
- An abstract data type with the LOB attribute is defined in the table to be moved.

(3) Notes on RDAREAs (Important)

1. Before table definition information is migrated, RDAREAs with the same names as in the source system must be created in the target system.
2. If there are differences between the migration source system and the migration target system in the names of the RDAREAs for storing the table to be migrated or in the configurations of the servers for storing the RDAREAs, the `-g` option must be specified when the database reorganization utility (`pdrorg` command) is executed. Figure 12-3 shows an example of system configurations for which the `-g` option must be specified.

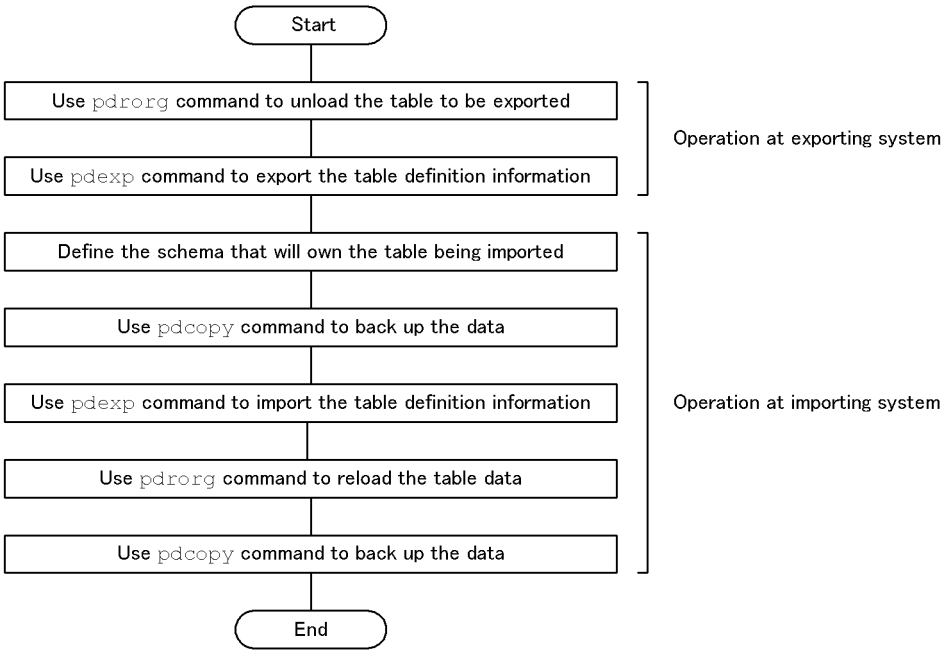
Figure 12-3: Example of system configuration requiring the -g option



(4) Migration procedure

Figure 12-4 provides an overview of the procedure for migrating a table to another HiRDB system.

Figure 12-4: Procedure for migrating a table to another HiRDB system

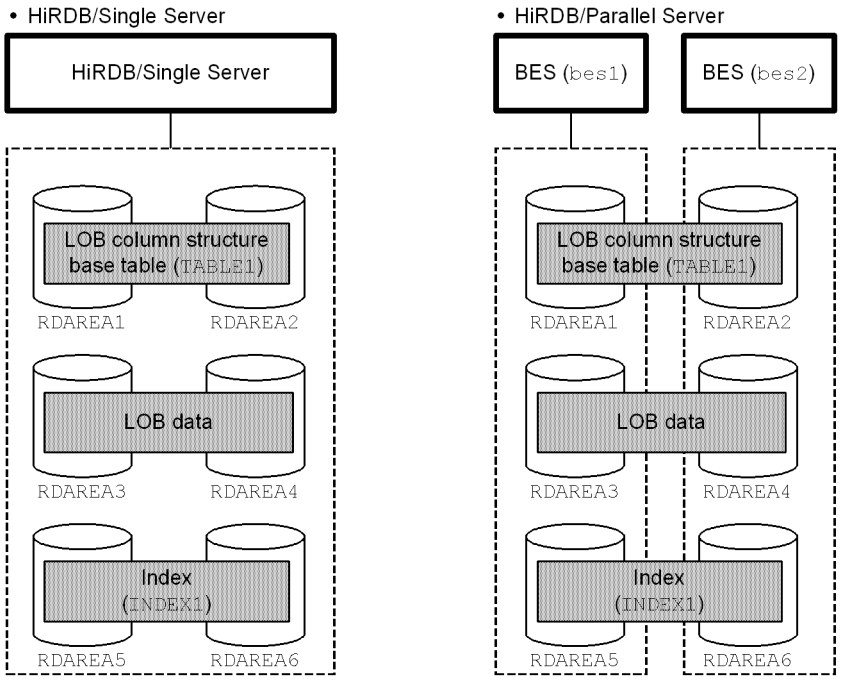


12.1.2 Example 1: Migrating a table

This example migrates a table (TABLE1) from HiRDB system A to HiRDB system B.

- A LOB column is defined for TABLE1.
- An index (INDEX1) is defined for TABLE1.

It is assumed that RDAREAs with the same names (RDAREA1 to RDAREA6) have already been created in the target system.



(1) Enter the pdhold command to shut down RDAREA1 to RDAREA6

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (/pdrorg/rorg01):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1
```

Explanation

Specifies the name of the unload data file.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1
```

Explanation

Specifies the name of the unload data file for `bes1`.

(3) Use the pdrorg command to unload data for TABLE1

```
pdrorg -k unld -j -g -t TABLE1 /pdrorg/rorg01
```

Explanation

`-k`: Specifies `unld` in order to unload data.

`-j`: Specifies that the table that is to be unloaded is one of the following:

- A table containing a LOB column
- A table in which an abstract data type with the LOB attribute is defined

`-g`: Specifies that `TABLE1` is a row-partitioned table on a server of the HiRDB/Parallel Server. A single unload data file is created.

`-t`: Specifies the name of the table that is to be unloaded.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (2).

(4) Use the pdrels command to release RDAREA1 to RDAREA6 from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(5) Create the control statements file for the pdexp command

The following are the contents of the control statements file (`/pdexp/exp01`):

```
-t USR01.TABLE1
```

Explanation

`USR01`: Name of the schema that owns `TABLE1`.

`TABLE1`: Name of the table to be exported.

(6) Use the pdexp command to export table definition information for TABLE1

```
pdexp -e /pdexp/expfile1 -f /pdexp/exp01
```

Explanation

`-e`: Specifies the name of the file to be exported.

`-f`: Specifies the name of the control statements file for the `pdexp` command created in step (5).

(7) Store the unload data file and export file on a medium such as CMT

The unload data file created in step (3) and the export file created in step (6) are stored on a medium such as CMT.

The operations at the source system (HiRDB system A) are now complete.

(8) Store the unload data file and export file in the target system

The operations from this point on are performed at the target system (HiRDB system B).

The unload data file and export file stored on a medium such as CMT are copied into the target system.

(9) Use the pddef command to define a schema for the user who owns TABLE1

```
pddef
   CREATE SCHEMA AUTHORIZATION USR01;
```

(10) Use pdcopy command to back up the data

```
pdcopy -m /rdarea/mast/mast01 -M r
-r RDMAST,RDDIR,RDDIC,RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
-b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

The RDAREAs are backed up as a safeguard in the event of errors during database migration. The following RDAREAs must be backed up:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs
- User RDAREAs and user LOB RDAREAs for storing the imported table (`RDAREA1` to `RDAREA6`)

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

-r: Specifies the names of the RDAREAs to be backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination for the pdcopy command's processing results listing.

(11) Use the pdexp command to import table definition information for TABLE1

```
pdexp -i /pdexp/expfile1
```

Explanation

-i: Specifies the name of the file to be imported.

(12) Use the pdhold command to shut down RDAREA1 to RDAREA6

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(13) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (/pdrorg/rorg01):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which to create an index information file. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1               1
idxwork bes1 /pdrorg/idxwork              2
sort bes1 /sortwork,8192                  3
idxwork bes2 /pdrorg/idxwork              4
sort bes2 /sortwork,8192                  5
```

Explanation

1. Specifies the name of the unload data file (for `bes1`).
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).
4. Specifies the name of the directory in which an index information file is to be created (for `bes2`). The index information file is created under this directory.
5. Specifies the name of the work directory for sorting (for `bes2`).

(14) Use the pdrorg command to reload data for TABLE1

```
pdrorg -k reld -j -g -t TABLE1 /pdrorg/rorg01
```

Explanation

To re-create the index (`INDEX1`) at the same time, the `-i` option is omitted and the index batch creation mode is used.

`-k`: Specifies `reld` in order to reload data.

`-j`: Specifies that the table that is to be reloaded is one of the following:

- A table containing a LOB column
- A table in which an abstract data type with the LOB attribute is defined

`-g`: Specifies that `TABLE1` is a row-partitioned table on a server of the HiRDB/Parallel Server. A single unload data file is created.

`-t`: Specifies the name of the table to be reloaded.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (13).

(15) Use the pdcopy command to back up the data

```
pdcopy -m /rdarea/mast/mast01 -M r
-r RDMAST,RDDIR,RDDIC,RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
-b /pdcopy/backup02 -p /pdcopy/list02
```

Explanation

The following RDAREAs must be backed up:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs
- User RDAREAs and user LOB RDAREAs for storing the imported table (RDAREA1 to RDAREA6)

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-r: Specifies the names of the RDAREAs to be backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

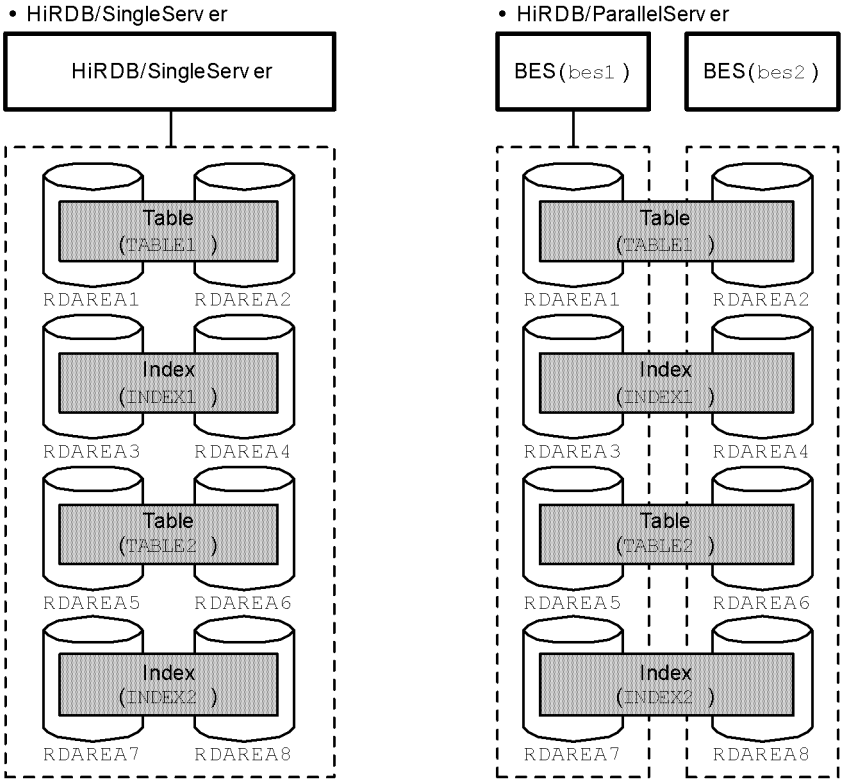
(16) Use the pdrels command to release RDAREA1 to RDAREA6 from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

12.1.3 Example 2: Migrating tables of a schema

This example migrates to HiRDB system B all tables in HiRDB system A owned by the user with authorization identifier USR01. It is assumed that RDAREAs with the same names have been created in the migration target system. Unloading and reloading of the tables is performed by schema (all tables in the specified schema are unloaded and reloaded).



(1) Use the pdhold command to shut down RDAREAs storing data to be migrated

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,...
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (`/pdrorg/rorg01`):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1
```

Explanation

Specifies the name of the unload data file.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1
```

Explanation

Specifies the name of the unload data file. Because the `-g` option is assumed, the unload data file is created at one location.

(3) Use the pdrorg command to unload tables by schema

```
pdrorg -k unld -t USR01.all /pdrorg/rorg01
```

Explanation

`-k`: Specifies `unld` for unloading.

`-t`: Specifies the authorization identifier of the schema whose tables are to be unloaded.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (2).

Remarks

When tables are unloaded by schema, the `-j` option (unloading with LOB data present) is assumed. In the case of a HiRDB/Parallel Server, the `-g` option (unload data file standardization) is also assumed.

(4) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,...
```

(5) Create the control statements file for the pdexp command

The following are the contents of the control statements file (`/pdexp/exp01`):

```
-t USR01.TABLE1
-t USR01.TABLE2
```

Explanation

`USR01`: Name of the schema to which `TABLE1` and `TABLE2` belong.

`TABLE1`, `TABLE2`: Names of the tables being exported.

(6) Use the pdexp command to export table definition information on all tables

```
pdexp -e /pdexp/expfile1 -f /pdexp/exp01
```

Explanation

`-e`: Specifies the name of the file that is to be exported.

`-f`: Specifies the name of the control statements file for the `pdexp` command that was created in step (5).

(7) Store the unload data file and export file on a medium such as CMT

The unload data file created in step (3) and the export file created in step (6) are stored on a medium such as CMT.

Operations at the migration source system (HiRDB system A) have now been completed.

(8) Store the unload data file and export file at the migration target system

The operations from here on are performed at the migration target system (HiRDB system B).

The unload data file and export file that have been stored on a medium such as CMT are now stored at the migration target system.

(9) Use the pddef command to define a schema (authorization identifier: USR01)

This operation is not necessary if the tables are being migrated to a different schema (a schema other than `USR01`).

```
pddef
   CREATE SCHEMA AUTHORIZATION USR01;
```

(10) Use the pdcopy command to back up the data

```
pdcopy -m /rdarea/mast/mast01 -M r
-r RDMAST,RDDIR,RDDIC,RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
-b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

A backup is made as a safeguard in the event errors occur during database migration. The following RDAREAs are backed up:

- Master directory RDAREA

- Data directory RDAREA
- Data dictionary RDAREA
- User RDAREAs and user LOB RDAREAs that store the tables being imported (RDAREA1-RDAREA6)

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-r: Specifies the names of the RDAREAs to be backed up.

-b: Specifies a name for the backup file.

-p: Specifies the output destination for the pdcopy command's processing results listing.

(11) Use the pdexp command to import table definition information

```
pdexp -i /pdexp/expfile1
```

Explanation

-i: Specifies the name of the export file.

(12) Use the pdhold command to shut down RDAREA1-RDAREA6

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,...
```

(13) Create the control statements file for the pdrorg command

Shown below are the contents of the control statements file (/pdrorg/rorg01).

If the tables are to be migrated to a different schema (a schema other than USR01), the contents of the control statements file will be different. For an example of a control statements file when tables are migrated to a different schema, see *12.1.4 Example of a control statements file when migrating tables to a different schema*.

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.

2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1                    1
idxwork bes1 /pdrorg/idxwork                   2
sort bes1 /sortwork,8192                       3
idxwork bes2 /pdrorg/idxwork                   4
sort bes2 /sortwork,8192                       5
```

Explanation

1. Specifies the name of the unload data file. Because the `-g` option is assumed, the unload data file is created at one location.
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).
4. Specifies the name of the directory in which an index information file is to be created (for `bes2`). The index information file is created under this directory.
5. Specifies the name of the work directory for sorting (for `bes2`).

(14) Use the pdrorg command to reload all tables

```
pdrorg -k reld -t USR01.all /pdrorg/rorg01
```

Explanation

Because an index (`INDEX1`) is re-created simultaneously, the `-i` option is omitted and indexes are created in the batch index creation mode.

`-k`: Specifies `reld` for reloading.

`-t`: Specifies the authorization identifier of the schema to be reloaded. If the tables are to be migrated to a different schema (a schema other than `USR01`), specify the authorization identifier at the migration target. For example, if the tables were to be migrated to authorization identifier `USR02`, the specification would be `-t USR02.all`.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (13).

Remarks

When tables are unloaded by schema, the `-j` option (unloading with LOB data present) is assumed. In the case of a HiRDB/Parallel Server, the `-g` option (unload data file standardization) is also assumed.

(15) Use the pdcopy command to back up the data

```
pdcopy -m /rdarea/mast/mast01 -M r
-r RDMAST,RDDIR,RDDIC,RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
-b /pdcopy/backup02 -p /pdcopy/list02
```

Explanation

The following RDAREAs are backed up:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREA
- User RDAREAs and user LOB RDAREAs that store the tables being imported (`RDAREA10-RDAREA6`)

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-r`: Specifies the names of the RDAREAs to be backed up.

`-b`: Specifies a name for the backup file.

`-p`: Specifies the output destination for the `pdcopy` command's processing results listing.

(16) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

12.1.4 Example of a control statements file when migrating tables to a different schema

An example of a control statements file of the database reorganization utility for migrating tables to a different schema is shown below. The authorization identifier of

the migration source schema is USR01 and that of the migration target schema is USR02.

(1) HiRDB/Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
tblname USR01                             4
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.
4. Specifies the authorization identifier of the migration source schema.

(2) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1               1
idxwork bes1 /pdrorg/idxwork              2
sort bes1 /sortwork,8192                  3
idxwork bes2 /pdrorg/idxwork              4
sort bes2 /sortwork,8192                  5
tblname USR01                             6
```

Explanation

1. Specifies the name of the unload data file. Because the -g option is assumed, the unload data file is created at one location.
2. Specifies the name of the directory in which an index information file is to be created (for bes1). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for bes1).
4. Specifies the name of the directory in which an index information file is to be created (for bes2). The index information file is created under this directory.
5. Specifies the name of the work directory for sorting (for bes2).
6. Specifies the authorization identifier of the migration source schema.

(3) Example of the pdrorg command

```
pdrorg -k reld -t USR02.all /pdrorg/rorg01
```

Explanation

`-t`: Specifies the authorization identifier of the schema that is to be unloaded.

12.2 Migrating a stored procedure to another HiRDB system

Executor: HiRDB administrator

The dictionary import/export utility enables a stored procedure that is currently in use to be migrated to another HiRDB system. This facility can be used to achieve the following:

- Migrating a stored procedure from a test system to the actual system
- Migrating a stored procedure created in one HiRDB system to another HiRDB system
- During system reconstruction, storing an existing stored procedure and restoring it after the system has been reconstructed

12.2.1 Preparations for migrating a stored procedure to another HiRDB system

(1) Limitations

A procedure (stored procedure) specified in `CREATE TYPE` cannot be migrated to another HiRDB system.

(2) Information to be migrated

The following information is migrated:

- Table definition information and table data
- Stored procedure

(a) Procedure for migrating table definition information and table data

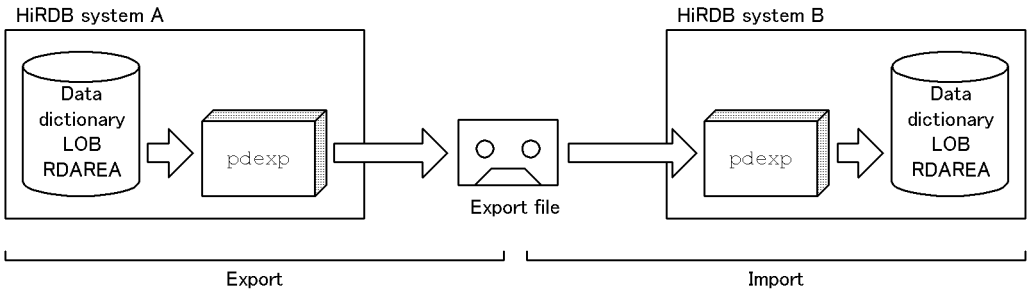
The table definition information and the table data for the table that is used by a stored procedure are migrated to the target system; for the migration procedure, see *12.1 Migrating a table to another HiRDB system*.

This procedure can be skipped if the target system already contains the table definition information and table data for the table that is used by the stored procedure.

(b) Procedure for migrating the stored procedure

The dictionary import/export utility (`pdexp` command) to used to migrate a stored procedure from the data dictionary LOB RDAREAs to the target HiRDB system. Figure 12-5 shows the procedure for migrating a stored procedure with the dictionary import/export utility.

Figure 12-5: Migration of a stored procedure with the dictionary import/export utility



Note: The stored procedure remains the same as in HiRDB system A.

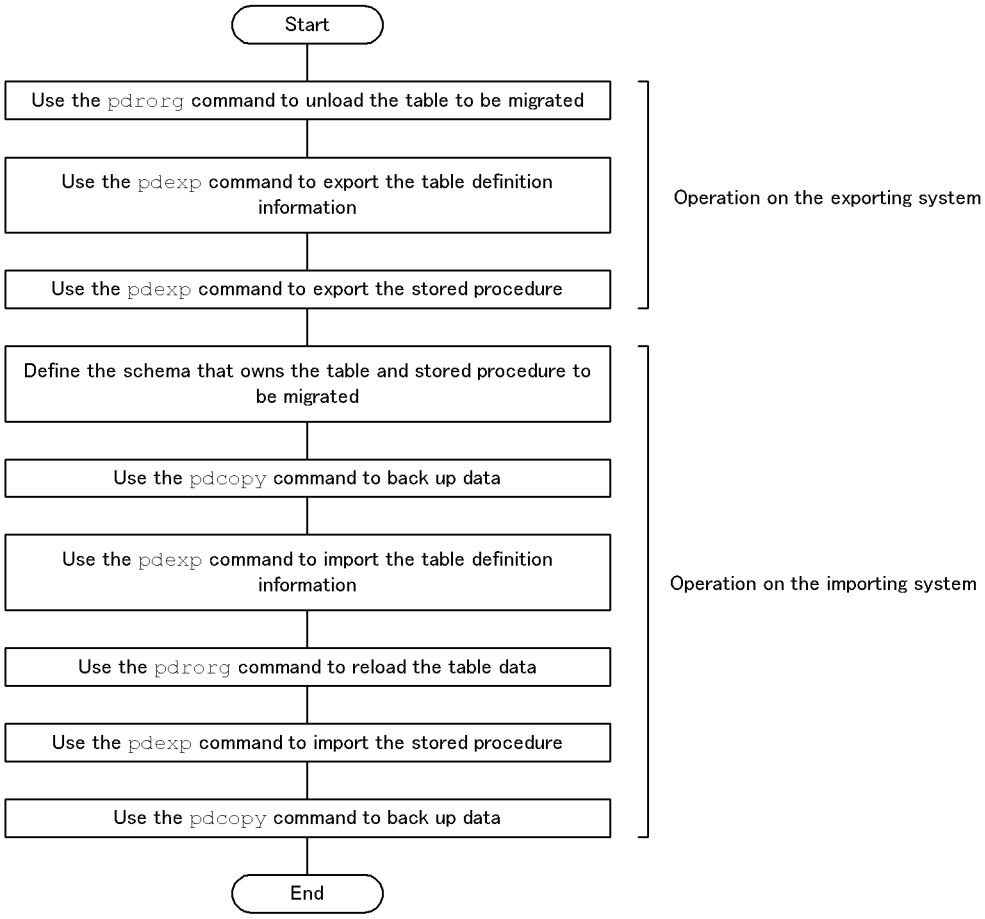
(3) Rules for migration

1. The stored procedure and the table definition information cannot be migrated at the same time. The table definition information should be migrated first, then the stored procedure can be migrated.
2. Before migration is performed, the table that is to be migrated and a schema for the stored procedure must be defined in the target system.
3. Before migration is performed, the data dictionary LOB RDAREAs must be created in the target system.

(4) Migration procedure

Figure 12-6 provides an overview of the procedure for migrating a stored procedure.

Figure 12-6: Procedure for migrating a stored procedure

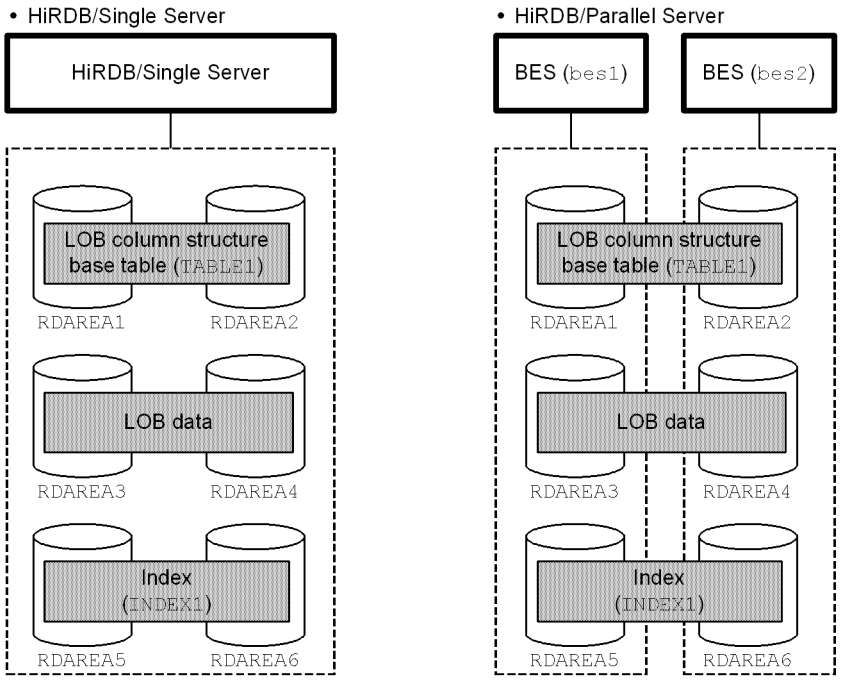


12.2.2 Example

This example migrates a stored procedure (routine identifier PROC1) and a table (TABLE1) used by the PROC1 stored procedure from HiRDB system A to HiRDB system B.

- A LOB column is defined for TABLE1.
- An index (INDEX1) is defined for TABLE1.

It is assumed that RDAREAs with the same names (RDAREA1 to RDAREA6) have already been created in the target system.



(1) Enter the pdhold command to shut down RDAREA1 to RDAREA6

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (`/pdrorg/rorg01`):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1
```

Explanation

Specifies the name of the unload data file.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1              1
unload bes2:/pdrorg/unfile2              2
```

Explanation

1. Specifies the name of the unload data file at a back-end server (`bes1`).
2. Specifies the name of the unload data file at a back-end server (`bes2`).

(3) Use the pdrorg command to unload data for TABLE1

```
pdrorg -k unld -j -t TABLE1 /pdrorg/rorg01
```

Explanation

`-k`: Specifies `unld` in order to unload data.

`-j`: Specifies that a table containing a LOB column is to be unloaded.

`-t`: Specifies the name of the table to be unloaded.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (2).

(4) Use the pdrels command to release RDAREA1 to RDAREA6 from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(5) Create the control statements file for the pdexp command

The following are the contents of the control statements file (`/pdexp/exp01`):

```
-t USR01.TABLE1
```

Explanation

`USR01`: Name of the schema that owns `TABLE1`

`TABLE1`: Name of the table to be exported

(6) Use the pdexp command to export table definition information for TABLE1

```
pdexp -e /pdexp/expfile1 -f /pdexp/exp01
```

Explanation

`-e`: Specifies the name of the file to be exported.

`-f`: Specifies the name of the control statements file for the `pdexp` command created in step (5).

(7) Create the control statements file for the pdexp command

The following are the contents of the control statements file (`/pdexp/exp02`):

```
-p USR01.PROC1
```

Explanation

`USR01`: Name of the schema that owns `PROC1`

`PROC1`: Name of the stored procedure to be exported

(8) Use the pdexp command to export the stored procedure

```
pdexp -e /pdexp/expfile2 -f /pdexp/exp02
```

Explanation

`-e`: Specifies the name of the file to be exported.

`-f`: Specifies the name of the control statements file for the `pdexp` command created in step (7).

(9) Store the unload data file and export files on a medium such as CMT

The unload data file created in step (3) and the export files created in steps (6) and (8) are stored on a medium such as CMT.

The operations at the source system (HiRDB system A) are now completed.

(10) Store the unload data file and export files in target system

The operations from this point on are performed at the target system (HiRDB system B).

The unload data file and export files stored on a medium such as CMT are copied into the target system.

(11) Use the pddef command to define a schema for the user who owns PROC1 and TABLE1

```
pddef
   CREATE SCHEMA AUTHORIZATION USR01;
```

(12) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r
-r RDMAST,RDDIR,RDDIC,DICLOB,RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,
RDAREA6 -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

The RDAREAs are backed up as a safeguard in the event of errors during database migration. The following RDAREAs must be backed up:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs
- Data dictionary LOB RDAREAs
- User RDAREAs and user LOB RDAREAs for storing the imported table (`RDAREA1` to `RDAREA6`)

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-r`: Specifies the names of the RDAREAs to be backed up.

`-b`: Specifies the name of the backup file.

`-p`: Specifies the output destination for the `pdcopy` command's processing results listing.

(13) Use the pdexp command to import table definition information for TABLE1

```
pdexp -i /pdexp/expfile1
```

Explanation

`-i`: Specifies the name of the file to be imported.

(14) Use the pdhold command to shut down RDAREA1 to RDAREA6

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(15) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (`/pdrorg/rorg02`):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                          1
idxwork /pdrorg/idxwork                         2
sort /sortwork,8192                             3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1                     1
idxwork bes1 /pdrorg/idxwork                    2
sort bes1 /sortwork,8192                        3
unload bes2:/pdrorg/unfile2                     4
idxwork bes2 /pdrorg/idxwork                    5
sort bes2 /sortwork,8192                        6
```

Explanation

1. Specifies the name of the unload data file (for `bes1`).
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).
4. Specifies the name of the unload data file (for `bes2`).
5. Specifies the name of the directory in which an index information file is to be created (for `bes2`). The index information file is created under this directory.

6. Specifies the name of the work directory for sorting (for `bes2`).

(16) Use the pdrorg command to reload data for TABLE1

```
pdrorg -k reld -j -t TABLE1 /pdrorg/rorg01
```

Explanation

To re-create the index (`INDEX1`) at the same time, the `-i` option is omitted and the index batch creation mode is used.

`-k`: Specifies `reld` in order to reload data.

`-j`: Specifies that a table containing a LOB column is to be reloaded.

`-t`: Specifies the name of the table to be reloaded.

`/pdrorg/rorg02`: Specifies the name of the control statements file for the `pdrorg` command created in step (15).

(17) Use the pdcopy command to back up the data

```
pdcopy -m /rdarea/mast/mast01 -M r
-r RDMAST,RDDIR,RDDIC,DICLOB,RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,
RDAREA6 -b /pdcopy/backup02 -p /pdcopy/list02
```

Explanation

The following RDAREAs must be backed up:

- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs
- Data dictionary LOB RDAREAs
- User RDAREAs and user LOB RDAREAs for storing the imported table (`RDAREA1` to `RDAREA6`)

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-r`: Specifies the names of the RDAREAs to be backed up.

`-b`: Specifies the name of the backup file.

`-p`: Specifies the output destination of the `pdcopy` command's processing results listing.

(18) Use the pdrels command to release RDAREA1 to RDAREA6 from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(19) Use the pdexp command to import the stored procedure

```
pdexp -i /pdexp/expfile2
```

Explanation

`-i`: Specifies the name of the file to be imported.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

Chapter
13. Handling Tables

This chapter explains the procedures for handling tables.

It contains the following sections:

13.1 Checking table storage efficiency

Executor: HiRDB administrator or DBA privilege holder

As data is added or deleted repeatedly, order is lost in the placement of rows, resulting in a reduction in the data retrieval or data storage efficiency. The HiRDB administrator should check regularly for deterioration in data storage efficiency. If the data storage efficiency does deteriorate, take one of the following actions:

- Reorganize the table.
- Increase the size of the RDAREA.
- Release used free pages or used free segments.

Reference note:

HiRDB identifies the tables, indexes, and RDAREAs that require these operations. For details, see *13.4 Predicting table reorganization time (facility for predicting reorganization time)*.

13.1.1 Executing the database condition analysis utility on a regular basis

The database condition analysis utility (`pddbst` command) should be executed periodically in order to check the data storage efficiency.

(1) Checking table storage efficiency

Table storage efficiency is checked by executing the database condition analysis utility's *condition analysis by table*. The obtained information is used to determine whether or not the following procedures are necessary:

- Table reorganization
- Releasing used free pages or used free segments
- Deletion of unneeded rows
- Changing the table row partitioning storage conditions
- Changing the hash function
- Re-creating or reorganizing indexes
- Expansion, re-initialization, addition, or deletion of RDAREAs

Condition analysis by table cannot be applied to a table for which an abstract data type is defined. In such a case, condition analysis by RDAREA (physical analysis) must be used in order to check the table's data storage efficiency.

(2) Checking index storage efficiency

Index storage efficiency is checked by executing the database condition analysis utility's *condition analysis by index*. The obtained information is used to determine whether or not the following procedures are necessary:

- Re-creating or reorganizing indexes
- Releasing used free pages or used free segments
- Reducing the number of index page splits

Note, however, that condition analysis by index does not obtain information on plug-in indexes.

(3) Checking cluster key and clustering data page storage efficiency

Cluster key and clustering data page storage efficiency are checked by executing the database condition analysis utility's *cluster key and clustering data page storage condition analysis*. The obtained information is used to determine whether or not the following procedures are necessary:

- Table reorganization
- Redefinition of a table with a cluster key specified and without the index component column containing many duplicated data items
- Reevaluation of the column structure in the index definition

(4) Checking data storage efficiency in RDAREAs

Data storage efficiency in RDAREAs is checked by executing the database condition analysis utility's *condition analysis by RDAREA*. The obtained information is used to determine whether or not the following procedures are necessary:

- Table reorganization
- Re-creating or reorganizing indexes
- Releasing used free pages or used free segments.
- Changing the table row partitioning storage conditions
- Changing the hash function
- Expansion, re-initialization, addition, or deletion of RDAREAs

Other methods for checking RDAREA utilization status

- The `pddbls -a` command can be used to check RDAREA utilization status.
- The RDAREA utilization status that is output during backup processing by the database copy utility (`pdcopy` command) can also be used to check the RDAREA storage efficiency.

13.1.2 Messages indicating poor data storage efficiency

When the data storage efficiency becomes poor, HiRDB outputs the following messages:

- KFPA12300-I or KFPH00211-I
- KFPH00212-I
- KFPH22017-I

(1) KFPA12300-I or KFPH00211-I is output

If an RDAREA begins to run out of free space, the KFPA12300-I or KFPH00211-I message is output (indicating the last file in the RDAREA or the overall RDAREA utilization factor).

Action to be taken by the HiRDB administrator

Execute the pddbls -a command or the database condition analysis utility's *condition analysis by RDAREA* to determine whether or not the RDAREA structure should be modified; for details on RDAREA structure modification, see *15. Handling RDAREAs*.

(2) KFPH00212-I is output

When the table retrieval efficiency or storage efficiency has become degraded, the KFPH00212-I message is output.

Action to be taken by the HiRDB administrator

Use conditional analysis by table in the database condition analysis utility to take one of the following actions:

- Reorganize the table
- Release used free pages and used free segments

For details about reorganizing a table, see *13.2 Reorganizing a table* and *13.3 Reorganizing a table (examples)*; for details about releasing used free pages and used free segments, see *15.9 Re-using used free pages and used free segments*.

In the cases listed below, however, you must first use the database structure modification utility (pdmod command) to change the structure of the RDAREA (for details about changing the structure of an RDAREA, see *15. Handling RDAREAs*):

- This message is issued frequently for tables in the same RDAREA.
- This message is issued while a table is being reorganized or immediately after it has been reorganized.
- This message is issued while an index is being reorganized or immediately

after it has been reorganized.

(3) KFPH22017-I is output

When LOB data retrieval efficiency or storage efficiency is reduced, the `KFPH22017-I` message is output.

Action to be taken by the HiRDB administrator

Execute the database condition analysis utility's *condition analysis by table* and determine whether or not table reorganization is called for; for details on table reorganization, see *13.2 Reorganizing a table* and *13.3 Reorganizing a table (examples)*.

The RDAREA to be reorganized depends on the RDAREA type displayed in the `KFPH22017-I` message:

- **User LOB RDAREA**

  Reorganize the LOB data in the user LOB RDAREA.

- **Data dictionary LOB RDAREA**

  Reorganize the dictionary tables for the following stored procedures and stored functions:

  - `SQL_ROUTINES`
  - `SQL_ROUTINE_RESOURCES`
  - `SQL_ROUTINE_PARAMS`

13.1.3 When expected retrieval performance cannot be achieved

If expected retrieval performance cannot be achieved, the database condition analysis utility should be executed to check the data storage condition. Based on this information, whether or not the data storage condition should be improved must be evaluated.

To improve the system performance, system tuning may be necessary based on the statistical information that is obtained by executing the following commands:

- Statistics analysis utility (`pdstedit` command)
- `pdbufls` command

For details about system tuning, see *20. Obtaining Tuning Information* and *21. Tuning*.

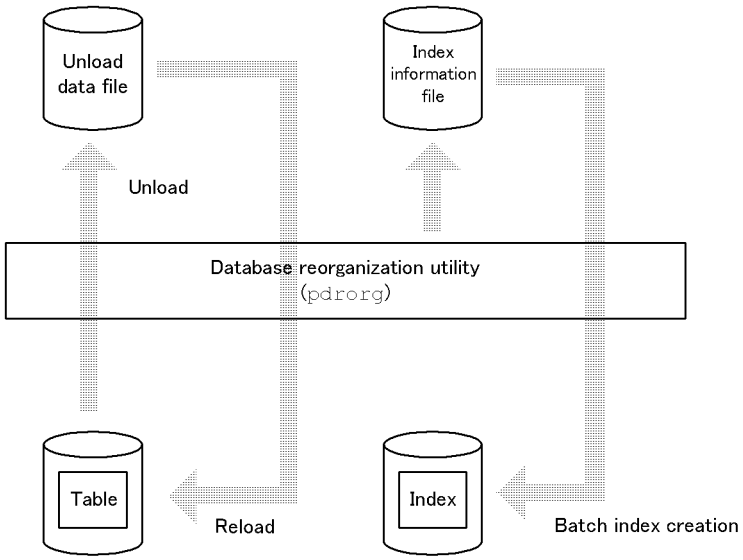
13.2 Reorganizing a table

Executor: HiRDB administrator and table owner (or user with DBA privilege)

13.2.1 Table reorganization

As data is added or deleted repeatedly, the data storage efficiency becomes poor, which has an adverse effect on data retrieval performance. To avoid this, tables should be reorganized periodically with the `pdrorg` command (database reorganization utility). Figure 13-1 provides an overview of table reorganization processing.

Figure 13-1: Overview of table reorganization processing



Explanation

- The table data is copied into an unload data file; this is called table data *unloading*. The data is then stored back into the table; this is called *reloading*. This 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. HiRDB uses that information to create the index in the batch mode, thus also reorganizing the index.

13.2.2 Execution units for table reorganization

Table reorganization can be executed in the following units:

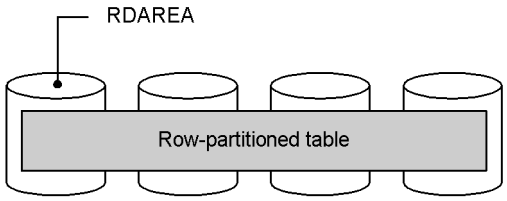
- By table

- By RDAREA
- By schema

(1) Reorganization by table

Reorganization processing can be performed on an entire table; this is the method that is usually used. Reorganization by table is executed when the results of executing the database condition analysis utility indicate that the entire table needs to be reorganized. Figure 13-2 illustrates reorganization of an entire table.

Figure 13-2: Reorganization of an entire table



Note: The data indicated by shading is subject to reorganization.

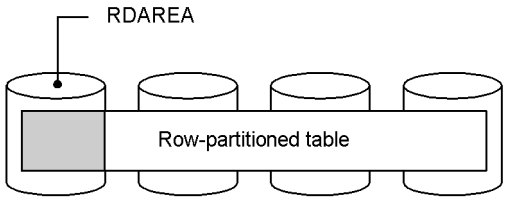
Explanation

The table to be reorganized is specified in the `-t` option of the database reorganization utility.

(2) Reorganization by RDAREA

Reorganization by RDAREA reorganizes the RDAREAs in which the table is stored. This method can be used only on tables that are row-partitioned. Reorganization of RDAREAs is executed when the results of the database condition analysis utility indicate that to reorganizing only a portion of a row-partitioned table would suffice. This reduces the processing time compared with reorganization of the entire table. Figure 13-3 illustrates reorganization of an RDAREA.

Figure 13-3: Reorganization of an RDAREA



Note: The data indicated by shading is subject to reorganization.

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.

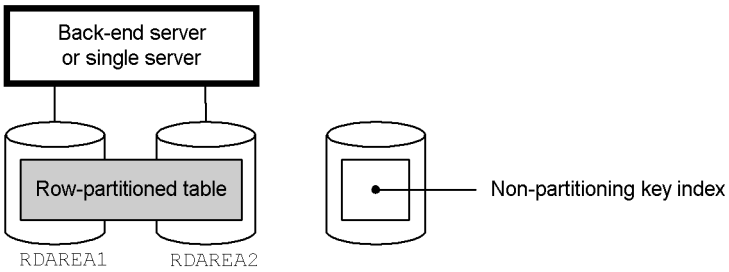
Reference note:

If an RDAREA satisfying all of the conditions listed below is reorganized by RDAREA, a non-partitioning key index will not be created. To create a non-partitioning key index, the database reorganization utility (re-creating an index) must be executed again.

- The table to be reorganized has row partitioning within a server.
- A non-partitioning key index is defined for the table to be reorganized.
- The non-partitioning key index is not row-partitioned.

Figure 13-4 shows an example in which a non-partitioning key index is not created during reorganization by RDAREA.

Figure 13-4: Example in which a non-partitioning key index is not created during reorganization by RDAREA



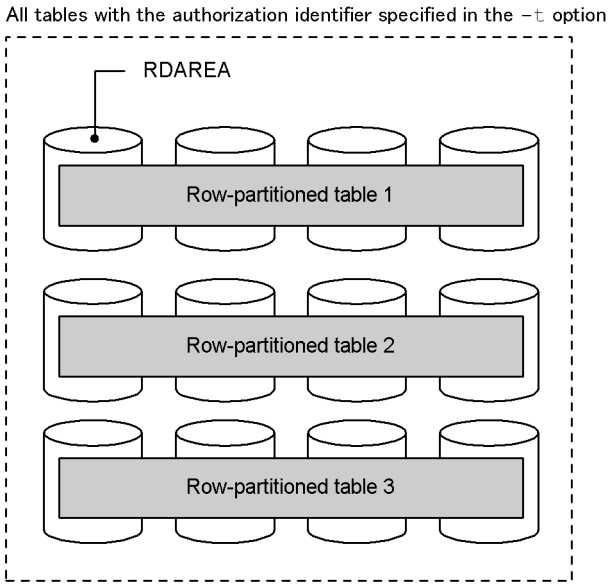
Explanation

Even though reorganization by RDAREA is applied to `RDAREA1`, a non-partitioning key index is not created. Index information will simply be output to an index information file. To create a non-partitioning key index, the index must be re-created with the database reorganization utility. Thus, reorganization by table, rather than by RDAREA, is recommended in such a case.

(3) 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 13-5 illustrates reorganization of a schema.

Figure 13-5: Reorganization of a schema



Note: The data indicated by shading is subject to reorganization.

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`.

13.2.3 Selecting an update log acquisition mode for a database

The following three update log acquisition modes are provided for reorganizing a database when the database reorganization utility is executed:

- Log acquisition mode
- Pre-update log acquisition mode (default)
- No-log mode

For details about the applicability of these modes, see *7.1 Database update log acquisition modes*.

(1) Mode selection criteria

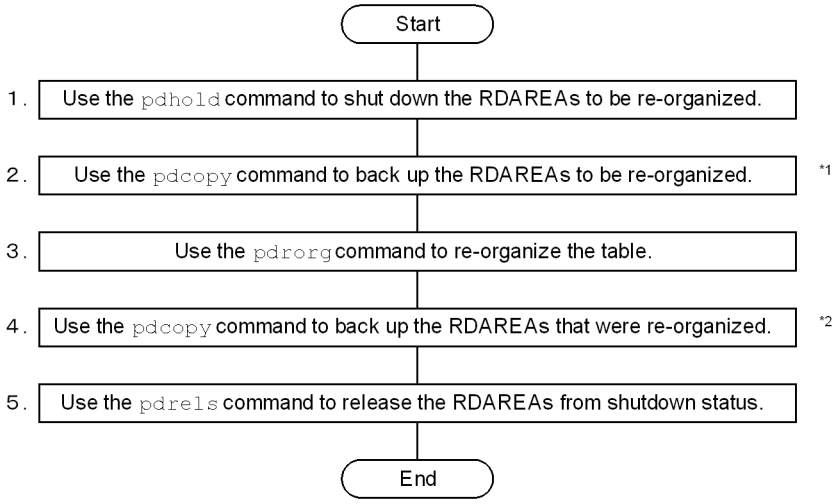
In most cases, select the default mode, which is the pre-update log acquisition mode. Consider using another mode only under the following conditions:

Condition	Mode to select
There is a large amount of data to be reorganized, and it will take a long time.	No-log mode
There is very little table data to be reorganized.	Log acquisition mode

(2) Differences in operating methods

The manner in which database reorganization is performed depends on the mode that is selected, as shown in Figure 13-6.

Figure 13-6: Differences in how tables are reorganized depending on the database update log acquisition mode



[1] This step is required when the no-log mode is selected. The backup that is obtained is used to recover the RDAREA in the event the `pdrorg` command terminates abnormally while the no-log mode is in effect. However, you need not make this backup if the conditions described in *(3) Conditions under which a backup is not needed before reorganizing* are satisfied.

[2] This step is required when the pre-update log acquisition mode or the no-log mode is selected. If you do not make a backup at this point, the RDAREAs cannot be recovered to their most recent status with the `pdrstr` command in the event the need to recover them arises (they cannot be recovered to include the updates implemented by the `pdrorg` command). The RDAREAs can be recovered only to their status at the time the `pdrorg` command is executed.

Tip

When the pre-update log acquisition mode or the no-log mode is selected, keep the RDAREAs to be reorganized in shutdown status from steps 1 through 4 (in Figure 13-6). Otherwise, if the need arises to recover the RDAREAs with the `pdrstr` command, any updates to the RDAREAs that were performed before the backup is made in step 4 will not be recovered. You will only be able to recover the RDAREAs to the point at which the `pdrorg` command is executed. This is because if the system log files that the `pdrstr` command uses as input contain logs collected in the pre-update log acquisition mode or the no-log mode, the `pdrstr` command that is used to recover the RDAREAs will terminate in an error.

(3) Conditions under which a backup is not needed before reorganizing

Normally in the no-log mode, a backup must be made before the `pdrorg` command is executed. However, if either of the conditions described in 1 and 2 below is satisfied, it is not necessary to make the backup because the RDAREAs can be recovered to their status at the time the `pdrorg` command executed even if the command terminates abnormally. We do recommended, however, that in general you make a backup, because recovering RDAREAs is simpler if a backup is available.

No.	Condition		How to recover the RDAREAs if an error occurs
1	Unload data can be used to recover the RDAREAs to their status at the time the `pdrorg` command was executed.	If the RDAREAs contain only the table to be reorganized and its indexes.	Recover with the database reorganization utility (`pdmod` command) in order to re-initialize the RDAREAs that are being reorganized, and then reload the data.
		If all of the following conditions are satisfied: • The table being reorganized contains LOB data. • The RDAREAs contain only a LOB column structure base table being reorganized. • The unload data was obtained by specifying the `-j` option.	Recover with the database reorganization utility (`pdmod` command) in order to re-initialize the RDAREAs that are being reorganized, and then specify the `-j` option.
2	The backup and the system logs can be used to recover the RDAREAs to their status at the time the `pdrorg` command was executed.		Recover with the `pdclose` command in order to close the RDAREAs that are being reorganized, the `pdlogswap` command to swap the system log files, and then the database recovery utility (`pdrstr` command) to input the system logs output up to this point.

13.2.4 Before reorganizing a table

(1) Reorganizing a table in which a LOB column is defined

A LOB column structure base table and the LOB data can be reorganized at the same time. It is also possible to reorganize only the LOB column structure base table or only the LOB data. To reorganize at the same time both a table in which a LOB column is defined and the LOB data, Hitachi recommends that the `-j` option be specified in the `pdrorg` command.

(2) Reorganizing a table in which an abstract data type is defined

Not all tables in which an abstract data type is defined can be reorganized, as shown in Table 13-1.

Table 13-1: Reorganizability of a table in which an abstract data type is defined

Condition		Reorganizability
Abstract data type provided by a plug-in	Without LOB attribute	Can be reorganized.
	With LOB attribute	Only column structure base tables of abstract data type can be reorganized.#
User-defined abstract data type		Can be reorganized.

#: If the plug-in has the UNLOAD facility or the constructor parameter reverse generation facility, the entire table can be reorganized by specifying the `-j` option.

(3) 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.

Note:

1. Use of this facility reduces processing efficiency because synchronization point processing is also executed.
2. If the utility terminates abnormally, the error handling that will be required will depend on the timing of the termination; for the error handling methods, see *18.18 When a utility terminates abnormally during execution of a reorganization with synchronization points set*.
3. If a table is row-partitioned into multiple back-end servers, you should consolidate the unload data files (by specifying the `-g` option in the database reorganization utility). If this is not done, the error handling should the utility terminate abnormally will become quite complicated; for details, see *18.18.3 Actions to be taken when a utility terminates abnormally before unload data files have been consolidated (HiRDB/ Parallel Server only)*.
4. Because data is stored on a new page for each synchronization point, the number of pages needed for storing data is greater than would be the case if this facility were not used. Therefore, do not use this facility to reorganize a table that is full. If it is used in such a case, a space shortage may cause an error in the database reorganization utility.
5. Reorganization with synchronization points set cannot be performed on falsification prevented tables.

(4) Reorganizing a table in an RDAREA that is full

The value specified in the `PCTFREE` operand of `CREATE TABLE` is used as the percentage of unused space to be left in each page when a table is reorganized. Therefore, if a table in an RDAREA that is full is reorganized, an RDAREA space shortage may occur during table reorganization. To prevent this, you should specify the `tblfree` and `idxfree` operands in the `option` statement of the database reorganization utility (`pdrorg` command), and change the percentage of unused space per page specified in the `PCTFREE` operand of `CREATE TABLE`.

You use the `tblfree` operand to specify a percentage of unused space in the table, and you use the `idxfree` operand to specify a percentage of unused space in its indexes.

It must be noted that these are temporary measures that are used when an RDAREA cannot be expanded immediately before table reorganization. In preparation for data updating, you should use the database structure modification utility (`pdmod` command) to expand the RDAREA so that reorganization can be performed within the value specified in the `PCTFREE` operand of `CREATE TABLE`.

(5) When HiRDB Datareplicator is used to link data

When the database reorganization utility is executed on an extracted database, `n` or `p` should be specified in the `-l` option (which specifies execution of the database reorganization utility in the no-log mode or the pre-update log acquisition mode).

When the database reorganization utility is executed in the log acquisition mode, only some of the update information is transferred from the extracted database to the target database, thereby losing conformity between the two databases.

(6) Creating an unload data file in a character special file

To create an unload data file in a character special file, the character special file must be created in a HiRDB file system area for utilities. To do this, `UTL` must be specified in the `-k` option of the `pdfmkfs` command.

(7) Size of the file used during reorganization

If a message reporting a disk space shortage is output during execution of the database reorganization utility, even though ample disk space is available, the following are possible causes:

- Use of large files is not specified (`pd_large_file_use = Y` is not displayed).
- The upper limit of the kernel parameter has been exceeded.

In this case, specify use of large files or change the kernel parameter value. You can also avoid the problem by creating multiple unload data files. However, if the OS does not support large files, you must keep the disk partition size to 2 GB or smaller to be able to handle multiple files.

(8) After reorganization has been completed

If necessary, the optimizing information collection utility (`pdgetcst` command) should be executed after reorganization has been completed. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.

13.3 Reorganizing a table (examples)

Executor: HiRDB administrator and table owner (or user with DBA privilege)

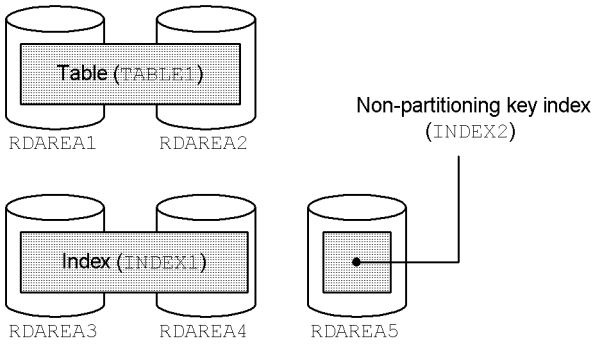
This section presents and explains examples of table reorganization. The following is a list of the examples:

- Example 1: Reorganizing a table (HiRDB/Single Server)
- Example 2: Reorganizing a table (HiRDB/Parallel Server)
- Example 3: Reorganizing an RDAREA
- Example 4: Reorganizing a schema
- Example 5: Reorganizing a table in which a LOB column is defined
- Example 6: Reorganizing data dictionary tables
- Example 7: Reorganizing in no-log mode
- Example 8: Reorganizing a table in which an abstract data type is defined

13.3.1 Example 1: Reorganizing a table (HiRDB/Single Server)

This example reorganizes a row-partitioned table (`TABLE1`) by table. The conditions for this reorganization are as follows:

- `TABLE1` is row-partitioned in user RDAREAs (`RDAREA1` and `RDAREA2`).
- A row partitioning index (`INDEX1`) and a non-partitioning key index (`INDEX2`) are defined for `TABLE1`.
- `INDEX1` is row-partitioned in user RDAREAs (`RDAREA3` and `RDAREA4`).
- `INDEX2` is stored in a user RDAREA (`RDAREA5`).
- The indexes are created in the batch mode (default) when the table is reorganized.
- The table is reorganized in the pre-update log acquisition mode (default).



Note: The shaded data will be reorganized.

Procedure

1. Use the `pdhold` command to shut down the RDAREAs to be reorganized.
2. Create a control statements file for the `pdrorg` command.
3. Use the `pdrorg` command to reorganize the table.
4. Back up the RDAREAs that were reorganized.
5. Use the `pdrels` command to release the shutdown status of the RDAREAs.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

- Because you are using the `pdrorg` command in the pre-update log acquisition mode, you must make a backup after executing the `pdrorg` command, as described in step 4.
- Because you are using the `pdrorg` command in the pre-update log acquisition mode, keep the RDAREAs being reorganized in shutdown status from step 1 through step 4.

(1) Use the pdhold command to shut down RDAREAs to be reorganized

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (/pdrorg/rorg01):

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(3) Use the pdrorg command to reorganize the table

```
pdrorg -k rorg -t TABLE1 /pdrorg/rorg01
```

Explanation

Because two indexes (INDEX1 and INDEX2) are to be re-created simultaneously in the batch mode, the -i option is omitted and the indexes are created in the batch index creation mode.

-k: Specifies rorg for reorganization.

-t: Specifies the name of the table that is to be reorganized.

/pdrorg/rorg01: Specifies the name of the control statements file for the pdrorg command created in step (2).

(4) Back up the RDAREAs that were reorganized

Back up the RDAREAs that were reorganized (RDAREA1 to RDAREA5). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release RDAREAs from shutdown status

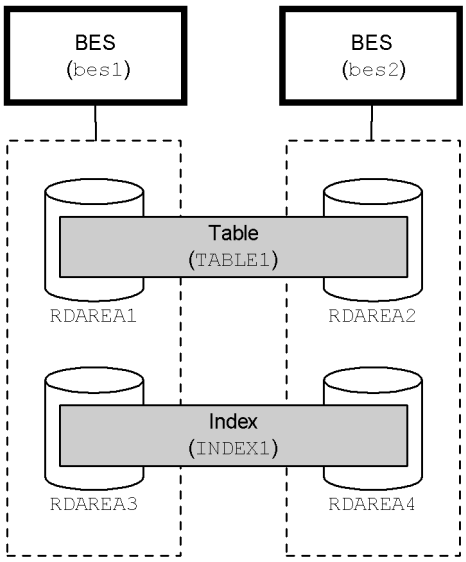
```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

13.3.2 Example 2: Reorganizing a table (HiRDB/Parallel Server)

This example reorganizes a row-partitioned table (TABLE1) by table. The conditions for this reorganization are as follows:

- TABLE1 is row-partitioned in user RDAREAs (RDAREA1 and RDAREA2).
- A row partitioning index (INDEX1) is defined for TABLE1. INDEX1 is row-partitioned in user RDAREAs (RDAREA3 and RDAREA4).
- The index is created in the batch mode (default) when the table is reorganized.
- The table is reorganized in the pre-update log acquisition mode (default).



Note: The data indicated by shading is subject to reorganization.

Procedure

1. Use the pdhold command to shut down the RDAREAs to be reorganized.
2. Create a control statements file for the pdrorg command.
3. Use the pdrorg command to reorganize the table.
4. Back up the RDAREAs that were reorganized.
5. Use the pdrels command to release the RDAREAs from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

- Because you are using the `pdrorg` command in the pre-update log acquisition mode, you must make a backup after executing the `pdrorg` command, as described in step 4.
- Because you are using the `pdrorg` command in the pre-update log acquisition mode, keep the RDAREAs being reorganized in shutdown status from steps 1 through 4.

(1) Use the pdhold command to shut down RDAREAs to be reorganized

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (`/pdrorg/rorg01`):

```
unload bes1:/pdrorg/unfile1                 1
idxwork bes1 /pdrorg/idxwork                2
sort bes1 /sortwork,8192                    3
unload bes2:/pdrorg/unfile2                 4
idxwork bes2 /pdrorg/idxwork                5
sort bes2 /sortwork,8192                    6
```

Explanation

1. Specifies the name of the unload data file (for `bes1`).
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).
4. Specifies the name of the unload data file (for `bes2`).
5. Specifies the name of the directory in which an index information file is to be created (for `bes2`). The index information file is created under this directory.
6. Specifies the name of the work directory for sorting (for `bes2`).

(3) Use the pdrorg command to reorganize the table

```
pdrorg -k rorg -t TABLE1 /pdrorg/rorg01
```

Explanation

Because indexes (INDEX1 and INDEX2) are to be re-created simultaneously in the batch mode, the -i option is omitted and indexes are created in the batch index creation mode.

-k: Specifies rorg for reorganization.

-t: Specifies the name of the table that is to be reorganized.

/pdrorg/rorg01: Specifies the name of the control statements file for the pdrorg command created in step (2).

(4) Back up the RDAREAs that were reorganized

Back up the RDAREAs that were reorganized (RDAREA1 to RDAREA4). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release RDAREAs from shutdown status

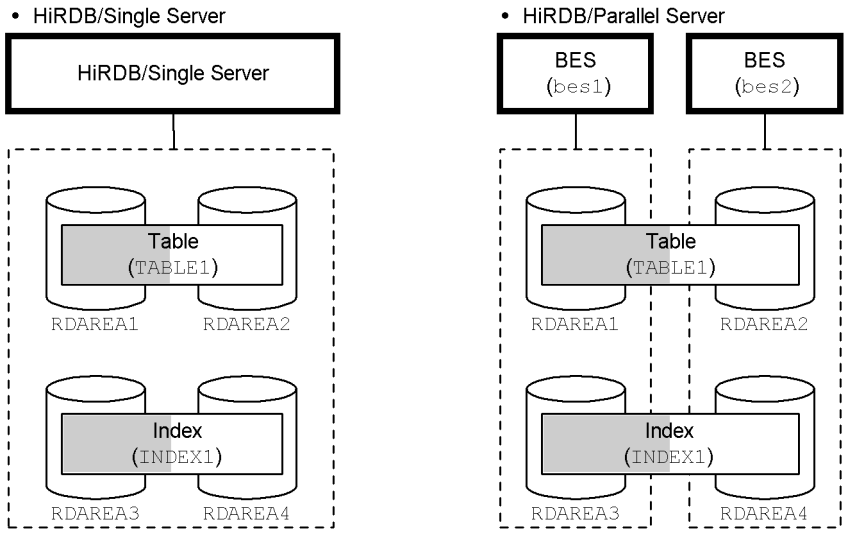
```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.3.3 Example 3: Reorganizing an RDAREA

This example reorganizes a row-partitioned table (TABLE1) by RDAREA. The conditions for this reorganization are as follows:

- TABLE1 is row-partitioned in user RDAREAs (RDAREA1 and RDAREA2).
- An index (INDEX1) is defined for TABLE1. INDEX1 is row-partitioned in user RDAREAs (RDAREA3 and RDAREA4).
- The data to be reorganized is the table data in RDAREA1 and the index data in RDAREA3.
- The table is reorganized in the pre-update log acquisition mode (default).



Note: The shaded data will be reorganized.

Procedure

1. Use the `pdhold` command to shut down the RDAREAs to be reorganized.
2. Create a control statements file for the `pdrorg` command.
3. Use the `pdrorg` command to reorganize the table.
4. Back up the RDAREAs that were reorganized.
5. Use the `pdrels` command to release the RDAREAs from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

- Because you are using the `pdrorg` command in the pre-update log acquisition mode, you must make a backup after executing the `pdrorg` command, as described in step 4.
- Because you are using the `pdrorg` command in the pre-update log acquisition mode, keep the RDAREAs being reorganized in shutdown status from steps 1 through 4.

(1) Use the pdhold command to shut down RDAREAs to be reorganized

```
pdhold -r RDAREA1,RDAREA3
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (`/pdrorg/rorg01`):

(a) HiRDB\Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1               1
idxwork bes1 /pdrorg/idxwork              2
sort bes1 /sortwork,8192                  3
```

Explanation

1. Specifies the name of the unload data file (for `bes1`).
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).

(3) Use the pdrorg command to reorganize the table

```
pdrorg -k rorg -t TABLE1 -r RDAREA1 /pdrorg/rorg01
```

Explanation

Because the index data of `INDEX1` stored in `RDAREA3` is also to be re-created

simultaneously, the -i option is omitted and the index is created in the batch index creation mode.

-k: Specifies rorg for reorganization.

-t: Specifies the name of the table that is to be reorganized.

-r: Specifies the name of the RDAREA in which the table (TABLE1) that is to be reorganized is stored.

/pdrorg/rorg01: Specifies the name of the control statements file for the pdrorg command created in step (2).

(4) Back up the RDAREAs that were reorganized

Back up the RDAREAs that were reorganized (RDAREA1 and RDAREA3). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release RDAREAs from shutdown status

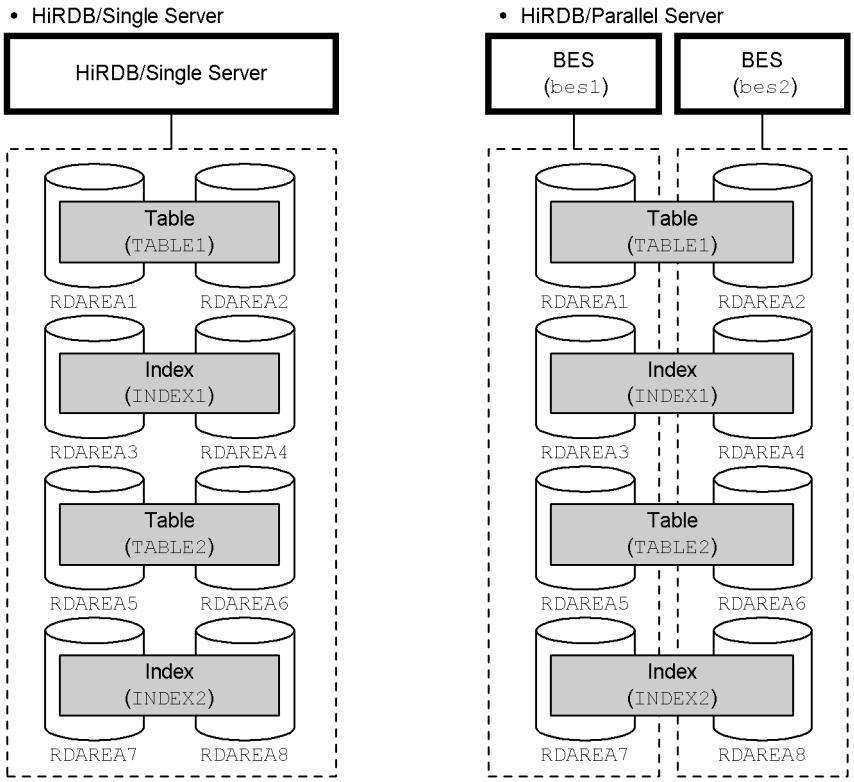
```
pdrels -r RDAREA1,RDAREA3
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

13.3.4 Example 4: Reorganizing a schema

This example reorganizes all tables owned by user USR01. The following are the conditions for this reorganization:

- The indexes are created in the batch mode (default) when the tables are reorganized.
- The tables are reorganized in the pre-update log acquisition mode (default).



Note: The data indicated by shading is subject to reorganization.

Procedure

1. Use SQL to check for RDAREAs to be reorganized.
2. Use the `pdhold` command to shut down the RDAREAs to be reorganized.
3. Create a control statements file for the `pdrorg` command.
4. Use the `pdrorg` command to reorganize the tables.
5. Back up the RDAREAs that were reorganized.
6. Use the `pdrels` command to release the RDAREAs from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

- Because you are using the pdrorg command in the pre-update log acquisition mode, you must make a backup after executing the pdrorg command, as described in step 5.
- Because you are using the pdrorg command in the pre-update log acquisition mode, keep the RDAREAs being reorganized in shutdown status from steps 2 through 5.

(1) Use SQL to check for RDAREAs to be reorganized

```
SELECT DISTINCT(RDAREA_NAME) FROM MASTER.SQL_TABLES            1
     WHERE TABLE_SCHEMA=USR01 AND RDAREA_NAME IS NOT NULL;
SELECT DISTINCT(RDAREA_NAME) FROM MASTER.SQL_DIV_TABLE         2
     WHERE TABLE_SCHEMA=USR01;
SELECT DISTINCT(RDAREA_NAME) FROM MASTER.SQL_INDEXES           3
     WHERE TABLE_SCHEMA=USR01 AND RDAREA_NAME IS NOT NULL;
SELECT DISTINCT(RDAREA_NAME) FROM MASTER.SQL_DIV_INDEX         4
     WHERE TABLE_SCHEMA=USR01;
```

Explanation

1. Retrieves RDAREAs that store non-row-partitioned tables.
2. Retrieves RDAREAs that store row-partitioned tables.
3. Retrieves RDAREAs that store non-row-partitioned indexes.
4. Retrieves RDAREAs that store row-partitioned indexes.

(2) Use the pdhold command to shut down RDAREAs to be reorganized

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,...
```

(3) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (/pdrorg/rorg01):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1                 1
idxwork bes1 /pdrorg/idxwork                2
sort bes1 /sortwork,8192                    3
idxwork bes2 /pdrorg/idxwork                4
sort bes2 /sortwork,8192                    5
```

Explanation

1. Specifies the name of the unload data file. Because the `-g` option is assumed, the unload data file is created at one location.
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).
4. Specifies the name of the directory in which an index information file is to be created (for `bes2`). The index information file is created under this directory.
5. Specifies the name of the work directory for sorting (for `bes2`).

(4) Use the pdrorg command to reorganize the table

```
pdrorg -k rorg -t USR01.all /pdrorg/rorg01
```

Explanation

Because indexes (`INDEX1` and `INDEX2`) are to be re-created simultaneously, the `-i` option is omitted and the indexes are created in the batch index creation mode.

`-k`: Specifies `rorg` for reorganization.

`-t`: Specifies the authorization identifier of the schema that is to be reorganized.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (2).

Remarks

When reorganization by schema is specified, the -j option (reorganization when LOB data is present) and the -g option (unload data file standardization) are assumed.

(5) Back up the RDAREAs that were reorganized

Back up the RDAREAs that were reorganized (RDAREA1 to RDAREA8). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(6) Use the pdrels command to release RDAREAs from shutdown status

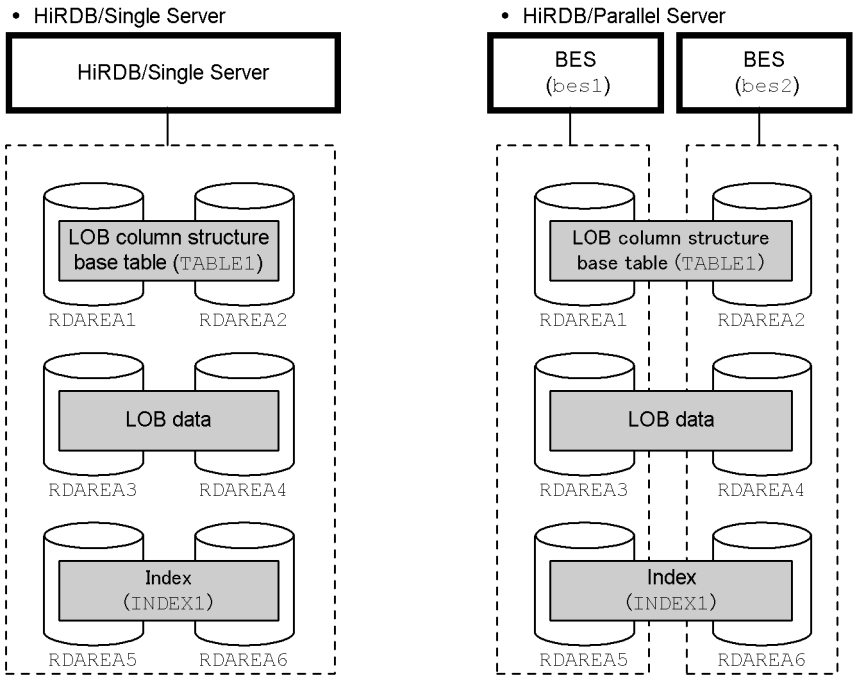
```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

13.3.5 Example 5: Reorganizing a table in which a LOB column is defined

This example reorganizes by table a table (TABLE1) in which a LOB column is defined; the LOB data is reorganized at the same time. The conditions for this reorganization are as follows:

- The LOB column structure base table is row-partitioned in user RDAREAs (RDAREA1 and RDAREA2).
- The LOB data is row-partitioned in user LOB RDAREAs (RDAREA3 and RDAREA4).
- An index (INDEX1) is defined for TABLE1. INDEX1 is row-partitioned in user RDAREAs (RDAREA5 and RDAREA6).
- The index is created in the batch mode (default) when the table is reorganized.
- The table is reorganized in the pre-update log acquisition mode (default).



Note: The data indicated by shading is subject to reorganization.

Procedure

1. Use the `pdhold` command to shut down the RDAREAs to be reorganized.
2. Create a control statements file for the `pdrorg` command.
3. Use the `pdrorg` command to reorganize the table.
4. Back up the RDAREAs that were reorganized.
5. Use the `pdrels` command to release the RDAREAs from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

- Because you are using the `pdrorg` command in the pre-update log acquisition mode, you must make a backup after executing the `pdrorg` command, as described in step 4.
- Because you are using the `pdrorg` command in the pre-update log acquisition mode, keep the RDAREAs being reorganized in shutdown status from steps 1 through 4.

(1) Use the pdhold command to shut down RDAREAs to be reorganized

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (`/pdrorg/rorg01`):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1               1
idxwork bes1 /pdrorg/idxwork              2
sort bes1 /sortwork,8192                  3
unload bes2:/pdrorg/unfile2               4
idxwork bes2 /pdrorg/idxwork              5
sort bes2 /sortwork,8192                  6
```

Explanation

1. Specifies the name of the unload data file (for `bes1`).

2. Specifies the name of the directory in which an index information file is to be created (for bes1). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for bes1).
4. Specifies the name of the unload data file (for bes2).
5. Specifies the name of the directory in which an index information file is to be created (for bes2). The index information file is created under this directory.
6. Specifies the name of the work directory for sorting (for bes2).

(3) Use the pdrorg command to reorganize the table

```
pdrorg -k rorg -j -t TABLE1 /pdrorg/rorg01
```

Explanation

Because an index (INDEX1) is also to be re-created simultaneously, the -i option is omitted and the index is created in the batch index creation mode.

-k: Specifies rorg for reorganization.

-j: Specifies that the table to be reorganized contains a LOB column.

-t: Specifies the name of the table that is to be reorganized.

/pdrorg/rorg01: Specifies the name of the control statements file for the pdrorg command created in step (2).

(4) Back up the RDAREAs that were reorganized

Back up the RDAREAs that were reorganized (RDAREA1 to RDAREA6). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.3.6 Example 6: Reorganizing data dictionary tables

This example reorganizes data dictionary tables that are stored in the data dictionary RDAREA (DATADIC) and in a data dictionary LOB RDAREA (DATALOB).

Reorganize the dictionary table in the pre-update log acquisition mode (default).

Procedure

1. Use the `pdhold` command to shut down the data dictionary RDAREA and data dictionary LOB RDAREA.
2. Back up the data dictionary RDAREA and the data dictionary LOB RDAREA.
3. Create a control statements file for the `pdrorg` command.
4. Use the `pdrorg` command to reorganize the dictionary table.
5. Back up the data dictionary RDAREA and the data dictionary LOB RDAREA.
6. Use the `pdrels` command to release the data dictionary RDAREA and data dictionary LOB RDAREA from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

Because you are using the `pdrorg` command in the pre-update log acquisition mode, keep the data dictionary RDAREA and data dictionary LOB RDAREA being reorganized in shutdown status from steps 1 through 5.

(1) Use the pdhold command to shut down the data dictionary RDAREA and data dictionary LOB RDAREA

You will be backing up the RDAREAs while they are in shutdown status. After you have backed them up, keep the RDAREAs in shutdown status so that other users do not update their contents.

```
pdhold -r DATADIC,DATALOB
```

(2) Back up the data dictionary RDAREA and the data dictionary LOB RDAREA

To be prepared for the possibility that the RDAREAs will prove to be too small during reorganization, back up the data dictionary RDAREA and the data dictionary LOB RDAREA. If the size of the data dictionary RDAREA or the data dictionary LOB RDAREA becomes insufficient during reorganization, you will have to be able to recover the RDAREAs from a backup. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(3) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (/pdrorg/rorg01):

```
unload /pdrorg/unfile1                    1
lobunld /pdrorg/unfile2                   2
```

Explanation

1. Specifies the name of the unload data file for the data dictionary RDAREA.
2. Specifies the name of the unload data file for the data dictionary LOB RDAREA.

(4) Use the pdrorg command to reorganize the data dictionary table

```
pdrorg -k rorg -c dic /pdrorg/rorg01
```

Explanation

-k: Specifies rorg in order to execute reorganization.

-c: Specifies that a data dictionary table is to be reorganized.

/pdrorg/rorg01: Specifies the name of the control statements file for the pdrorg command created in step (3).

(5) Back up the data dictionary RDAREA and the data dictionary LOB RDAREA

Back up the data dictionary RDAREA and the data dictionary LOB RDAREA (DATADIC and DATALOB). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(6) Use the pdrels command to release the data dictionary RDAREA and data dictionary LOB RDAREA from shutdown status

```
pdrels -r DATADIC,DATALOB
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

(7) Note

Exercise caution if the following error occurs during the reorganization of a dictionary table:

- *Although unloading was completed, during reloading a capacity shortage occurred in the data dictionary RDAREA.*

In such a case, use the procedure explained below to reorganize the dictionary table.

Procedure

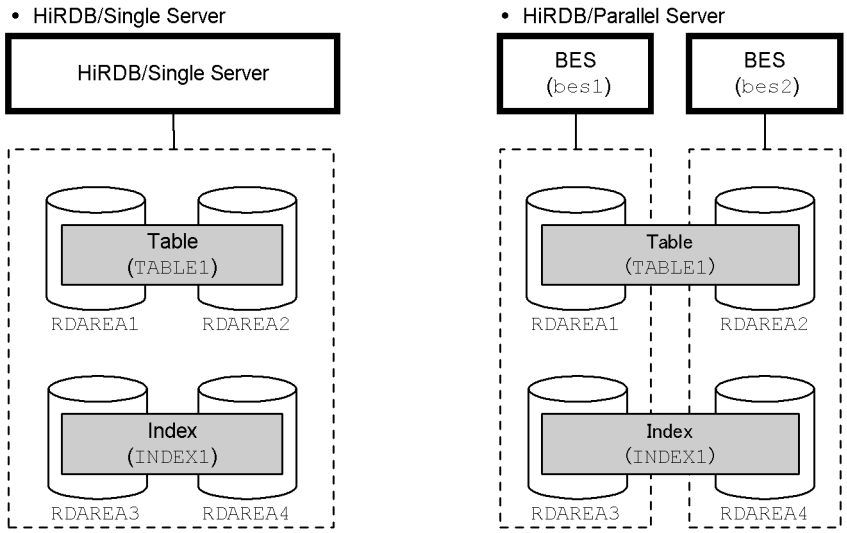
1. Using the backup made before reorganizing, execute the database recovery utility to recover the data dictionary RDAREA and the data dictionary LOB RDAREA.
2. Use the database reorganization utility to expand or add the data dictionary RDAREA and the data dictionary LOB RDAREA.
3. Reorganize the dictionary table again.

   Using the unload data file created before the error occurred (before the RDAREAs were expanded or added), perform everything but reloading.

13.3.7 Example 7: Reorganizing in no-log mode

This example reorganizes a row-partitioned table (`TABLE1`) by table. The conditions for this reorganization are as follows:

- `TABLE1` is row-partitioned in user RDAREAs (`RDAREA1` and `RDAREA2`).
- An index (`INDEX1`) is defined for `TABLE1`. `INDEX1` is row-partitioned in user RDAREAs (`RDAREA3` and `RDAREA4`).
- The index is to be created in the batch mode when the table is reorganized.
- The table is to be reorganized in the pre-update log acquisition mode or the no-log mode.



Note: The data indicated by shading is subject to reorganization.

Procedure

1. Use the `pdhold` command to shut down the RDAREAs to be reorganized.
2. Back up the RDAREAs to be reorganized.
3. Create a control statements file for the `pdrorg` command.
4. Use the `pdrorg` command to reorganize the table.
5. Back up the RDAREAs that were reorganized.
6. Use the `pdrels` command to release the RDAREAs from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

- Because you are using the pdrorg command in the no-log mode, you must make a backup before you execute the pdrorg command. This backup will be used to recover the RDAREAs if the pdrorg command terminates abnormally. However, you need not make this backup if certain conditions are met. For details about these conditions, see *13.2.3(3) Conditions under which a backup is not needed before reorganizing*.
- Because you are using the pdrorg command in the no-log mode, you must make a backup after executing the pdrorg command, as described in step 5.
- Because you are using the pdrorg command in the no-log mode, keep the RDAREAs being reorganized in shutdown status from steps 1 through 5.

(1) Use the pdhold command to shut down RDAREAs to be reorganized

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4
```

(2) Back up the RDAREAs to be reorganized

Back up the RDAREAs to be reorganized (RDAREA1 to RDAREA4). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(3) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (/pdrorg/rorg01):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.
3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1                 1
idxwork bes1 /pdrorg/idxwork                2
sort bes1 /sortwork,8192                    3
unload bes2:/pdrorg/unfile2                 4
idxwork bes2 /pdrorg/idxwork                5
sort bes2 /sortwork,8192                    6
```

Explanation

1. Specifies the name of the unload data file (for `bes1`).
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).
4. Specifies the name of the unload data file (for `bes2`).
5. Specifies the name of the directory in which an index information file is to be created (for `bes2`). The index information file is created under this directory.
6. Specifies the name of the work directory for sorting (for `bes2`).

(4) Use the pdrorg command to reorganize the table

```
pdrorg -k rorg -t TABLE1 -l n /pdrorg/rorg01
```

Explanation

To re-create the index (`INDEX1`) at the same time, the `-i` option is omitted and the index batch creation mode is used.

`-k`: Specifies `rorg` in order to execute reorganization.

`-t`: Specifies the name of the table to be reorganized.

`-l`: Specifies the no-log mode (`n`).

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (2).

(5) Back up the RDAREAs that were reorganized

Back up the RDAREAs that were reorganized (`RDAREA1` to `RDAREA4`). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(6) Use the pdrels command to release RDAREAs from shutdown status

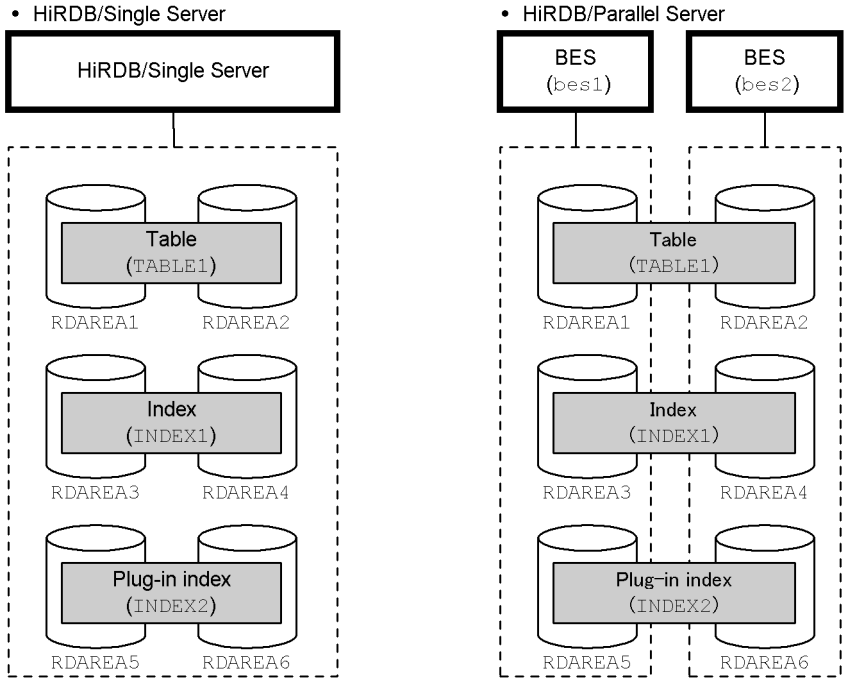
```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.3.8 Example 8: Reorganizing a table in which an abstract data type is defined

This example reorganizes a table (TABLE1) in which is defined an abstract data type (GEOMETRY or FILELINK type) without the LOB attribute. For an example of reorganizing with an abstract data type (SGMLTEXT type) and the LOB attribute, see *13.3.5 Example 5: Reorganizing a table in which a LOB column is defined*. The following are the conditions for this reorganization:

- TABLE1 is row-partitioned in user RDAREAs (RDAREA1 and RDAREA2)
- The table is reorganized in the pre-update log acquisition mode (default).



Note: The data indicated by shading is subject to reorganization.

Procedure

1. Use the pdhold command to shut down the RDAREAs to be reorganized.
2. Create a control statements file for the pdrorg command.
3. Use the pdrorg command to reorganize the table.
4. Back up the RDAREAs that were reorganized.
5. Use the pdrels command to release the RDAREAs from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

Hint:

- Because you are using the pdrorg command in the pre-update log acquisition mode, you must make a backup after executing the pdrorg command, as described in step 4.
- Because you are using the pdrorg command in the pre-update log acquisition mode, keep the RDAREAs being reorganized in shutdown status from steps 1 through 4.

(1) Use the pdhold command to shut down RDAREAs to be reorganized

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

(2) Create the control statements file for the pdrorg command

The following are the contents of the control statements file (/pdrorg/rorg01):

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                    1
idxwork /pdrorg/idxwork                   2
sort /sortwork,8192                       3
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies the name of the directory in which an index information file is to be created. The index information file is created under this directory.

3. Specifies the name of the work directory for sorting.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1                 1
idxwork bes1 /pdrorg/idxwork                2
sort bes1 /sortwork,8192                    3
unload bes2:/pdrorg/unfile2                 4
idxwork bes2 /pdrorg/idxwork                5
sort bes2 /sortwork,8192                    6
```

Explanation

1. Specifies the name of the unload data file (for `bes1`).
2. Specifies the name of the directory in which an index information file is to be created (for `bes1`). The index information file is created under this directory.
3. Specifies the name of the work directory for sorting (for `bes1`).
4. Specifies the name of the unload data file (for `bes2`).
5. Specifies the name of the directory in which an index information file is to be created (for `bes2`). The index information file is created under this directory.
6. Specifies the name of the work directory for sorting (for `bes2`).

(3) Use the pdrorg command to reorganize the table

```
pdrorg -k rorg -t TABLE1 /pdrorg/rorg01
```

Explanation

`-k`: Specifies `rorg` for reorganization.

`-t`: Specifies the name of the table that is to be reorganized.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command created in step (2).

(4) Back up the RDAREAs that were reorganized

Back up the RDAREAs that were reorganized (`RDAREA1` to `RDAREA6`). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5,RDAREA6
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.4 Predicting table reorganization time (facility for predicting reorganization time)

Executor: User with DBA privilege

This section explains how to predict the time at which it will be necessary to reorganize a table (including indexes) or expand an RDAREA. The following items are explained:

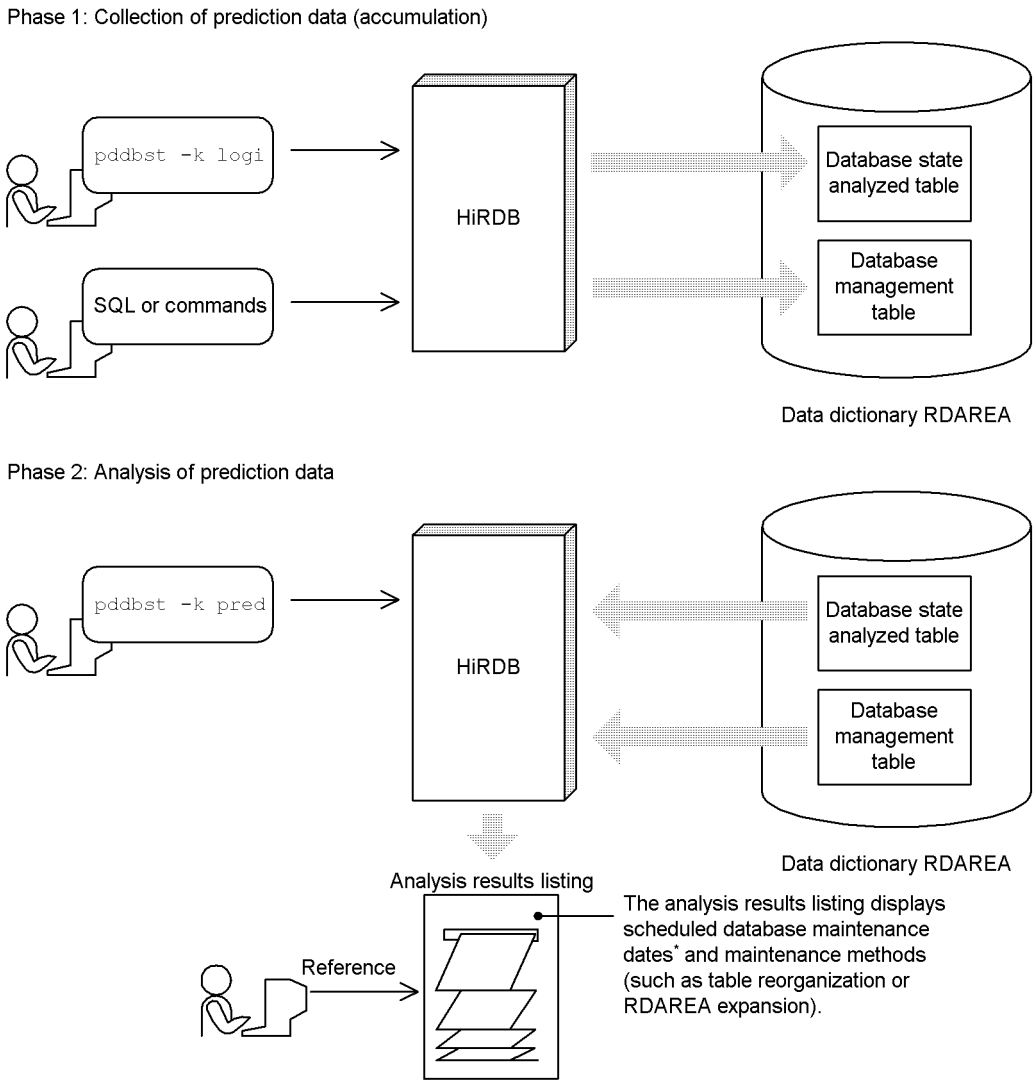
- Predicting reorganization time
- Preparations for using the facility for predicting reorganization time
- Operational flow
- Notes on using the facility for predicting reorganization time
- Stopping reorganization time prediction
- Customizing reorganization time prediction

13.4.1 Predicting reorganization time

In the past, to determine when it was necessary to reorganize tables and indexes or to expand RDAREAs, the user had to make a comprehensive evaluation of the tables to be reorganized, and of the timing for reorganization based on messages that were output, and of the execution results of the `pddbst` command. This meant that there was a risk that tables that didn't need to be reorganized were reorganized, and that tables that needed to be reorganized were not reorganized because output messages were not read.

To simplify these operations, HiRDB can now predict when reorganization will become necessary. This is called the *facility for predicting reorganization time*. Figure 13-7 provides an overview of the facility for predicting reorganization time.

Figure 13-7: Overview of the facility for predicting reorganization time



* The scheduled date on which an RDAREA will need to be maintained is called the *scheduled database maintenance date*. For an overview of how HiRDB analyzes prediction data, see *13.4.6(1) HiRDB's analysis of prediction data*.

Reorganization time prediction is divided into the following two phases:

- Phase 1: Collection of reorganization time prediction data
  - The `pddbst` command is executed regularly to accumulate database analysis

results in a database state analyzed table.

- When any of the SQL statements or commands listed below is executed, database management history information is output to a database management table:

  - Definition SQL (`DROP SCHEMA`, `DROP TABLE`, `DROP INDEX`, `ALTER TABLE`)
  - Data manipulation SQL (`PURGE TABLE`)
  - `pdrorg` command
  - `pdreclaim` command
  - `pdload` command
  - `pdmod` command

- Phase 2: Analysis of reorganization time prediction data

  Using the database state analyzed table and the database management table as input information, the `pddbst` command is executed to analyze the reorganization time prediction data. The user views the execution results of the `pddbst` command and takes one of the actions listed below, as appropriate. Note that RDAREA automatic extension is performed automatically by HiRDB if automatic extension was specified when the RDAREA was created (no user action is required).

  - Execution of the `pdrorg` command to reorganize a table or index
  - Execution of the `pdreclaim` command to release free pages and segments that are being used
  - Execution of the `pdmod` command to extend an RDAREA
  - RDAREA automatic extension
  - Execution of the `pdmod` command to re-initialize an RDAREA

Reference note:

- Prediction levels 1 and 2 are supported by the facility for predicting reorganization time. Prediction level 1 is used principally to monitor space shortages in RDAREAs. Prediction level 2 is used to monitor the effect of poor data storage efficiency on online performance as well as to monitor space shortages in RDAREAs.
- Because the primary purpose of the facility for predicting reorganization time is to prevent RDAREA space shortages, this facility predicts reorganization time by assuming the maximum amount of data storage after reorganization. For example, a branched row (where data is partitioned and stored on multiple pages) is assumed to remain a branched row after reorganization. Therefore, when determining whether to reorganize a table or extend an RDAREA, the facility tends to predict RDAREA extension.

13.4.2 Preparations for using the facility for predicting reorganization time

Before using the facility for predicting reorganization time, the preparations explained in this subsection must be performed.

(1) Creating a data dictionary RDAREA for storing the database state analyzed table and database management table

After estimating the size of the data dictionary RDAREA for storing the database state analyzed table and the database management table, use the database structure modification utility (`pdmod` command) to create a data dictionary RDAREA.

For details about how to estimate the size of a data dictionary RDAREA for storing a database state analyzed table and database management table, see the manual *HiRDB Version 8 Installation and Design Guide*.

Reference note:

If a space shortage occurs in the data dictionary RDAREA for storing the database state analyzed table and database management table, HiRDB cannot predict reorganization time (although such a space shortage does not cause an error in SQL statements or commands).

If the initial estimate results in a space shortage because a table has been extended, extend the data dictionary RDAREA that stores the database state analyzed table and database management table.

(2) Specifying the pd_rorg_predict operand

Specify `Y` in the `pd_rorg_predict` operand.

(3) Estimating the size of the system log file

Whenever the database state analyzed table or the database management table is updated, a system log is output. For this reason, you must re-estimate the size of the system log file. For details about how to estimate the size of the system log file, see the manual *HiRDB Version 8 Installation and Design Guide*.

Reference note:

For a HiRDB/Parallel Server, system log information is output to system log files at the dictionary server.

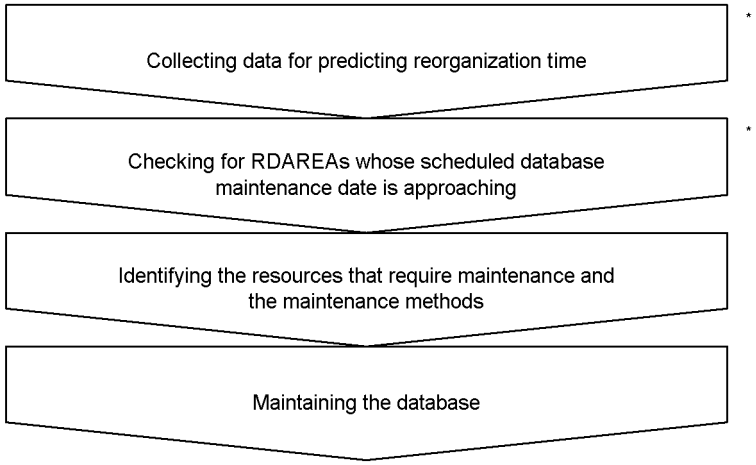
13.4.3 Operational flow

This subsection describes the operational flow for predicting the reorganization timing. For prediction level 1, see (1); for prediction level 2, see (1) and (2).

(1) For prediction level 1

Figure 13-8 shows the operational flow for predicting table reorganization time.

Figure 13-8: Operational flow for predicting table reorganization time



* Hitachi recommends that you perform these operations every day.

(a) Collecting data for predicting reorganization time

Execute the command shown below to collect data for predicting reorganization time and to store the data in the database state analyzed table. You should execute this command on a daily basis to accumulate data for predicting reorganization time.

```
pddbst -k logi -r ALL -e 1
```

Note:

The operation in step (a) must be performed at least four times before going on to step (b); otherwise, the `pddbst` command terminates in an error. In determining the number of times step (a) has been performed, you cannot count an execution of the command if there has been no change in RDAREA status since the last time the command was executed. In other words, an execution of step (a) counts only if there has been a change in RDAREA status. For details, see *13.4.4(2)(a) When prediction data has not been collected correctly*.

Reference note:

If the data storage status has changed significantly, there is a high probability that reorganization will be required. Therefore, after you update a large volume of data, Hitachi recommends that you perform the operations in steps (a) and (b).

(b) Checking for RDAREAs whose scheduled database maintenance date is approaching

Execute the command shown below to check for RDAREAs that are close to their scheduled database maintenance date. You should execute this command on a daily basis.

```
pddbst -k pred -r ALL -e 1
```

A listing is produced of RDAREAs that are close to their scheduled database maintenance dates. An example follows:

Output example

```
 pddbst 07-02       ***** Rdarea resource forecast *****     2005/05/15 11:07:10
No         Date        RdareaName                       ResKind
---------- ----------- -------------------------------- ----------
         1 2005/05/31  "RDUSER01"                       Segment
         2 2005/06/01  "RDUSER02"                       Segment
---------- ----------- -------------------------------- ----------
```

Explanation

`Date`: Displays the scheduled database maintenance dates.

`RdareaName`: Displays the RDAREAs that will require maintenance.

Note:

Because a scheduled database maintenance date is simply an estimate, an RDAREA space shortage may occur before the scheduled date depending on operations.

Reference note:

- It may take a considerable amount of time for the `pddbst` command in step (c) to execute. To avoid adverse effects on online jobs, you should execute the operations in steps (b) and (c) during time periods when online jobs will not be affected adversely.
- If the execution time of the operation in step (c) does not pose a problem, you may skip the operation in step (b) and perform only the operation in step (c).

(c) Identifying the resources that require maintenance and the maintenance methods

If the results of step (b) identify RDAREAs that are close to their scheduled database maintenance dates, execute the command shown below to check for the tables, indexes, and RDAREAs that require maintenance.

```
pddbst -k pred -r ALL -e 1 -m
```

Tables, indexes, and RDAREAs that need to be maintained are listed, together with the appropriate maintenance method for each RDAREA. An output example follows:

Output example

```
 pddbst 07-02       ***** Rdarea resource forecast *****     2005/05/15 11:07:10
No         Date        RdareaName                       ResKind
---------- ----------- -------------------------------- ----------
         1 2005/05/31  "RDUSER01"                       Segment
         2 2005/06/01  "RDUSER02"                       Segment
---------- ----------- -------------------------------- ----------

 pddbst 07-02       ***** Maintenance Information *****      2005/05/15 11:07:10
No          : 1
Rdarea Name : "RDUSER01"
Method      :           ...[1]
Segment     : 42560
------------------------------------------------------------------------------------
--
 Reclaim      Reorganize   Type Name                                        Date
------------ ------------ ---- ------------------------------------------
----------
      28089 *       9363    T   "k1234567"."table01"   ...[2]              2005/05/31
          0        12342 *  T   "k1234567"."table02"   ...[3]              2005/05/31
        851         7235 *  I   "k1234567"."index01"   ...[4]              2005/05/31
          3          -10    T   "k1234567"."table10"   ...[5]              2005/05/31
====================================================================================
==
No          : 2
Rdarea Name : "RDUSER02"
Method      : Expand   ...[6]
Segment     : 1523
------------------------------------------------------------------------------------
--
 Reclaim      Reorganize   Type Name                                        Date
------------ ------------ ---- ------------------------------------------
----------
        228         1035    T   "k1234567"."table01"                      2005/06/03
          0          121    T   "k1234567"."table03"                      2005/06/03
         30           60    I   "k1234567"."index07"                      2005/06/03
====================================================================================
==
```

Explanation

1. For RDAREA `RDUSER01`, there is no entry in the `Method` field, so no maintenance is necessary for this RDAREA.
2. There is an asterisk (`*`) in the `Reclaim` field for the table named `table01` in RDAREA `RDUSER01`. This means that the `pdreclaim` command can be used to release free pages and segments that are being used.
3. There is an asterisk (`*`) in the `Reorganize` field for the table named `table02` in RDAREA `RDUSER01`. This means that the `pdrorg` command can be used to reorganize this table.
4. There is an asterisk (`*`) in the `Reorganize` field for the index named `index01` in RDAREA `RDUSER01`. This means that the `pdrorg` command can be used to reorganize this index.

5. There is no asterisk (*) shown for the table named table10 in RDAREA RDUSER01. This means that no maintenance is necessary for this table.
6. For RDAREA RDUSER02, an RDAREA maintenance method is shown in the Method field. The following entries can appear in the Method field:

   Expand: Expand the RDAREA. In this example, the number of segments that need to be expanded is 1523, as shown in the Segment field.

   Extend: HiRDB will extend the RDAREA automatically.

   Reinit: Re-create the database (unload the data from the RDAREA, reinitialize the RDAREA, and then reload the unloaded data).

   No display: No maintenance is necessary for the RDAREA.

Reference note:

- The meanings of the other headers follow:

  Type: Indicates the resource type:

  T: Table

  I: Index

  L: LOB RDAREA

  Name: Table name or index name.

  Date: Scheduled maintenance date.
- The number of segments shown is the minimum number of segments that need to be expanded by the scheduled maintenance date. It is advisable to expand more segments than the indicated number. Note that there is a limit to the number of times an RDAREA can be expanded.

(d) Maintaining the database

Pursuant to instructions issued by HiRDB, you should take one of the following actions, as appropriate:

- Reorganize a table.
- Reorganize an index.
- Expand an RDAREA.
- Reinitialize an RDAREA.

Note:

Do not perform the operation in step (a) while any of the operations in step (d) is underway. Otherwise, correct predictions cannot be made.

(2) For prediction level 2

For the collection of reorganization time prediction data under prediction level 1, the facility collects information for the RDAREAs only; therefore, only the directory pages are subject to analysis. Under prediction level 2, on the other hand, the facility also collects information for each table and each index, so the data and index pages also require analysis. This means that the execution time required for `pddbst` to collect prediction data is longer under prediction level 2 than under prediction level 1.

If reorganization time prediction data is collected under prediction level 2 while an online job is executing, there may be adverse effects on the application. To minimize the effects on the online job application, you can choose one (or both) of the following execution methods:

- Interval analysis
- Merge analysis

For details about interval analysis and merge analysis, see the manual *HiRDB Version 8 Command Reference*.

13.4.4 Notes on using the facility for predicting reorganization time

(1) Resources that are excluded from predictions

The following type of user LOB RDAREA is excluded from reorganization time prediction:

- User LOB RDAREA that stores a plug-in index

(2) Cases in which correct prediction cannot be made

Correct reorganization time prediction cannot be made in the cases explained below.

(a) When prediction data has not been collected correctly

For the facility for predicting reorganization time to be able to execute, at least four rounds of prediction data reflecting data storage status changes must have been accumulated. In addition, if prediction data is not collected regularly, it may not be possible to make valid predictions. For example, if prediction data is collected only for the first four days of the week in a system that is updated at the end of each week, valid predictions will not be forthcoming.

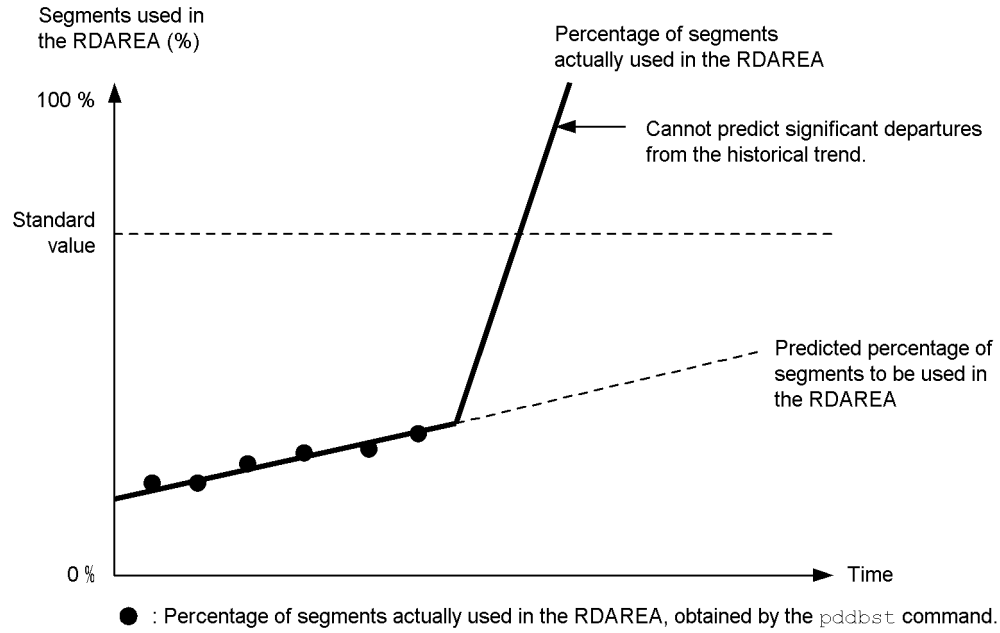
Reference note:

The more regularly prediction data is collected, the higher the prediction accuracy becomes.

(b) Operation in which data storage status changes abruptly

Because HiRDB predicts future usage based on the historical record of the percentages of segments used in an RDAREA, it cannot make valid predictions if the data storage status changes abruptly. Figure 13-9 shows an example.

Figure 13-9: Example in which valid prediction cannot be made due to abrupt changes in the data storage status



Note: For the standard value, see *13.4.6 Customizing reorganization time prediction*.

(c) When a table is stored in an RDAREA that does not have an optimum page size

When a table is stored in an RDAREA that does not have an optimum page size matching the table row length, reorganization time cannot be predicted correctly. For example, reorganization may be performed on a table that will not benefit from reorganization, or a reorganization instruction may be issued immediately following reorganization.

(d) Operation in which both the original RDAREA and a replica RDAREA coexist in the current RDAREA

Reorganization time prediction is targeted at the original RDAREA. A replica RDAREA is not the target of reorganization time prediction. Results of a command and SQL analysis for the replica RDAREA are not incorporated into the database state

analyzed table or database management table. Therefore, correct prediction results cannot be computed if both the original RDAREA and the replica RDAREA coexist in the current RDAREA.

(e) Rebalancing table in which key values are not balanced

When you define a rebalancing table, you should select a hash function and a partitioning key that result in a uniform distribution of the data. If the key values are not balanced, too much data may be stored in a particular hash group (segment), and as a result, correct prediction will not be possible.

(3) Prediction trend

If a table satisfies all of the following conditions, the prediction is likely to call for RDAREA expansion:

- Variable-length character strings (`VARCHAR`, `NVARCHAR`, or `MVARCHAR`) are defined in the table.
- The no-split option is not defined for these variable-length character strings.
- The data was stored using a method other than by data-loading with the `pdload` command.

(4) When a user-defined abstract data type exists

If a user-defined abstract data type (an abstract data type that is not provided by a plug-in) is defined for a table, reorganization by the `pdrorg` command or release of free pages being used by the `pdreclaim` command cannot be executed. Consequently, no reorganization instruction will be issued. If an RDAREA contains only such a table, only an RDAREA expansion or re-initialization instruction can be issued.

(5) Limits during prediction

If a variable-length character string (`VARCHAR`, `NVARCHAR`, or `MVARCHAR`) whose actual length is 255 bytes or less branches to another page (for example, the case in which data of 256 bytes or larger is updated so that it becomes 255 bytes or smaller), reorganization can cancel the branching and improve storage efficiency and access efficiency. However, because the data length information is not collected, it cannot be incorporated into the prediction.

In the case of a table that has the data type of variable-length character string for which `NO SPLIT` is not specified, if there is a large amount of data whose actual length is shorter than the page length, many pages with poor usage rate result even immediately after reorganization. As a result, customization by the standard value definition file becomes necessary.

(6) Reorganizing the database state analyzed table and the database management table

When the dictionary table is reorganized, both the database state analyzed table and the

database management table are also subject to reorganization. Reorganize these tables together with other dictionary tables. There is no need to reorganize only the database state analyzed table and the database management table.

Reference note:

When a large number of tables and indexes are deleted, invalid areas are created in the indexes that are defined to improve access efficiency to the two tables. However, because these areas are reused when new tables and indexes are defined, there is no need to reorganize these areas if the data dictionary RDAREA has sufficient capacity.

(7) Prediction accuracy

In the case of prediction level 1, when there is only one resource in an RDAREA (for example, only a single table is stored in the RDAREA), prediction is made based on the information in the database state analyzed table and database management table. When there are multiple resources in an RDAREA, prediction is made based on the information in the database state analyzed table only. Therefore, prediction accuracy is higher in the first case than in the second.

(8) Relationship between database recovery and prediction of reorganization timing

The reorganization timing is predicted on the basis of the database's history. If the database is not restored after a failure to its most recent status (such as when the database is restored to the point where a backup was made), correct prediction cannot be achieved. In such a case, correct prediction can be obtained by resetting the base information for prediction. This is called *resetting the accumulated condition analysis results*. For details about resetting the accumulated condition analysis results, see the manual *HiRDB Version 8 Command Reference*.

13.4.5 Stopping reorganization time prediction

To stop reorganization time prediction, change the specification value for the `pd_rorg_predict` operand to `N`. To resume reorganization time prediction, change the operand value back to `Y`.

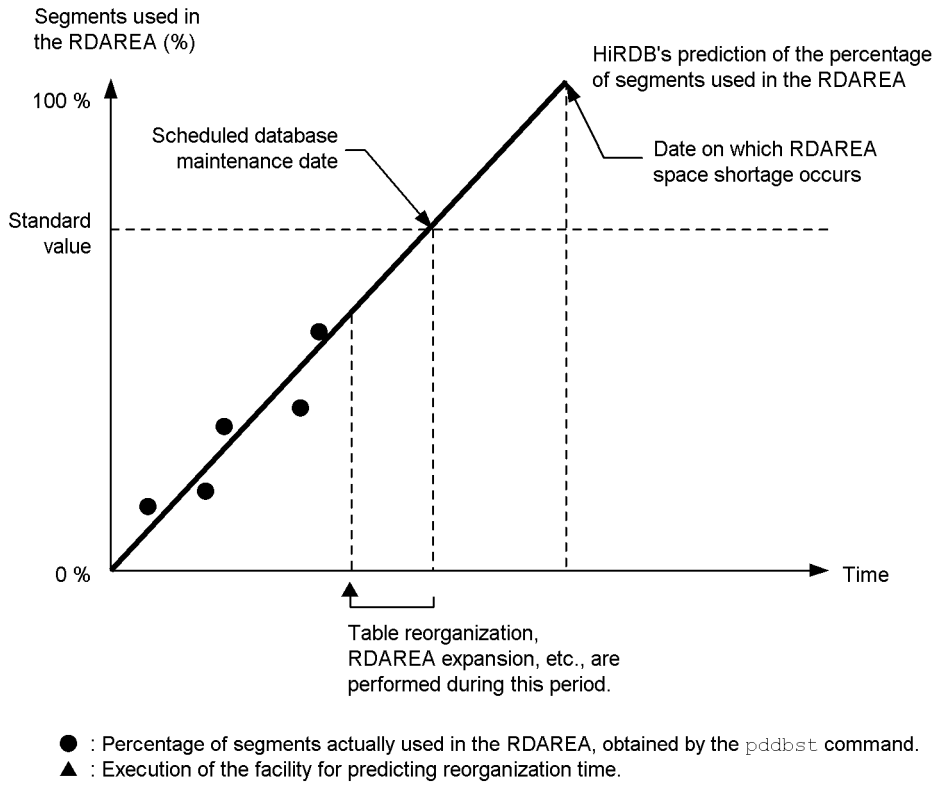
To completely terminate use of reorganization time prediction and to use the data dictionary RDAREA that stores the database state analyzed table and database management table as a user RDAREA, first use the `pdmod` command to delete the data dictionary RDAREA that stores the database state analyzed table and database management table (do not delete them while the `pddbst` command is executing); then, either change the specification value for the `pd_rorg_predict` operand to `N` or delete this operand.

13.4.6 Customizing reorganization time prediction

(1) HiRDB's analysis of prediction data

HiRDB predicts a scheduled database maintenance date based on the historical trend of the percentage of segments used in an RDAREA. Figure 13-10 provides an overview of how HiRDB analyzes prediction data.

Figure 13-10: Overview of how HiRDB analyzes prediction data



Explanation

- The date on which the percentage of segments being used in the RDAREA is predicted to exceed the standard value (the default is 80%) is used as the scheduled database maintenance date.
- You can change the standard value in the standard value definition file. For details about the standard value definition file, see the manual *HiRDB Version 8 Command Reference*.

(2) Customizing the standard value

If the standard value is not appropriate for your operations, such as when the storage conditions change before the predicted database maintenance date because initial data loading is performed periodically, you can change the standard value to customize the reorganization time prediction performed by HiRDB.

13.5 Deleting data from a table

Executor: Table owner or DELETE privilege holder

The DELETE or the PURGE TABLE statement is used to delete an unneeded row from a table.

Hint:

- When a row is deleted from a view table, the corresponding row is deleted from its base table.
- For falsification prevented tables, only rows whose deletion prevented duration has elapsed can be deleted.

(1) Deleting a selected row

To delete a selected row, the WHERE clause must be specified in the DELETE statement.

(2) Deleting all rows

To delete all rows from a table, the following SQL must be executed:

- DELETE statement with the WHERE clause not specified
- PURGE TABLE statement

Points to be considered

- When all rows are to be deleted from a table that contains many rows, the PURGE TABLE statement deletes rows faster than the DELETE statement.
- When many rows are to be deleted with the DELETE statement, the lock mode should be set by first executing the LOCK statement with EXCLUSIVE specified in order to reduce the overhead for exclusive control.
- Deleting many rows with the DELETE statement has no effect on the free space in the user RDAREAs, because the segments remain allocated. To create free space, the PURGE TABLE statement must be used.

13.6 Adding a column

Executor: HiRDB administrator and table owner

You add a column to a table by executing `ALTER TABLE` with the `ADD` operand specified.

13.6.1 Preparations for adding a column

(1) Notes on adding a column to a table with the FIX attribute

A column cannot be added to a table with the FIX attribute in which data is stored. If it is necessary to add a column to a table with the FIX attribute in which data is stored, you must use the following procedure:

Procedure

1. Use the `pdrorg` command to unload the table data.
2. Use the `PURGE TABLE` statement to delete the table data.
3. Use the `ALTER TABLE` statement to add a column.
4. Use the `pdhold` command to shut down RDAREAs storing the table.
5. Use the `pdload` command to load the table data.
6. Use the `pdcopy` command to back up the table.
7. Use the `pdrels` command to release the RDAREAs from shutdown status.

Notes

1. Either DAT or binary format can be used for unloading the table's data in step 1. It is preferable to use DAT format because it is a simpler operation.
2. If the table contains character data that cannot be converted to DAT format (0x00 and 0x0a), the table data cannot be unloaded in DAT format; binary format must be used in such a case.
3. For an example of unloading a table in DAT format, see *13.6.5 Example 4: Adding a column to a table with the FIX attribute (unloading in DAT format)*.
4. For an example of unloading a table in binary format, see *13.6.6 Example 5: Adding a column to a table with the FIX attribute (unloading in binary format)*.

(2) Notes

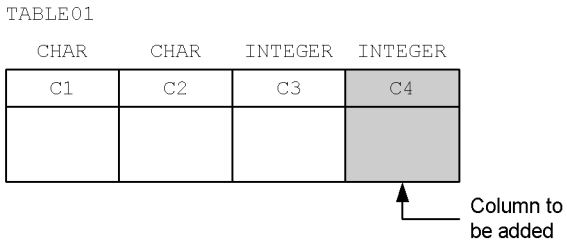
- Columns cannot be added to falsification prevented tables.
- Adding a column to a table invalidates any stored routines that use that table. If

this happens, use the ALTER PROCEDURE or ALTER ROUTINE statement to re-create each stored routine.

- Adding a NOT NULL column to a table specified in a trigger SQL statement invalidates the trigger. If this happens, use the ALTER TRIGGER or ALTER ROUTINE statement to re-create the trigger.
- After the column has been added to the table, execute the optimizing information collection utility (pdgetcst command) if necessary. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.

13.6.2 Example 1: Adding a column to a table without the FIX attribute

This example adds a column (C4) to a table without the FIX attribute (TABLE01).



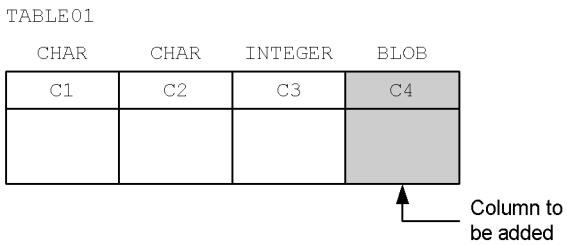
(1) Use ALTER TABLE to add column C4

```
ALTER TABLE TABLE01 ADD C4 INTEGER;
```

In this case, the null value is stored in column C4.

13.6.3 Example 2: Adding a LOB column

This example adds a BLOB type column (C4) to a table (TABLE01). The data for column C4 is stored in user LOB RDAREAs (ULOB01 and ULOB02).



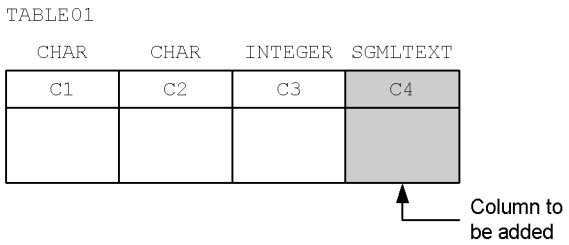
(1) Use ALTER TABLE to add column C4

```
ALTER TABLE TABLE01 ADD C4 BLOB(1M) IN ((ULOB1),(ULOB2));
```

In this case, the null value is stored in column C4.

13.6.4 Example 3: Adding an abstract data type column

A column (C4) of the SGMLTEXT type is added to a table (TABLE01). The data for column C4 is stored in user LOB RDAREAs (ULOB01 and ULOB02).



(1) Use ALTER TABLE to add column C4

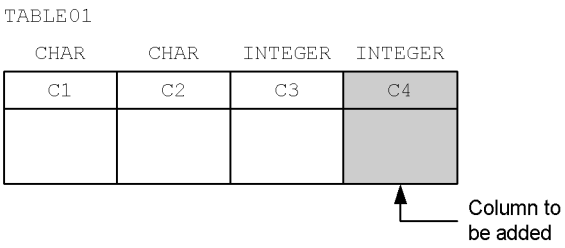
```
ALTER TABLE TABLE01 ADD C4 SGMLTEXT
    ALLOCATE(SGMLTEXT IN ((ULOB1),(ULOB2))
    PLUGIN plug-in-option;
```

```
ALTER TABLE TABLE01 ADD C4 SGMLTEXT
    ALLOCATE(SGMLTEXT IN ((ULOB1),(ULOB2)))
    PLUGIN plug-in-option;
```

In this case, the null value is stored in column C4.

13.6.5 Example 4: Adding a column to a table with the FIX attribute (unloading in DAT format)

This example adds a column (C4) to a table with the FIX attribute (TABLE01).



(1) Use the pdhold command to shut down RDAREAs in which TABLE01 is stored

```
pdhold -r RDAREA1,RDAREA2,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -W dat -t TABLE01 -g /pdrorg/unld01
```

Explanation

`-k`: Specifies `unld` for unloading.

`-W dat`: Specifies that the unload data file is to be used as the input file (DAT format) for the `pdload` command.

`-t`: Specifies the name of the table that is to be unloaded.

`-g`: Specifies that `TABLE01` is row-partitioned between servers in a HiRDB\Parallel Server. Specifying the `-g` option consolidates the unload data files (into a single file).

`/pdrorg/unld01`: Specifies the name of the control statements file for the `pdrorg` command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,...
```

(4) Use the PURGE TABLE statement to delete data from TABLE01

```
PURGE TABLE TABLE01;
```

(5) Use the ALTER TABLE statement to add column C4

```
ALTER TABLE TABLE01 ADD C4 INTEGER WITH DEFAULT;
```

The WITH DEFAULT operand must be specified in this step. In this case, 0 is stored in column C4.

(6) Create the column structure information file (/pdload/column01)

```
C1
C2
C3
```

Because the format of the column structure of the input data file created in step (2) differs from the column structure of TABLE01, a column structure information file is needed for data loading.

Note on specifying a column structure information file

When a column structure information file is specified after a column has been added, it is important not to specify the added column. That way, HiRDB will detect that the input data does not contain any data for that column and will store the default value or the NULL value in the added column. Because the NULL value cannot be stored in a table with the FIX attribute, the default value must be stored in such a case by specifying the WITH DEFAULT operand when the column is added with an ALTER TABLE statement.

(7) Use the pdhold command to shut down RDAREAs in which TABLE01 is stored

```
pdhold -r RDAREA1,RDAREA2,...
```

(8) Load data into TABLE01

```
pdload -c /pdload/column01 TABLE01 /pdload/load01
```

Explanation

-c /pdload/column01: Specifies the name of the column structure information file created in step (6).

TABLE01: Specifies the name of the table into which the data is to be loaded.

`/pdload/load01`: Specifies the name of the control statements file for the `pdload` command.

(9) Back up the RDAREAs in which data was loaded

Because you loaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was loaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

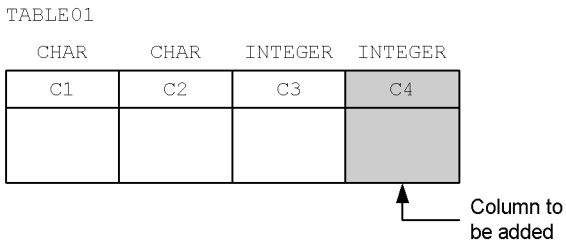
(10) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

13.6.6 Example 5: Adding a column to a table with the FIX attribute (unloading in binary format)

This example adds a column (`C4`) to a table with the FIX attribute (`TABLE01`). It is assumed that the table contains character data (0x00 and 0x0a) that cannot be converted to DAT format.



(1) Use the pdhold command to shut down RDAREAs in which TABLE01 is stored

```
pdhold -r RDAREA1,RDAREA2,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -W bin -t TABLE01 -g /pdrorg/unld01
```

Explanation

-k: Specifies unld for unloading.

-W bin: Specifies that the unload data file is to be used as the input file (binary format) for the pdload command. Because TABLE01 contains character data (0x00 and 0x0a) that cannot be converted to DAT format, the data is converted to binary format and unloaded.

-t: Specifies the name of the table that is to be unloaded.

-g: Specifies that TABLE01 is row-partitioned between servers in a HiRDB/ Parallel Server. Specifying the -g option consolidates the unload data files (into a single file).

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,...
```

(4) Use the PURGE TABLE statement to delete data from TABLE01

```
PURGE TABLE TABLE01;
```

(5) Use the ALTER TABLE statement to add column C4

```
ALTER TABLE TABLE01 ADD C4 INTEGER WITH DEFAULT;
```

The WITH DEFAULT operand must be specified in this step. In this case, 0 is stored in column C4.

(6) Create the column structure information file (/pdload/column01)

```
C1,type=char(4)
C2,type=char(10)
C3,type=integer
```

A column structure information file in which the data types of TABLE01 are specified is created.

Although a binary format file was created in step (2), it is treated as a fixed-size data format file because the table has the FIX attribute. Therefore, a column structure information file necessary for unloading data from a fixed-size data format file is

created. This way, data can be loaded into TABLE01, which has a column structure that is different from the column structure of the input data file.

Note on specifying a column structure information file

When a column structure information file is specified after a column has been added, it is important not to specify the added column. That way, HiRDB will detect that the input data does not contain any data for that column and will store the default value or the NULL value in the added column. Because the NULL value cannot be stored in a table with the FIX attribute, the default value must be set in such a case by specifying the WITH DEFAULT operand when the column is added with an ALTER TABLE statement.

(7) Use the pdhold command to shut down RDAREAs in which TABLE01 is stored

```
pdhold -r RDAREA1,RDAREA2,...
```

(8) Load data into TABLE01

```
pdload -a -c /pdload/column01 TABLE01 /pdload/load01
```

Explanation

-a: Specifies that the input file is a fixed-size data format file.

-c /pdload/column01: Specifies the name of the column structure information file created in step (6).

TABLE01: Specifies the name of the table into which the data is to be loaded.

/pdload/load01: Specifies the name of the control statements file for the pdload command.

(9) Back up the RDAREAs in which data was loaded

Because you loaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was loaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(10) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,...
```

It is recommended that after the command has executed you check whether or not the

execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.7 Deleting a column

Executor: HiRDB administrator and table owner

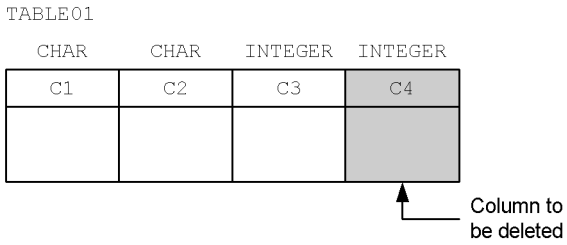
You delete a column from a table by executing `ALTER TABLE` with the `DROP` operand specified.

Notes

1. A column of a table in which data is stored cannot be deleted.
2. A column for which a cluster key is defined cannot be deleted.
3. A LOB column cannot be deleted.
4. If an abstract data type is defined for a table, none of the columns in that table can be deleted.
5. Columns cannot be deleted from falsification prevented tables.
6. Deleting a column from a table invalidates any stored routines that use that table. If this happens, use the `ALTER PROCEDURE` or `ALTER ROUTINE` statement to re-create each stored routine.
7. Deleting a column from a table specified in a trigger SQL statement invalidates the trigger. In addition, deleting a column that is accessed by the trigger action condition or from within an SQL statement using an old and new values correlation name invalidates the trigger. If this happens, use the `ALTER TRIGGER` or `ALTER ROUTINE` statement to re-create the trigger.
8. After the column has been deleted from the table, execute the optimizing information collection utility (`pdgetcst` command) if necessary. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.

13.7.1 Example: Deleting a column

This example deletes a column (`C4`) from a table (`TABLE01`).



(1) Use the pdhold command to shut down RDAREAs in which TABLE01 is stored

```
pdhold -r RDAREA1,RDAREA2,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -W dat -t TABLE01 -g /pdrorg/unld01
```

Explanation

`-k`: Specifies `unld` for unloading.

`-W dat`:

Specifies that the unload data file is to be used as the input file (DAT format) for the `pdload` command.

If the table contains character data (0x00 and 0x0a) that cannot be converted to DAT format, the table data cannot be unloaded in DAT format. In such a case, the data must be unloaded in binary format (`-W bin` specified); note that the operations in the steps beginning with step (6) will be different. See *13.6.6 Example 5: Adding a column to a table with the FIX attribute (unloading in binary format)*.

`-t`: Specifies the name of the table that is to be unloaded.

`-g`: Specifies that `TABLE01` is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the `-g` option consolidates the unload data files (into a single file).

`/pdrorg/unld01`: Specifies the name of the control statements file for the `pdrorg` command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,...
```

(4) Use the PURGE TABLE statement to delete data from TABLE01

```
PURGE TABLE TABLE01;
```

(5) Use the ALTER TABLE statement to delete column C4

```
ALTER TABLE TABLE01 DROP C4;
```

(6) Create the column structure information file (/pdload/column01)

```
C1
C2
C3
*skipdata*
```

Because the format of the column structure of the input data file created in step (2) is different from the column structure of `TABLE01`, a column structure information file is needed for data loading.

(7) Use the pdhold command to shut down RDAREAs in which TABLE01 is stored

```
pdhold -r RDAREA1,RDAREA2,...
```

(8) Load data into TABLE01

```
pdload -c /pdload/column01 TABLE01 /pdload/load01
```

Explanation

`-c /pdload/column01`: Specifies the name of the column structure information file created in step (6).

`TABLE01`: Specifies the name of the table into which the data is to be loaded.

`/pdload/load01`: Specifies the name of the control statements file for the `pdload` command.

(9) Back up the RDAREAs in which data was loaded

Because you loaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was loaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(10) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA1,RDAREA2,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.8 Modifying a table's definition

Executor: Table owner

You modify a table's definition by executing `ALTER TABLE` with the `CHANGE` operand specified. The following definitions can be modified:

- Data size of a variable-length character string
- Change from `CHAR` to `MCHAR`
- Maximum number of elements in repetition columns
- Changing the non-NULL value constraint without a default value to one with a default value
- Changing a cluster key without the uniqueness constraint to one with the uniqueness constraint
- Changing a cluster key with the uniqueness constraint to one without the uniqueness constraint
- Minimum unit of locked resources for the table
- Hash function

Notes

1. The definition of a LOB column cannot be modified.
2. The definition of an abstract data type column cannot be modified.
3. Changing the definition of a table invalidates any stored routines that use that table. If this happens, use the `ALTER PROCEDURE` or `ALTER ROUTINE` statement to re-create each stored routine.
4. Changing the definition of a table specified in a trigger SQL statement invalidates the trigger. In addition, deleting a column that is accessed by the trigger action condition or from within an SQL statement using an old and new values correlation name invalidates the trigger. If the trigger becomes invalid, use the `ALTER TRIGGER` or `ALTER ROUTINE` statement to re-create the trigger.
5. After a table's definition has been changed, execute the optimizing information collection utility (`pdgetcst` command) if necessary. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.

13.8.1 Example: Changing the data size of a column

This example changes the definition size of the `DATA01` column in the `TABLE01` table

from VARCHAR(150) to VARCHAR(200).

(1) Use the ALTER TABLE statement to change the definition of column DATA01

```
ALTER TABLE TABLE01 CHANGE DATA01 VARCHAR(200);
```

13.9 Changing a table name or column name

Executor: Table owner

You modify a table name or column name by executing `ALTER TABLE` with the `RENAME` operand specified.

Note

1. The table name or column name of a falsification prevented table cannot be changed.
2. Changing a table name or column name invalidates any stored routines that use that table. If this happens, use the `ALTER PROCEDURE` or `ALTER ROUTINE` statement to re-create each stored routine.
3. Changing a table name or column name of a table specified in a trigger SQL statement invalidates the trigger. If this happens, use the `ALTER TRIGGER` or `ALTER ROUTINE` statement to re-create the trigger.
4. The table name of a table that defines a trigger cannot be changed.
5. The column names of the following columns cannot be changed in a table that defines a trigger:
   - Trigger event column
   - Column that is referenced by a trigger action condition using an old and new values correlation name
   - Column that is referenced by a trigger SQL statement using an old and new values correlation name

13.9.1 Example 1: Changing a table name

This example changes the name of a table from `TABLE01` to `TABLE02`.

(1) Use the ALTER TABLE statement to change a table's name

```
ALTER TABLE TABLE01 RENAME TABLE TO TABLE02;
```

13.9.2 Example 2: Changing a column name

This example changes the name of column `C1` in table `TABLE01` to `C2`.

(1) Use the ALTER TABLE statement to change a column's name

```
ALTER TABLE TABLE01 RENAME COLUMN FROM C1 TO C2;
```

13.10 Increasing the number of table row partitions

Executor: HiRDB administrator and table owner (or user with DBA privilege)

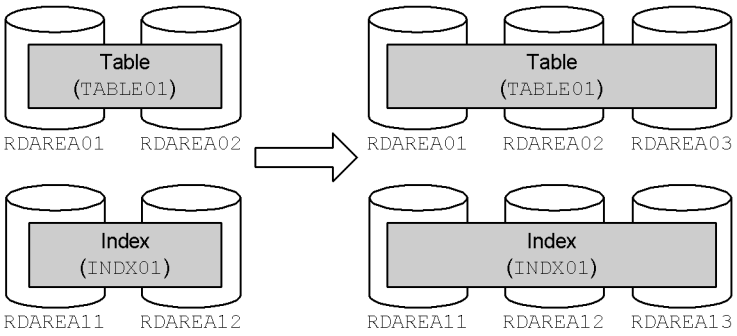
When the amount of data in a table increases significantly because data is added repeatedly, processing performance for the table may become degraded. In such a case, it is advisable to increase the number of table row partitions.

Notes

1. In the case of a table with key range partitioning, first use `DROP TABLE` to delete the table, then use `CREATE TABLE` to redefine the table. While you are redefining the table, you can increase the number of table row partitions.
2. In a table with hash partitioning, you use `ALTER TABLE` to increase the number of table row partitions.
3. Increasing the number of row partitions in a table invalidates any stored routines that use that table. If this happens, use the `ALTER PROCEDURE` or `ALTER ROUTINE` statement to re-create each stored routine.
4. Increasing the number of row partitions in a table specified in a trigger SQL statement invalidates the trigger. If this happens, use the `ALTER TRIGGER` or `ALTER ROUTINE` statement to re-create the trigger.
5. After the number of table row partitions has been increased, execute the optimizing information collection utility (`pdgetcst` command) if necessary. Note, however, that this utility cannot be applied to an abstract data type. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.

13.10.1 Example 1: Increasing the number of row partitions in a table with key range partitioning

This example increases from 2 to 3 the number of row partitions for table `TABLE01`. Its index (`INDX01`) is partitioned likewise. `TABLE01` is key-range partitioned.



(1) Use the pdhold command to shut down RDAREAs to be unloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -j -t TABLE01 -g /pdrorg/unld01
```

Explanation

`-k`: Specifies `unld` for unloading.

`-j`: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be unloaded.

`-t`: Specifies the name of the table that is to be unloaded.

`-g`: Specifies that `TABLE01` is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the `-g` option consolidates the unload log data (into a single file).

`/pdrorg/unld01`: Specifies the name of the control statements file for the `pdrorg` command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

(4) Use the DROP TABLE statement to delete TABLE01

```
DROP TABLE TABLE01;
```

(5) Use the CREATE TABLE and CREATE INDEX statements to redefine TABLE01 and INDEX01, respectively

```
CREATE TABLE TABLE01 ...
    IN ((RDAREA01),(RDAREA02),(RDAREA03)) ...;
CREATE INDEX INDX01 ...
    IN ((RDAREA11),(RDAREA12),(RDAREA13));
```

(6) Use the pdhold command to shut down RDAREAs to be reloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(7) Use the pdrorg command to reload data into TABLE01

```
pdrorg -k reld -j -t TABLE01 -g /pdrorg/reld01
```

Explanation

`-k`: Specifies `reld` for reloading.

`-j`: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be reloaded.

`-t`: Specifies the name of the table that is to be reloaded.

`-g`: If the `-g` option was specified in step (2), specify it here as well.

`/pdrorg/reld01`: Specifies the name of the control statements file for the `pdrorg` command.

(8) Back up the RDAREAs in which data was reloaded

Because you reloaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was reloaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(9) Use the pdrels command to release RDAREAs from shutdown status

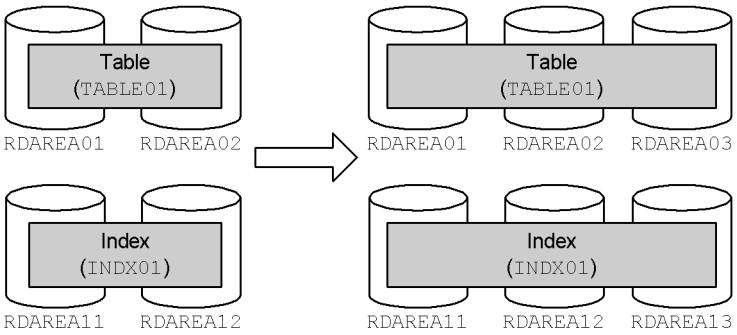
```
pdrels -r RDAREA01,RDAREA02,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

13.10.2 Example 2: Increasing the number of row partitions in a table with flexible hash partitioning

This example increases from 2 to 3 the number of row partitions for table TABLE01. Its index (INDEX01) is partitioned likewise. TABLE01 is flexible hash partitioned.

It is assumed that this table does not use the hash facility for hash row partitioning.



(1) Use the ALTER TABLE statement to add RDAREA03 to TABLE01

```
ALTER TABLE TABLE01 ADD RDAREA RDAREA03;
```

In this step, no data is stored in RDAREA03. Data that will be added subsequently will be stored using a hash function.

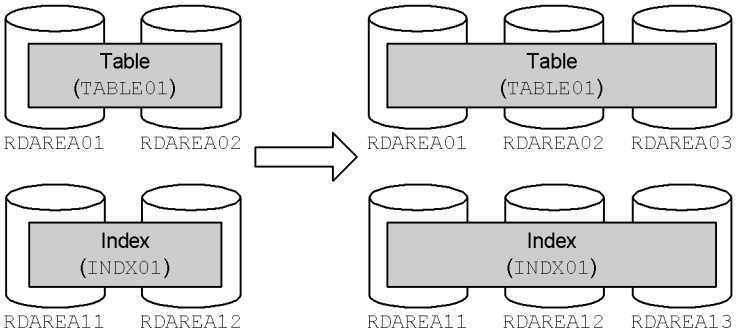
When the hash facility for hash row partitioning is used, data will be stored in RDAREA03, and this will prevent an imbalance in the amount of data stored in the existing RDAREAs and in the newly added RDAREA. For an example of the hash facility for hash row partitioning, see *13.11 Increasing the number of table row partitions (using the hash facility for hash row partitioning)*.

13.10.3 Example 3: Increasing the number of row partitions in a table with FIX hash partitioning

This example increases from 2 to 3 the number of row partitions in table TABLE01. Its index (INDEX01) is partitioned likewise. TABLE01 is FIX hash partitioned.

It is assumed that this table does not use the hash facility for hash row partitioning. When this hash facility is used, an RDAREA can be added with an ALTER TABLE statement even if the table contains data. Consequently, there is no need to unload and

reload the table data. For an example of the hash facility for hash row partitioning, see *13.11 Increasing the number of table row partitions (using the hash facility for hash row partitioning)*.



(1) Use the pdhold command to shut down RDAREAs to be unloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -j -t TABLE01 -g /pdrorg/unld02
```

Explanation

`-k`: Specifies `unld` for unloading.

`-j`: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be unloaded.

`-t`: Specifies the name of the table that is to be unloaded.

`-g`: Specifies that `TABLE01` is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the `-g` option consolidates the unload data files (into a single file).

`/pdrorg/unld02`: Specifies the name of the control statements file for the `pdrorg` command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

(4) Use the PURGE TABLE statement to delete TABLE01's data

```
PURGE TABLE TABLE01;
```

To increase the number of row partitions in a table with FIX hash partitioning, the data in the table must be deleted.

(5) Use the ALTER TABLE statement to add RDAREA03 to TABLE01

```
ALTER TABLE TABLE01 ADD RDAREA RDAREA03;
```

(6) Use the pdhold command to shut down RDAREAs to be reloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(7) Use the pdrorg command to reload data into TABLE01

```
pdrorg -k reld -j -t TABLE01 -g /pdrorg/reld02
```

Explanation

`-k`: Specifies `reld` for reloading.

`-j`: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be reloaded.

`-t`: Specifies the name of the table that is to be reloaded.

`-g`: If the `-g` option was specified in step (2), specify it here as well.

`/pdrorg/reld02`: Specifies the name of the control statements file for the `pdrorg` command.

(8) Back up the RDAREAs in which data was reloaded

Because you reloaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was reloaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(9) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.11 Increasing the number of table row partitions (using the hash facility for hash row partitioning)

Executor: HiRDB administrator and table owner

This section explains how to increase the number of row partitions using the hash facility for hash row partitioning. The following items are explained here:

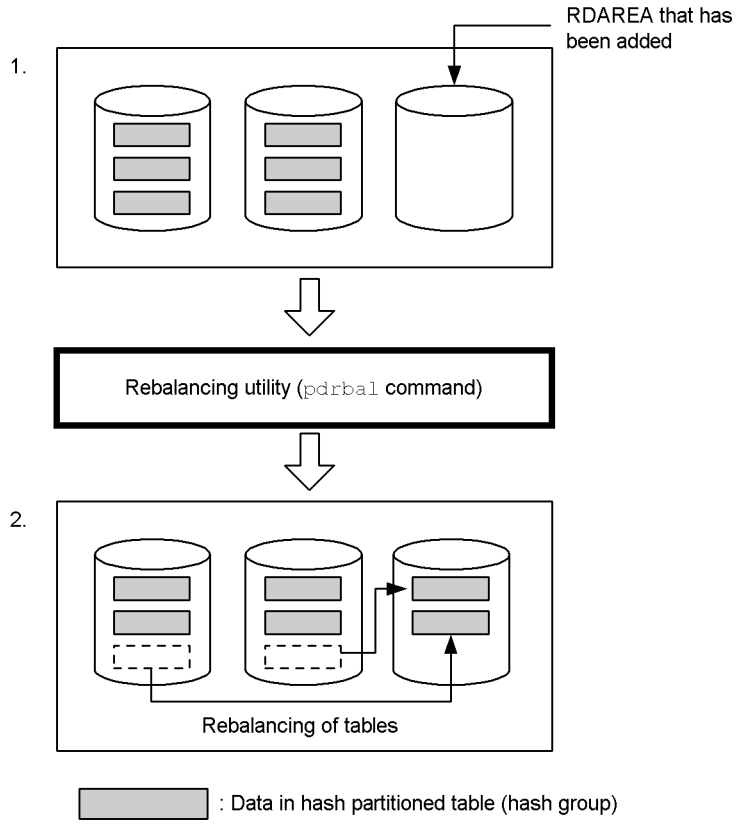
- Overview of hash facility for hash row partitioning
- Preparations for using hash facility for hash row partitioning
- Example: Increasing the number of row partitions in a rebalancing table
- Using the rebalancing utility (when table rebalancing takes time)
- Notes on a table with FIX hash partitioning

13.11.1 Overview of the hash facility for hash row partitioning

When the data volume in a table with hash partitioning has increased and an RDAREA is added (i.e., when the number of table row partitions is increased), an imbalance will occur between the amount of data in the existing RDAREAs and the amount of data in the newly added RDAREA. When the *hash facility* (*rebalancing facility) for hash row partitioning* is used, such an imbalance in the amount of data in the RDAREAs can be corrected. Figure 13-11 shows the hash facility for hash row partitioning.

Note that the hash facility for hash row partitioning can be applied to both FIX hashing and flexible hashing.

Figure 13-11: Hash facility for hash row partitioning



Explanation

1. Because data has filled the hash partitioned table, an RDAREA for storing the hash partitioned table is added (the number of table row partitions is increased). No data is placed in the added RDAREA, resulting in a data volume imbalance.
2. The rebalancing utility (`pdrbal` command) is executed to correct the data volume imbalance. Executing this utility relocates data by moving it by hash group. This is called table rebalancing.

   When the hash facility for hash row partitioning is used, HiRDB divides the data into 1024 groups (called a hash group) based on the hashing result of the partitioning key. RDAREA segments are allocated to these groups to store the data. Data relocating is also carried out using the hash group as the unit.

(1) Application criteria

- When the data volume is expected to increase and RDAREAs are likely to be added in the future#
- When it is difficult to re-create a table because of a large amount of data

#: An RDAREA cannot be added to a table with FIX hash partitioning if data is stored in it (this applies to a table with FIX hash partitioning that uses one of the hash functions `HASH1` to `HASH6`). However, with the rebalancing facility, an RDAREA can be added to a table with FIX hash partitioning that uses one of the hash functions `HASHA` to `HASHF`.

(2) Note

The amount of data in each hash group depends on the hashing result of the hash function. Therefore, if the partitioning key values are not distributed evenly, the data volume will also be uneven among the hash groups, and it may not be possible to divide the data evenly among the RDAREAs.

(3) Procedure

Following is an overview of the procedure for applying the hash facility for hash row partitioning:

Procedure

To apply the has facility:

1. When a hash partitioned table is defined, define it as a *rebalancing table*.
2. To increase the number of table row partitions, add an RDAREA to be used to store the table's data.
3. Rebalance the table by executing the rebalancing utility.#

#: The execution modes shown in Table 13-2 are available for the rebalancing utility.

Table 13-2: Rebalancing utility execution modes

Execution mode	Explanation
Shared mode	Table can be accessed while it is being rebalanced. In a very large database, table rebalancing may take as long as several days. Access to the table can be permitted while rebalancing is underway. However, because table access and table rebalancing are being executed simultaneously, the efficiency of both processes is degraded. To minimize such degradation, it is possible to divide the rebalancing process into multiple segments to be executed at times when the operational workload is low.

Execution mode	Explanation
Exclusive mode	Table cannot be accessed while it is being rebalanced. Because only rebalancing is executed, processing efficiency is higher than in the shared mode. If it is possible to suspend accesses to the table, executing the rebalancing utility in the exclusive mode is recommended. The rebalancing process can be divided into multiple segments in the exclusive mode as well.

13.11.2 Preparations for using the hash facility for hash row partitioning

(1) Defining a rebalancing table

You must specify in `CREATE TABLE` one of `HASHA` to `HASHF` as the hash function. A table for which one of these hash functions is specified is called a *rebalancing table*. For details on the hash functions `HASHA` to `HASHF`, see the manual *HiRDB Version 8 SQL Reference*.

```
CREATE TABLE TABLE01
    (PCODE DEC(5) NOT NULL,PNAME NCHAR(15), ... ) HASH HASHF BY PCODE
     IN (RDAREA01,RDAREA02);
CREATE INDEX INDX01 ...
     IN ((RDAREA11),(RDAREA12));
```

(2) RDAREAs for storing a rebalancing table

1. RDAREAs in which a rebalancing table is stored can accommodate only the rebalancing table; other tables or indexes cannot be stored in these RDAREAs.
2. An RDAREA for storing a rebalancing table requires the following number of segments: ↑*1024* ÷ *partitioning-RDAREAs-count*↑. This condition must be satisfied when the RDAREAs are created.
3. In estimating the size of an RDAREA, you should add some margin to the number of segments computed from ↑*1024* ÷ *partitioning-RDAREAs-count*↑.
4. Unused segments are needed in order to store data in a rebalancing table. For this reason, the RDAREA full error will result even if the RDAREA contains free pages. Therefore, you should manage the remaining space in the RDAREA by number of unused segments, rather than by number of free pages.

(3) Defining an index for a rebalancing table

- An RDAREA for storing a rebalancing table can store only that rebalancing table; other tables or indexes cannot be stored. Therefore, when `CREATE INDEX` is specified for a rebalancing table, the names of the RDAREAs for storing the index cannot be omitted. If they are omitted, an RDAREA that stores the rebalancing

table will be assumed, resulting in an error. An RDAREA that stores the rebalancing table must not be specified for an index.

- A cluster key index cannot be defined.

(4) Global buffer

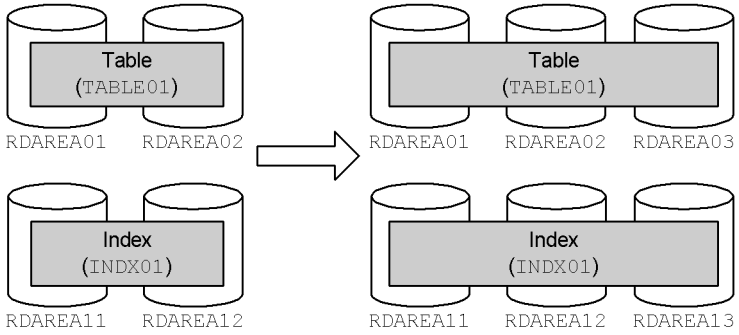
For each RDAREA that is storing the table, you must set the global buffer sectors count to at least ↑*1024* ÷ *partitioning-RDAREAs-count*↑ × *2*. Otherwise, the effects of buffering will not be realized.

13.11.3 Example: Increasing the number of row partitions in a rebalancing table

This example increases from 2 to 3 the number of row partitions in rebalancing table `TABLE01`. Its index (`INDX01`) is partitioned likewise.

Note

In order to execute the rebalancing utility, the number of RDAREAs storing indexes must match the number of RDAREAs storing tables.



(1) Use the pdfmkfs command to create a HiRDB file system area for RDAREAs

```
pdfmkfs -n 50 -l 10 -k DB -i /rdarea/area03
pdfmkfs -n 50 -l 10 -k DB -i /rdarea/area13
```

Explanation

`-n`: Specifies in megabytes the size of the HiRDB file system area. Use the database condition analysis utility (`pddbst` command) to analyze the status of the existing RDAREAs, and, taking existing RDAREAs into account, estimate the size of the RDAREAs to be added from the volume of data to be migrated.

`-l`: Specifies the maximum number of HiRDB files to be created in the HiRDB file system area.

-k: Specifies the purpose of the HiRDB file system area. Specify DB to create a HiRDB file system area for RDAREAs.

-i: Specifies that the entire HiRDB file system area is to be initialized. When the -i option is specified, the entire area is allocated. When the -i option is omitted, only management information for the HiRDB file system area is created.

/rdarea/area03: Specifies a name for the HiRDB file system area that is to be created for RDAREAs to be used to store tables.

/rdarea/area13: Specifies the name of the HiRDB file system area that is to be created for RDAREAs to be used to store indexes.

(2) Use the pdmod command to add RDAREAs

```
pdmod -a /pdmod/create01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command. Below is an example of a control statements file. For details on the operands, see the manual *HiRDB Version 8 Command Reference*.

```
create rdarea RDAREA03
   globalbuffer gbuf03
   for user used by PUBLIC
   server name bes1
   page 4096 characters
   storage control segment 10 pages
   file name "/rdarea/area03/file01"
      initial 1000 segments;
create rdarea RDAREA13
   globalbuffer gbuf13
   for user used by PUBLIC
   server name bes1
   page 4096 characters
   storage control segment 10 pages
   file name "/rdarea/area13/file01"
      initial 1000 segments;
```

(3) Use the ALTER TABLE statement to increase the number of table row partitions

```
ALTER TABLE TABLE01
    ADD RDAREA RDAREA03
      FOR INDEX INDX01 IN RDAREA13;
```

An RDAREA for storing table data (RDAREA03) and an RDAREA for storing indexes (RDAREA13) have been added. The number of row partitions in TABLE01 has now been increased from 2 to 3.

(4) Use the pdrbal command to rebalance the table

(a) Execution in shared mode

```
pdrbal -k share -t TABLE01 -c 100 /pdrbal/cfile01
```

Explanation

-k: Specifies share for execution in the shared mode.

-t: Specifies the name of the rebalancing table.

-c: Specifies the unit of commitment.

/pdrbal/cfile01: Specifies the name of the control statements file for the pdrbal command. The following are the contents of the control statements file:

```
report /tmp/output
execstop time,9:00
```

Explanation

report: Specifies the output destination for the execution results of the pdrbal command.

execstop: Specifies the execution time for the pdrbal command.

(b) Execution in exclusive mode

```
pdrbal -k exclusive -t TABLE01 -l a /pdrbal/cfile01
```

Explanation

-k: Specifies exclusive for execution in the exclusive mode.

-t: Specifies the name of the rebalancing table.

-l: Specifies the method for collecting the database updates log. In this example, the log acquisition mode is specified.

/pdrbal/cfile01: Specifies the name of the control statements file for the pdrbal command. The following are the contents of the control statements file:

```
report /tmp/output
idxwork /index/work
sort /sort/work
```

Explanation

`report`: Specifies the output destination for the execution results of the `pdrbal` command.

`idxwork`: Specifies the name of the directory for output of the index information file.

`sort`: Specifies the name of the work directory for sorting.

(5) Use the pdgetcst command to collect optimization information for the table

If necessary, use the optimizing information collection utility (`pdgetcst` command) to collect optimization information for the table. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.

```
pdgetcst -t TABLE01 -l /pdgetcst/output
```

Explanation

`-t`: Specifies the name of the table for which optimization information is to be collected.

`-l`: Specifies the output destination for the execution results of the `pdgetcst` command.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.11.4 Using the rebalancing utility (when table rebalancing takes time)

This section explains how to use the rebalancing utility when table rebalancing (processing by the rebalancing utility) takes time.

(1) Shared mode

When the shared mode is specified, accesses to the table being rebalanced are permitted while rebalancing is being executed. This mode is effective when online operations must continue around the clock; the explanation here assumes around-the-clock operations.

Guidelines on usage

- Execute the rebalancing utility at times when the operational workload is relatively light.
- If the rebalancing utility's processing cannot be completed in a single operation, divide the task into multiple segments for execution of the rebalancing utility.

Application example

Because the workload is relatively light between 21:00 and 9:00, specify that the rebalancing utility will be executed at 21:00, and specify with the `execstop time` operand in the control statements file that the utility is to execute for up to 12 hours.

Example of command

```
pdrbal -k share -t TABLE01 -c 100 /pdrbal/cfile01
```

Contents of control statements file

```
execstop time,12:00
```

Return code

If rebalancing processing is not completed within 12 hours (is still executing at 9:00), the rebalancing utility terminates with return code 4 and the rebalancing processing terminates. Execute the rebalancing utility again at 21:00. Repeat each day until rebalancing is completed and return code 0 is returned.

(2) Exclusive mode

When the exclusive mode is specified, the table being rebalanced cannot be accessed while the rebalancing utility is executing.

Guidelines on usage

- Execute the rebalancing utility at times when accesses to the rebalancing table can be suspended.
- If the rebalancing utility's processing cannot be completed in a single operation, divide the task into multiple segments for execution of the rebalancing utility.

Application example

Because accesses to the table to be rebalanced can be suspended between 23:00

and 5:00, specify that the rebalancing utility will be executed at 23:00, and specify with the `execstop time` operand in the control statements file that the utility is to execute for up to 6 hours.

Example of command

```
pdrbal -k exclusive -t TABLE01 /pdrbal/cfile01
```

Contents of control statements file

```
execstop time,6:00
```

Return code

If rebalancing processing is not completed within 6 hours (is still executing at 5:00), the rebalancing utility terminates with return code 4 and the rebalancing processing terminates. Execute the rebalancing utility again at 23:00. Repeat each day until rebalancing is completed and return code 0 is returned.

Notes on defining an index

If an index defined for a table is processed in the batch index creation mode, the rebalancing utility may not terminate at the time specified in the `execstop time` operand. This is because the rebalancing utility cannot stop until index creation is completed. Therefore, for processing the index in the batch index creation mode, increase the value specified for the `execstop time` operand to provide extra time.

13.11.5 Notes on a table with FIX hash partitioning

When the hash facility for hash row partitioning is applied to a table with FIX hash partitioning, the performance of the following retrieval processes will decline while table rebalancing is being executed:

- Retrieval in which a condition is specified for a partitioning key
- Retrieval in which all partitioning keys are included in the columns specified as the `GROUP BY` clause

An index for which `UNIQUE` is specified cannot be updated while rebalancing is being executed.

13.12 Changing a table's partitioning storage conditions

Executor: HiRDB administrator and table owner

You can use ALTER TABLE to change the partitioning storage conditions for a table that was row-partitioned using key range partitioning.# Changing a table's partitioning storage conditions enables you to reuse RDAREAs that have been storing obsolete data, thus reducing your workload because you do not have to delete tables and re-create them.

#

ALTER TABLE is used to change a table's partitioning storage conditions when the following partitioning methods have been used:

- Boundary value specification
- Storage condition specification (only when the equal sign (=) is used as the comparison operator in the storage conditions)

HiRDB Advanced Partitioning Option must be installed in order to change a table's partitioning storage conditions.

13.12.1 Purpose of changing partitioning storage conditions

The amount of data stored in a row-partitioned table increases over time. As a result, it becomes desirable to erase old data. However, if the data registration date/time, for example, is specified as the partitioning key, simply deleting old data cannot make the RDAREA that was storing the old data available for reuse. In the past, to reuse an RDAREA that was storing old data, you had to perform the following operations:

1. Unload all data from the table that was to be modified.
2. Delete the table that was to be modified.
3. Specify new partitioning storage conditions and re-create the table.
4. Reload the data into the re-created table.

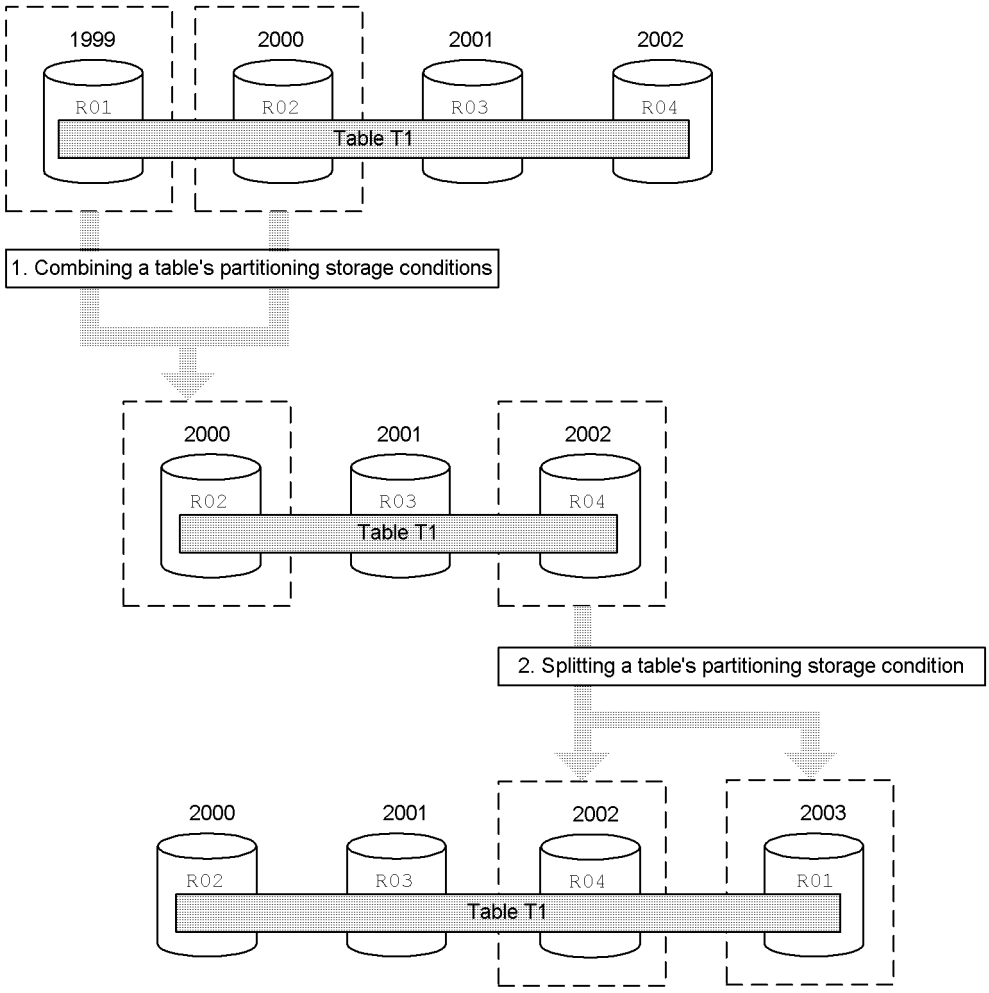
While these operations were being performed, jobs that access the table being modified had to be stopped temporarily. Because all data had to first be unloaded from the table that was to be modified, and then had to be reloaded, a long time was required for these operations, and the impact on jobs that were stopped could be significant.

Now, by using the ALTER TABLE to change the partitioning storage conditions of a row-partitioned table, you can reduce the amount of time it takes to make an RDAREA reusable.

(1) Overview of changing partitioning storage conditions (in the case of boundary value specification)

Figure 13-12 provides an overview of changing partitioning storage conditions (in the case of boundary value specification).

Figure 13-12: Overview of changing partitioning storage conditions (in the case of boundary value specification)



Explanation

1. The combine facility combines the 1999 and 2000 storage conditions (definition information), which are stored in RDAREAs `R01` and `R02`, respectively, and stores them in RDAREA `R02`. At this time, the data from

R01 can be deleted.

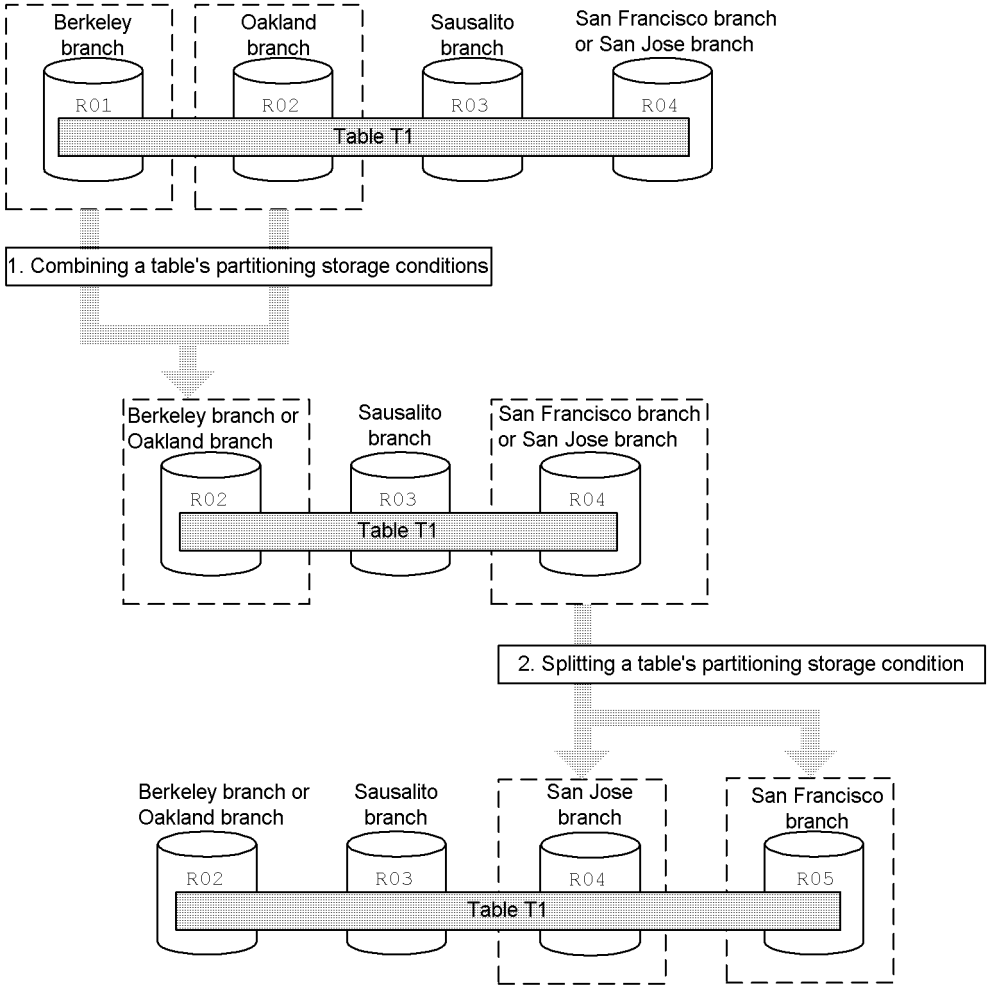
2. The split facility splits the storage condition (definition information) for 2002 into the data for 2002 and the data for 2003. As a result, the 2002 data is stored in R04 and the data for 2003 and beyond can be stored in R01, which previously stored the data for 1999, thereby reusing R01.

Therefore, in step 1, all that is needed is to unload the data from R01 and R02 and then, after the partitioning storage conditions have been changed, to reload the data from R02 only. In step 2, all that is needed is to unload the data from R04 and load data into R04 and R01 after the partitioning storage conditions have been changed. It was also possible to delete the contents of R01 in step 1. If it turned out that R04 did not contain any data for 2003 and beyond, there would be no need for unloading and reloading, and thus the RDAREA configuration could be modified quite quickly. This makes it possible to recycle and reuse RDAREAs that store data that increases in volume as new ascending key values are added.

(2) Overview of changing partitioning storage conditions (in the case of storage condition specification)

Figure 13-13 provides an overview of changing partitioning storage conditions (in the case of storage condition specification).

Figure 13-13: Overview of changing partitioning storage conditions (in the case of storage condition specification)



Explanation

The table is row-partitioned with a storage condition specification that uses the branch name as the key value. When branches need to be deleted or added due to restructuring, data for a single branch can be split into multiple branches or data for multiple branches can be combined into a single branch.

1. Because the Berkeley and Oakland branches are combined, the data from these branches is combined in RDAREA `R02` and the data in RDAREA `R01` is deleted.

2. Because the San Jose branch is split from the San Francisco branch, the data in RDAREA `R04` is split. The data for the San Jose branch is stored in `R04` and the data for the San Francisco branch is stored in RDAREA `R05`.

Hint:

- In *1.*, data must be unloaded only from `R01` and `R02`. There is no need to unload data from `R03` or `R04`.
- In *2.*, data must be unloaded only from `R04`. There is no need to unload data from `R02` or `R03`.

13.12.2 Facilities used to change partitioning storage conditions

This subsection describes the facilities used to change a table's partitioning storage conditions.

(1) Split facility

The split facility changes (splits) a table partitioning storage condition and stores the data from a single RDAREA into multiple RDAREAs. Figures 13-14 and 13-15 provide an overview of the split facility.

Figure 13-14: Overview of the split facility (in the case of boundary value specification)

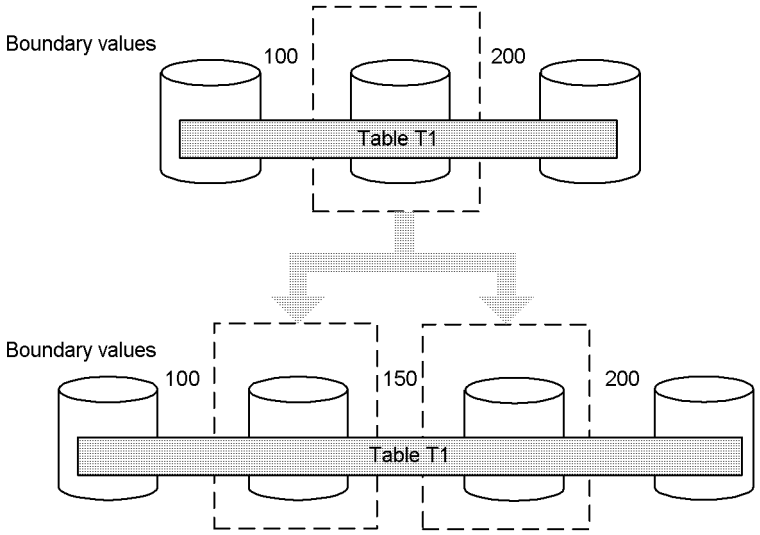
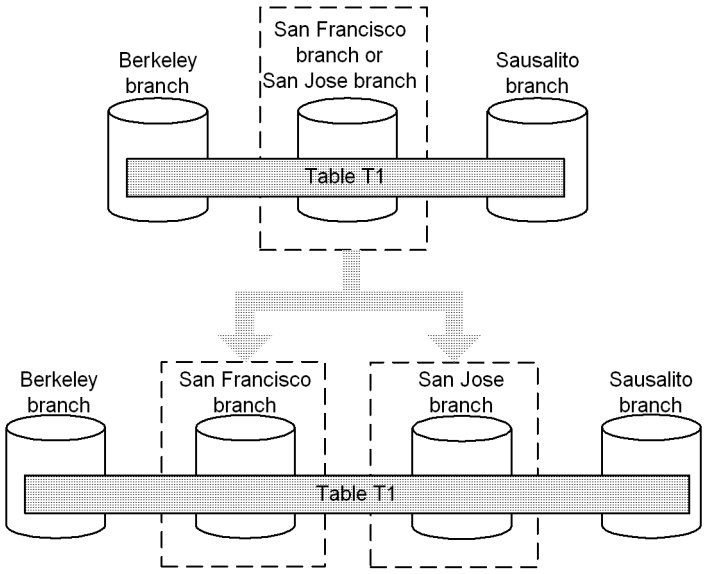


Figure 13-15: Overview of the split facility (in the case of storage condition specification)



When a table is split, the data stored in the RDAREA that is to be split is deleted. Therefore it is necessary to first unload the data from the RDAREA that stores data under the old partitioning storage conditions, and then reload this data into the RDAREAs that will store the data under the new partitioning storage conditions. However, there is one exception: if the pre-splitting RDAREA is to be used as a post-splitting RDAREA without any change, you can keep the data that was stored before splitting without unloading and reloading it.

If you do not use the pre-splitting RDAREA as a post-splitting RDAREA, you cannot keep the data that was stored in the pre-splitting RDAREA.

(2) Combine facility

The combine facility changes (combines) partitioning storage conditions for a table and stores data from multiple RDAREAs into a single RDAREA. Figures 13-16 and 13-17 provide an overview of the combine facility.

Figure 13-16: Overview of the combine facility (in the case of boundary value specification)

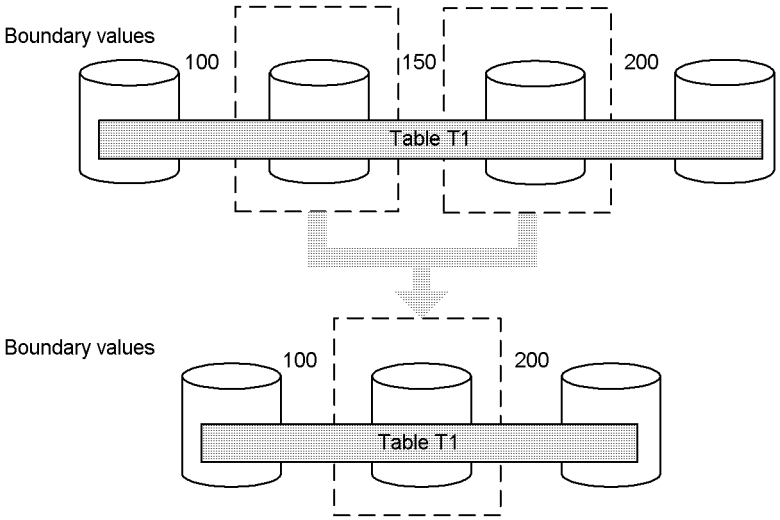
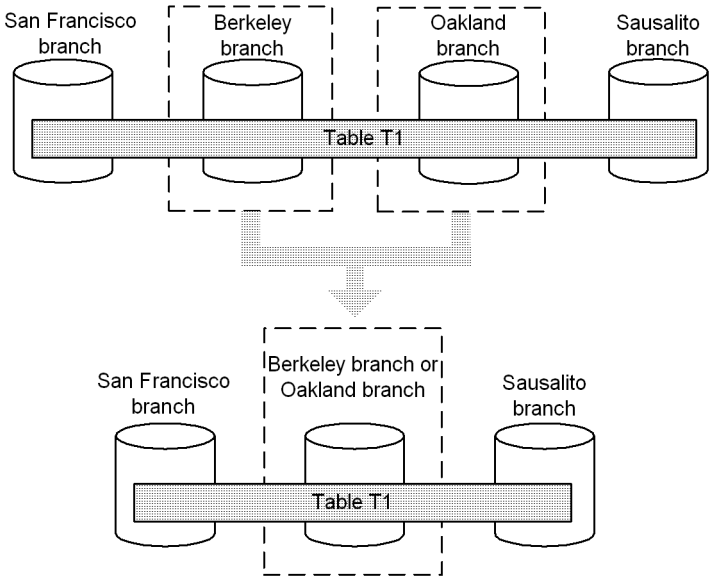


Figure 13-17: Overview of the combine facility (in the case of storage condition specification)



When RDAREAs are combined, the data stored in the RDAREAs that are to be combined is deleted. Therefore it is necessary to first unload the data from the RDAREAs to be combined that store data under the old partitioning storage

conditions, and then to reload this data into the RDAREA that will store the data under the new, combined partitioning storage conditions. However, there is one exception: if one of the pre-combination RDAREAs is to be used as the post-combination RDAREA, you can keep the data that was stored in it before combining.

If you do not use a pre-combining RDAREA as the post-combining RDAREA, you cannot keep the data that was stored in any of the pre-combining RDAREAs. To set the data that was stored in the pre-combining RDAREAs in the newly added RDAREA, you must unload the data from all RDAREAs before combining and then reload this data after combining.

13.12.3 Prerequisites

(1) Table partitioning method

Key range partitioning is the prerequisite. Table 13-3 shows for each table partitioning method whether or not the partitioning storage conditions can be changed.

Table 13-3: Table partitioning methods for which partitioning storage conditions can be changed

Table partitioning method			Applicability
Key range partitioning	Storage condition specification	Only = is used as the comparison operator for storage conditions	Y
		A condition other than = is used as the comparison operator for storage conditions	N
	Boundary values specification	Other than below	Y
		Matrix partitioning	N
Hash partitioning (including rebalancing)	FIX hash not specified		N#
	FIX hash specified		

Legend:

Y: Can be changed.

N: Cannot be changed.

#

Only RDAREA addition is supported by the `ALTER TABLE` definition SQL statement with `ADD RDAREA` specified.

(2) Applicability by table type

Partitioning storage conditions cannot be changed for the following types of tables:

- Non-partitioned table: Because there are no partitioning storage conditions for a non-partitioned table, change is not possible.
- Falsification-prevented table: Depending on the specified change in the partitioning storage conditions, data in the target table in the RDAREA may be deleted. Because data deletion is the same as data falsification, the partitioning storage conditions for a falsification prevented table cannot be changed.
- Table containing an abstract data type: The partitioning storage conditions of a table containing an abstract data type cannot be changed.

(3) Required products

To change partitioning storage conditions, HiRDB Advanced Partitioning Option must be installed. If you attempt to change the partitioning storage conditions (`CHANGE RDAREA` of `ALTER TABLE`) and this product has not been set up, the `KFPA11948-E` message will be displayed and `ALTER TABLE` will terminate with an error.

(4) Other applicability issues

(a) Indexes

If an index has been defined for a table whose partitioning storage conditions are to be changed, it may not be possible to change the partitioning storage conditions depending on the index definition conditions. Table 13-4 shows for each type of index whether or not the partitioning storage conditions can be changed.

Table 13-4: Index types for which partitioning storage conditions can be changed

Index type	Partitioned/Non-partitioned index	Applicability
Cluster key index	Partitioned key index	Y#1
Primary key index (including an index for which a primary key and cluster key are defined)		
B-tree index	Partitioned key index	Y#1
	Non-partitioned key index	Y#1, #2
Plug-in index	Not applicable	N

Legend:

Y: Can be changed.

N: Cannot be changed.

#1

Depending on the partitioning conditions for the index storage RDAREAs, it may not be possible to change the partitioning storage conditions. Table *13-5 Whether or not the partitioning storage conditions can be changed depending on the partitioning conditions for the index storage RDAREAs* shows whether or not the partitioning storage conditions can be changed depending on the partitioning conditions for the index storage RDAREAs.

#2

If the index storage RDAREAs have a 1-to-1 partition correspondence with the table storage RDAREAs, the partitioning storage conditions can be changed. Table *13-6 Applicability of changing partitioning storage conditions for a non-partitioning key index* shows whether or not change to the partitioning storage conditions is applicable for a non-partitioning key index.

Table 13-5: Whether or not the partitioning storage conditions can be changed depending on the partitioning conditions for the index storage RDAREAs

Partitioning conditions for the index storage RDAREAs			Whether or not the partitioning storage conditions can be changed	
			Split	Combine
Partitioning key index	Index storage RDAREAs are partitioned.		Y	Y#1
	Index storage RDAREAs are not partitioned.		N#2	N#3
Non-partitioning key index	Partitioning is performed in such a manner that the table storage RDAREAs and index storage RDAREAs have a 1-to-1 correspondence.	Index storage RDAREAs are partitioned.	Y	Y#1
		Index storage RDAREAs are not partitioned.	N#2	N#3
	Other		N	N

Legend:

Y: Partitioning storage conditions can be changed.

N: Partitioning storage conditions cannot be changed.

#1

If the total number of index storage RDAREAs is reduced after combining to a single index storage RDAREA, the partitioning storage conditions can no longer be changed because the row-partitioned index has been converted to a

non-row-partitioned index.

#2

The partitioning storage conditions cannot be changed because the non-row-partitioned index is converted to a row-partitioned index.

#3

A row-partitioned index can be changed to a non-row-partitioned index only when the target table is stored in a single RDAREA. Therefore, the partitioning storage conditions cannot be changed.

If the partitioning storage conditions cannot be changed due to the relationship between index storage RDAREAs and table storage RDAREAs, delete the applicable index first, execute `ALTER TABLE`, and then re-define the index.

Figures 13-18 and 13-19 show cases in which a table storage RDAREA cannot be split.

Figure 13-18: Case in which a table storage RDAREA cannot be split (part 1)

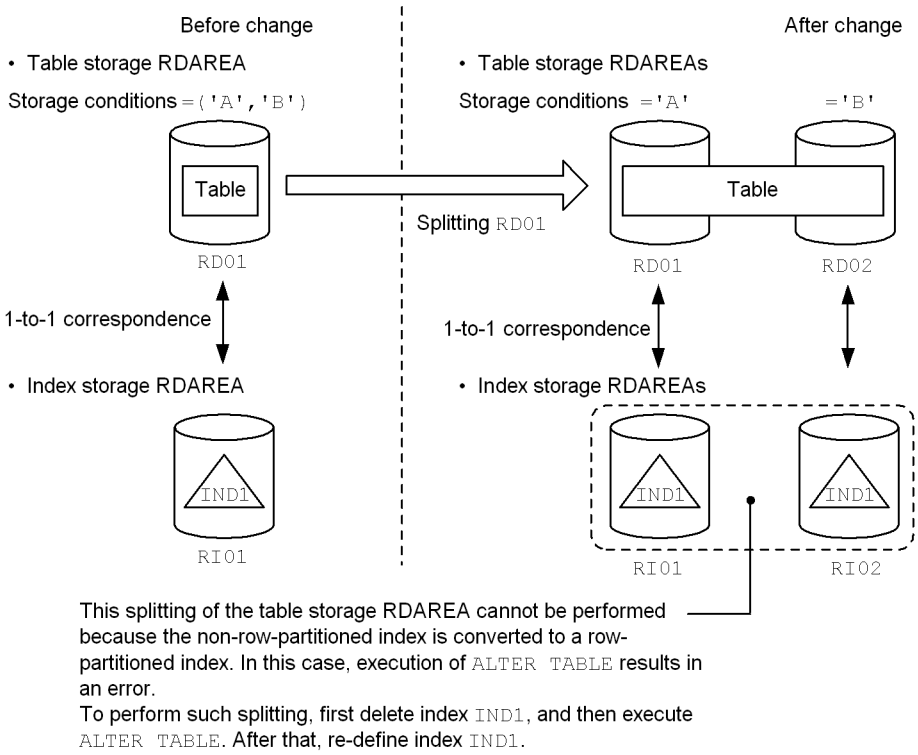


Figure 13-19: Case in which a table storage RDAREA cannot be split (part 2)

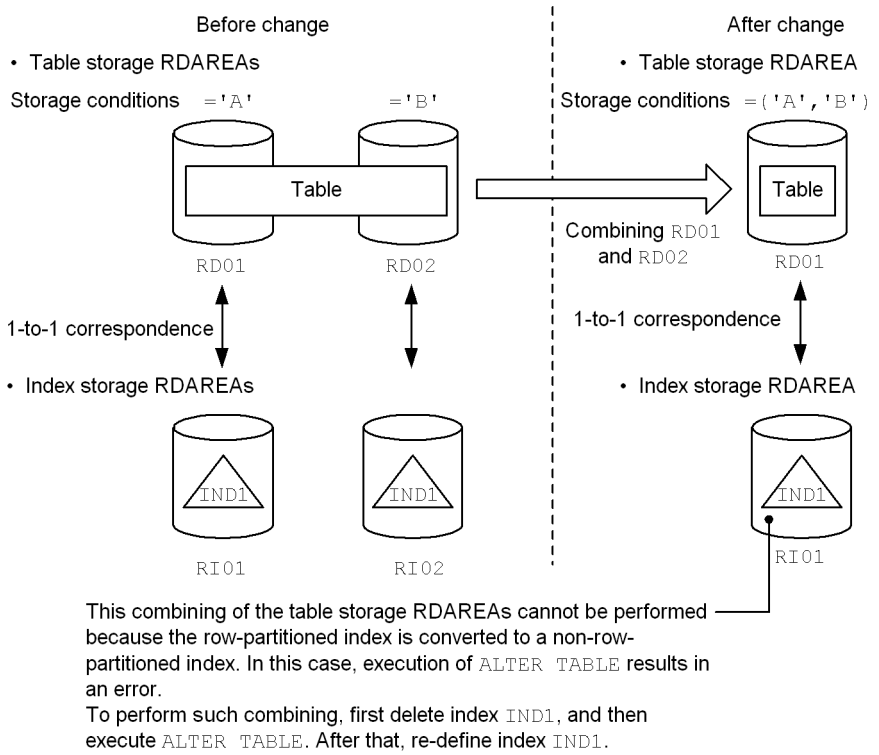


Table 13-6: Applicability of changing partitioning storage conditions for a non-partitioning key index

Table RDAREAs		Index RDAREAs	
Row partitioning between servers	**Row partitioning within servers**	**Row partitioning between servers**	**No row partitioning between servers**
Yes	Yes	Y	N
	No	—	Y#
No	Yes	Y	N
	No	—	—

Legend:

Y: Can be changed.

N: Cannot be changed.

—: Not applicable.

#: Cannot be changed if multiple partitioning storage conditions are stored in the same RDAREA.

(b) RDAREAs storing LOB and indexes

If RDAREAs storing LOB and indexes are defined for a table whose partitioning storage conditions are to be changed, these RDAREAs must also be partitioned or combined along with the table in the same manner.

(c) RDAREAs after change

RDAREAs after the change cannot be used for the following purposes:

- RDAREA that is not a user RDAREA cannot be used as a table or index RDAREA
- RDAREA that is not a user LOB RDAREA cannot be used as a BLOB column RDAREA
- Shared RDAREA
- RDAREA for storing a rebalancing table or resources related to a rebalancing table
- RDAREA of a replica generation when the inner replica facility is used (must be an RDAREA of the original generation)

13.12.4 How to change partitioning storage conditions (in the case of boundary value specification)

To change the partitioning storage conditions, specify `CHANGE RDAREA` in the `ALTER TABLE` statement. Splitting and combining cannot be executed at the same time. To execute both splitting and combining, use two separate `ALTER TABLE` statements.

(1) Partitioning storage conditions before combining

The following shows an example of partitioning storage conditions before combining or splitting:

```
CREATE FIX TABLE "T1"("C1" INT,"C2" CHAR(10)) PARTITIONED BY "C1"
 IN(("TA1")100,("TA2")200,("TA3")400,("TA4")500,("TA5")600,("TA6"))
CREATE INDEX "I1" ON "T1"("C1")
 IN(("IA1"),("IA2"),("IA3"),("IA4"),("IA5"),("IA6"))
```

Boundary values	100	200	400	500	600	
RDAREAs	TA1	TA2	TA3	TA4	TA5	TA6
	IA1	IA2	IA3	IA4	IA5	IA6

(2) Combining partitioning storage conditions

The following example combines boundary values 100 and 200 into 200:

```
ALTER TABLE "T1" CHANGE RDAREA
 ((100),(200)) INTO "TA2"
 FOR INDEX "I1" INTO "IA2"
```

Boundary values	200	400	500	600	
RDAREAs	TA2	TA3	TA4	TA5	TA6
	IA2	IA3	IA4	IA5	IA6

(3) Splitting a partitioning storage condition

The following example splits the storage range of 600 (& above) into 600 and 700 (& above):

```
ALTER TABLE "T1" CHANGE RDAREA
 ((MAX)) INTO (("TA6")700,("TA1"))
 FOR INDEX "I1" INTO (("IA6"),("IA1"))
```

Boundary values	200	400	500	600	700	
RDAREAs	TA2	TA3	TA4	TA5	TA6	TA1
	IA2	IA3	IA4	IA5	IA6	IA1

13.12.5 Splitting an RDAREA (in the case of boundary value specification)

The split facility splits into multiple RDAREAs the data stored in a specified range of a table that is partitioned by boundary values. The details of the split facility are explained below.

(1) Maximum values

Table 13-7 shows maximum values for the split facility.

Table 13-7: Maximum values for the split facility (in the case of boundary value specification)

Item	Maximum value	What happens when the maximum value is exceeded
Number of RDAREAs that can be split	1	Causes an error in `ALTER TABLE`.
Number of RDAREAs into which an RDAREA can be split in a single operation	16	
Total number of RDAREAs following splitting	1024	
Total number of specified boundary values following splitting (including `other`)	3000	

(2) Determining the RDAREA for splitting

The RDAREA that satisfies a specified storage condition is selected for splitting, based on a partitioning boundary value that is specified in `CHANGE RDAREA` of the `ALTER TABLE` definition SQL. Because specification of a boundary value identifies the target RDAREA, there is no need to specify the RDAREA itself. Table 13-8 shows the specification of `ALTER TABLE` and how the RDAREA to be split is determined.

Table 13-8: ALTER TABLE specification and determination of RDAREA to be split

Specification	Condition	Can be specified?	Determination of RDAREA to be split
Boundary value	The specified boundary value is specified in the table definition	Y	RDAREA that stores the specified boundary value range becomes the target for splitting.
	The specified boundary value is not specified in the table definition	N	—
`'MAX'`	—	Y	RDAREA that stores partitioning key values that are greater than the maximum boundary value becomes the target for splitting.

Legend:

Y: Can be specified.

N: Cannot be specified.

—: Not applicable.

(3) Determining the RDAREAs to be used after splitting

The RDAREAs to be used for storing data after splitting are determined on the basis

of the post-splitting boundary values and RDAREAs specified in CHANGE RDAREA of ALTER TABLE. Note that the post-splitting boundary values that are specified must be in ascending order, and the maximum value of the specified boundary values must be greater than the pre-splitting boundary value. Additionally, all partitioned boundary values must be within the range of the pre-splitting storage conditions. Table 13-9 shows the ALTER TABLE specification values and how the RDAREAs to be used after splitting are determined. Figure 13-20 shows the boundary value conditions before and after splitting.

Table 13-9: ALTER TABLE specification values and the method of determining the RDAREAs to be used after splitting

Pre-splitting boundary values	Post-splitting boundary values specification	Condition	Can be specified?	Determination of post-splitting RDAREAS
Minimum boundary value (1. in the figure)	Boundary values	Post-splitting boundary value is smaller than the minimum boundary value (4. in the figure)	Y	Specified storage conditions and RDAREAs are used to select the post-splitting RDAREAs.
		Post-splitting boundary value is equal to or greater than the minimum boundary value (1., 2., 3., and 5. in the figure)	N	Causes an error in ALTER TABLE.
	Omitted	None	Y	RDAREA specified to store the maximum value range among the post-splitting storage conditions
Intermediate or maximum boundary value (2. in the figure)	Boundary values	Post-splitting boundary value is equal to or smaller than the boundary value that precedes the pre-splitting boundary value (1. and 4. in the figure when the pre-splitting boundary value is 200)	N	Causes an error in ALTER TABLE.

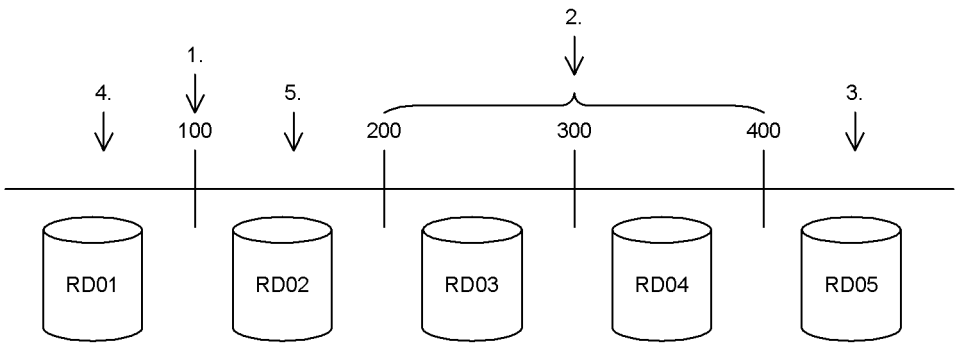
Pre-splitting boundary values	Post-splitting boundary values specification	Condition	Can be specified?	Determination of post-splitting RDAREAS
		Post-splitting boundary value is smaller than the pre-splitting boundary value and is greater than the preceding boundary value (5. in the figure when the pre-splitting boundary value is 200)	Y	Specified storage conditions and RDAREAs are used to select the post-splitting RDAREAs.
		Post-splitting boundary value is equal to or greater than the pre-splitting boundary value (200 when the pre-splitting boundary value is 200)	N	Causes an error in `ALTER TABLE`.
	Omitted	None	Y	RDAREA specified to store the maximum value range among the post-splitting storage conditions
`'MAX'` (3. in the figure)	Boundary values	Post-splitting boundary value is smaller than the maximum boundary value (1., 2., 4., and 5. in the figure)	N	Causes an error in `ALTER TABLE`.
		Post-splitting boundary value is greater than the maximum boundary value (3. in the figure)	Y	Specified storage conditions and RDAREAs are used to select the post-splitting RDAREAs.
	Omitted	None	Y	RDAREA specified to store the data that is greater than the maximum boundary value.

Legend:

Y: Can be specified.

N: Cannot be specified.

Figure 13-20: Boundary value conditions before and after splitting



The post-splitting RDAREAs may include (reuse) the pre-splitting RDAREA or may be all newly created. Furthermore, after splitting, multiple (non-contiguous) storage ranges can be stored in the same RDAREA. During this process, depending on the specifications, the system can automatically combine boundary values or cause an error in `ALTER TABLE`. The following describes the system actions:

1. Splits into RDAREAs as specified.

   If it is specified that contiguous storage ranges after splitting not be stored in the same RDAREA, the system splits the storage ranges as specified.

2. Automatically combines storage ranges.

   If the RDAREAs storing the storage ranges preceding and succeeding the storage range to be split are specified as the RDAREAs to be used after splitting, the system automatically combines contiguous boundary values in the following cases:

   - When the RDAREA storing the range preceding the storage range to be split is the same as the RDAREA for storing the beginning storage range after splitting
   - When the RDAREA storing the range succeeding the storage range to be split is the same as the RDAREA for storing the last storage range after splitting

3. Causes an error in `ALTER TABLE`.

   You cannot execute a split that stores multiple contiguous boundary values in the same RDAREA after the specified splitting. If such an attempt is made, the system causes an error in `ALTER TABLE`. In this case, you must modify and re-execute `ALTER TABLE` so that it combines the boundary value ranges and stores them in a single RDAREA.

Table 13-10 shows the system's action for storing multiple storage ranges in the same

RDAREA.

Table 13-10: System action for storing multiple storage ranges in the same RDAREA

Specification of RDAREAs after splitting			One of the specified RDAREAs that is being used is outside the splitting target storage range	Results of splitting/ combining	System action
If the focus is placed only on the specified post-splitting RDAREAs, multiple contiguous boundary values are not stored in the same RDAREA.	RDAREAs that are not used in the pre-splitting table are specified.	—	—	—	Splits into the RDAREAs as specified. Corresponds to Example 1-1 in *Figure 13-21*.
	RDAREAs that are used in the pre-splitting table are specified.	Being used only in the splitting target storage range.	—	—	Splits into the RDAREAs as specified. Corresponds to Example 1-2 in *Figure 13-21*.
		Being used outside the splitting target storage range.	Specified as the leading RDAREA after splitting.	Same as the RDAREA immediately preceding the splitting target storage range	Combines the storage range with the immediately preceding boundary value. Corresponds to Example 2-1 in *Figure 13-23*.
				Different from the RDAREA immediately preceding the splitting target storage range	Splits into the RDAREAs as specified. Corresponds to Example 1-3 in *Figure 13-21*.
			Specified in the middle of the splitting.	—	Splits into the RDAREAs as specified. Corresponds to Example 1-4 in *Figure 13-22*.

Specification of RDAREAs after splitting			One of the specified RDAREAs that is being used is outside the splitting target storage range	Results of splitting/ combining	System action
			Specified at the end of the splitting.	Same as the RDAREA immediately succeeding the splitting target storage range	Combines the storage range with the immediately succeeding boundary value. Corresponds to Example 2-2 in *Figure 13-23*.
				Different from the RDAREA immediately succeeding the splitting target storage range	Splits into the RDAREAs as specified. Corresponds to Example 1-5 in *Figure 13-22*.
If the focus is placed only on the specified post-splitting RDAREAs, multiple contiguous boundary values can be stored in the same RDAREA.	—	—	—	—	Causes an error in `ALTER TABLE`. Corresponds to Example 3-1 in *Figure 13-24*.

Legend:

—: Not applicable.

Figure 13-21: Example 1 of system action when storing multiple storage ranges in the same RDAREA (1 of 2)

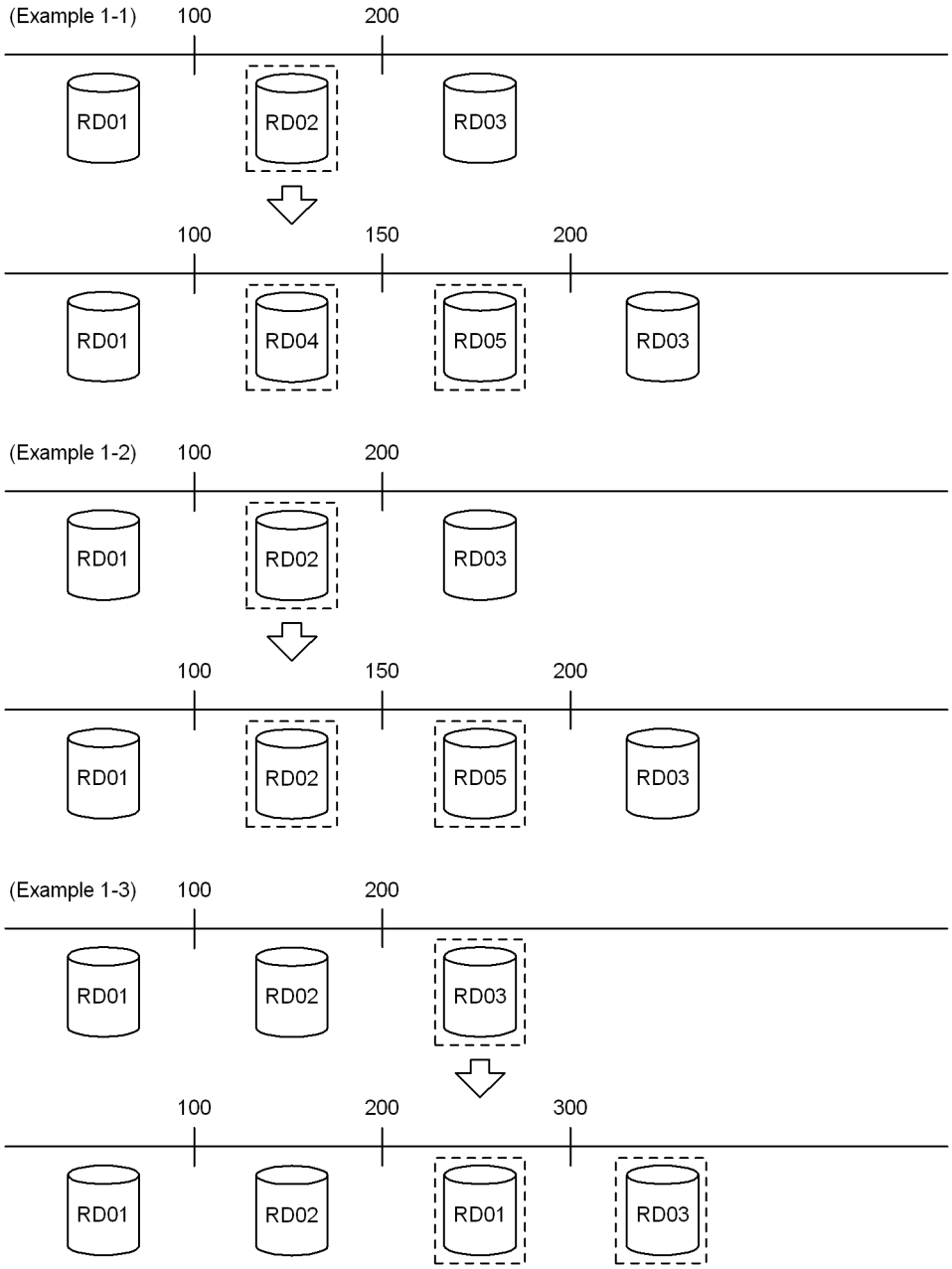


Figure 13-22: Example 1 of system action when storing multiple storage ranges in the same RDAREA (2 of 2)

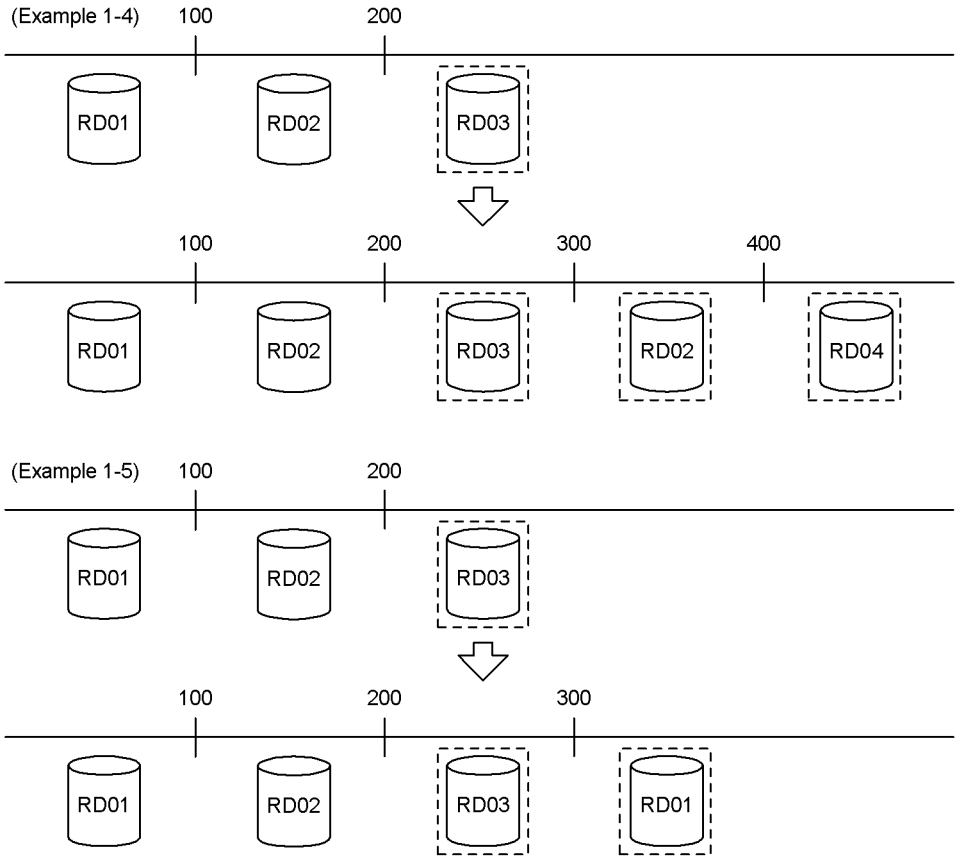


Figure 13-23: Example 2 of system action when storing multiple storage ranges in the same RDAREA

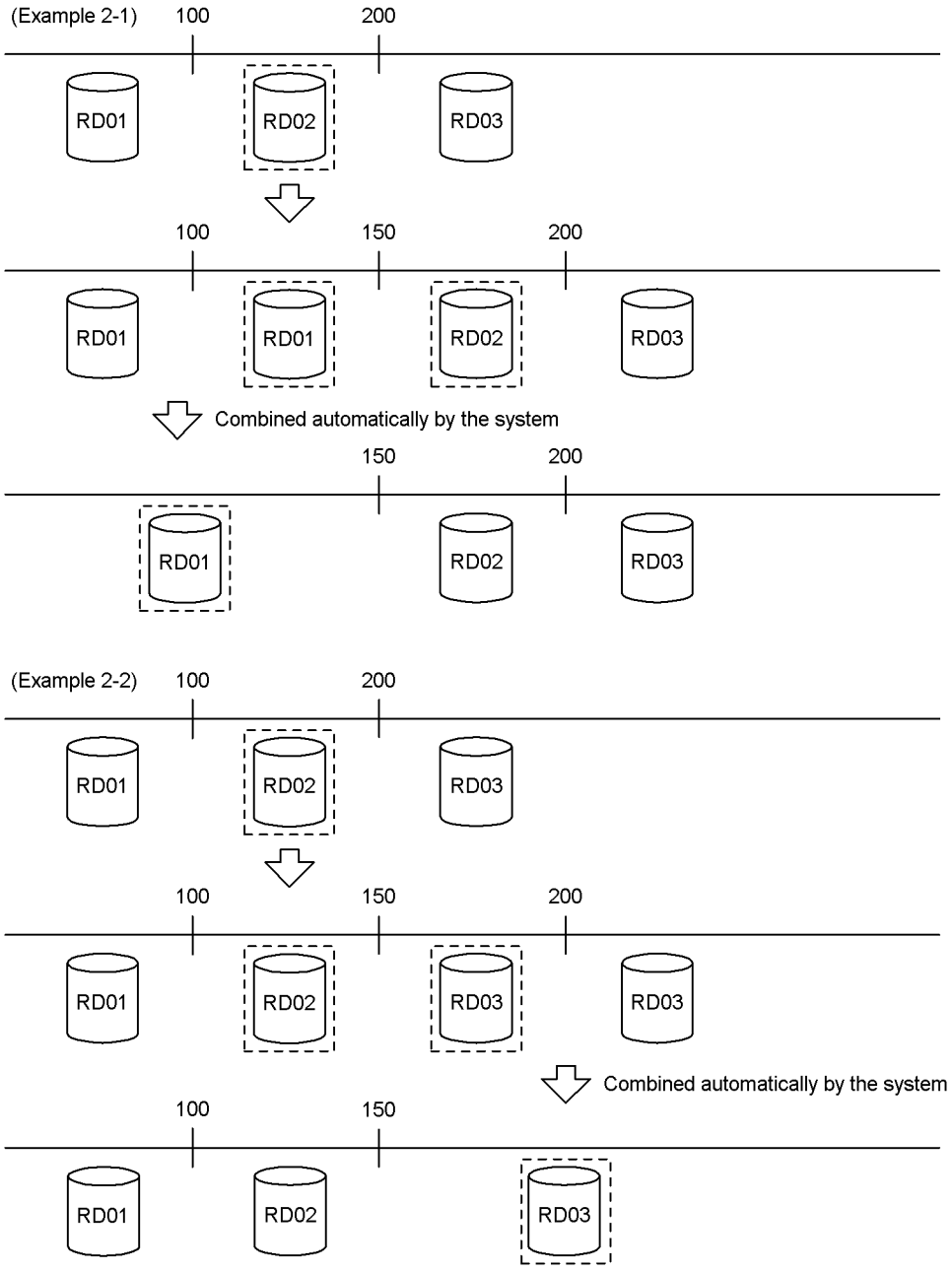
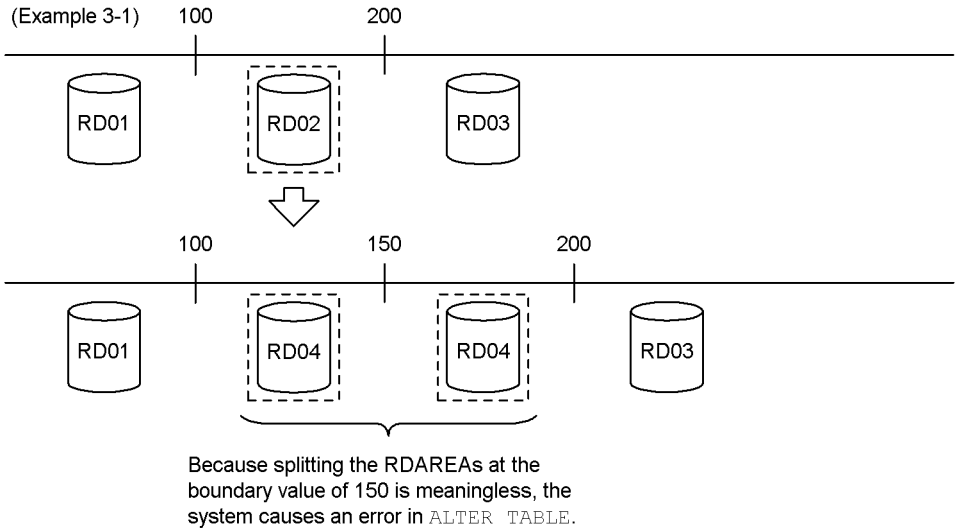


Figure 13-24: Example 3 of system action when storing multiple storage ranges in the same RDAREA



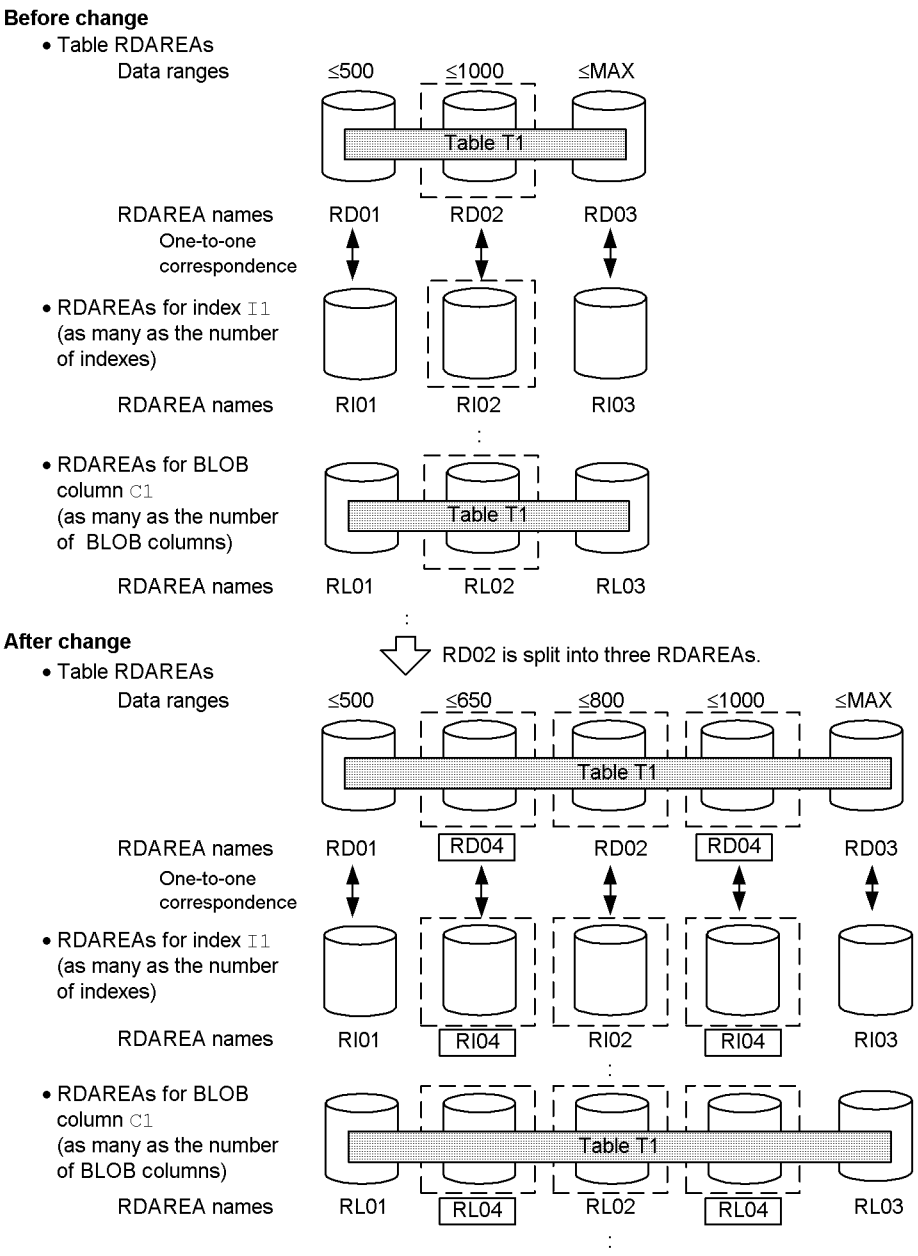
(4) Correspondence between a table and RDAREAs other than for the table

When, for example, a partitioning key index is defined for a table, the index data also must be stored in RDAREAs. If the partitioning storage conditions are changed for the table, the index must be split in the same way that the table RDAREAs are partitioned. Table 13-11 shows how to specify a table and RDAREAs other than for the table (combining storage conditions when partitioning boundary values). If more than one unit of a resource such as is shown in Table 13-11 is defined, the specification method must be applied to each of them. If any specification is incorrect, the system causes an error in `ALTER TABLE`. Figure 13-25 shows an example of the correspondence between a table and RDAREAs other than for the table. In this example, the table's contents and the contents of the RDAREAs other than for the table are split into three.

Table 13-11: Specifying a table and RDAREAs other than for the table (combining storage conditions when partitioning boundary values)

Resource name		Specification method
Column	BLOB column	Specify these resources so that they correspond to the table RDAREAs on a one-to-one basis. If a table RDAREA is specified more than once, specify the resources more than once also so that they correspond to the specified tables. If the existing table RDAREA is to be used after the change, you must specify the RDAREAs for storing the existing indexes and LOB data so that they correspond to the same boundary values.
Index	Cluster key index Primary key index (including an index for which the primary key and a cluster key are defined) B-tree index	

Figure 13-25: Example of the correspondence between a table and RDAREAs other than for the table



(5) Handling the data in an RDAREA to be split

The general rule is that when a storage range is split based on a boundary value, the system automatically deletes the applicable table's existing table data from the RDAREA. However, under some conditions, you can specify that the data is to be retained.

1. Deleting data

   When a storage range is split based on a boundary value, some of the data in the RDAREA to be split will likely become data that will not be stored in that RDAREA after splitting. Therefore, all the data in the RDAREA in the storage range that is to be split must be deleted. Note that only the data in the table whose partitioning storage conditions are to be changed is deleted; data of other tables contained in that RDAREA is not deleted. One of the following methods is used to delete the data from the RDAREA:

   - Deleting all definition information

     If the pre-splitting RDAREA is not used for the table after splitting has taken place, all information about the pre-splitting RDAREA is deleted from the dictionary (`MASTER.SQL_DIV_TABLE`) containing the RDAREA information used by the table for each of the storage conditions. Table information that is managed within the RDAREA is also deleted. As a result, all data from the splitting-target table that existed in the RDAREA is deleted. In concept, this is equivalent to executing `DROP TABLE` for the RDAREA.

   - Deleting data only

     If the pre-splitting RDAREA will still be used for the table after splitting, the dictionary information and information managed within the RDAREA is not deleted, and only the table's data in the RDAREA is deleted. If an RDAREA that is to be used after splitting is already being used for another storage range, the data in this other storage range is also deleted. In concept, this is equivalent to executing `PURGE TABLE` for that RDAREA.

   Note that when data in an RDAREA is deleted, all data in the following corresponding RDAREAs is also deleted:

   - Index keys in an index RDAREA
   - Data in a BLOB column RDAREA

   Additionally, if the inner replica facility is being used, all generation data is deleted.

2. Saving data

   As explained in *1. Deleting data*, when a storage range is split based on a boundary value, the general rule is that the table's data in the pre-splitting RDAREA is deleted. However, the data in the RDAREA can be used as is if all

the conditions listed below are satisfied, and therefore it is possible to avoid deletion of such data:

- The pre-splitting RDAREA is to be used without modification as a post-splitting RDAREA.
- All the existing data is at or below the boundary value for the pre-splitting RDAREA.
- All data in the pre-splitting RDAREA will be within the storage range for the same RDAREA after splitting.

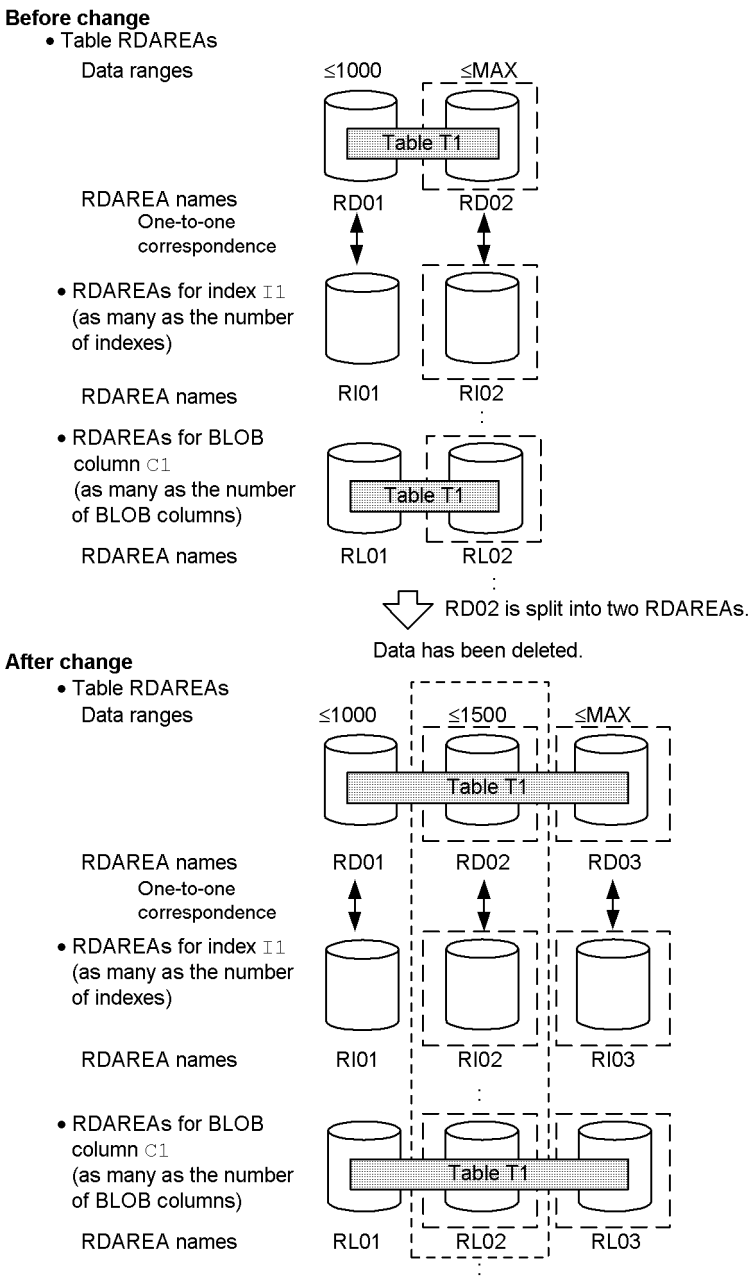
To instruct that the data in the RDAREA is not to be deleted, you must specify the `WITHOUT PURGE` clause in `ALTER TABLE`. Table 13-12 shows whether or not the data is deleted depending on the specification of the `WITHOUT PURGE` clause. Note that the system does not check whether or not all the data in the RDAREA matches the storage ranges after splitting.

Table 13-12: WITHOUT PURGE clause specification and data handling

Use of the RDAREA after splitting	Can the WITHOUT PURGE clause be specified?	Action when the WITHOUT PURGE clause is specified	Action when the WITHOUT PURGE clause is not specified
Pre-splitting RDAREA will be used as a post-splitting RDAREA.	Yes	Does not delete the data from the pre-splitting RDAREA.	Deletes the data from the pre-splitting RDAREA.
Pre-splitting RDAREA will not be used as a post-splitting RDAREA.	No	Not applicable	

Figure 13-26 shows an example of not specifying the `WITHOUT PURGE` clause. In this example, the table's data is deleted from the pre-splitting RDAREA because the `WITHOUT PURGE` clause is not specified.

Figure 13-26: Example of an RDAREA from which data is deleted because WITHOUT PURGE is not specified



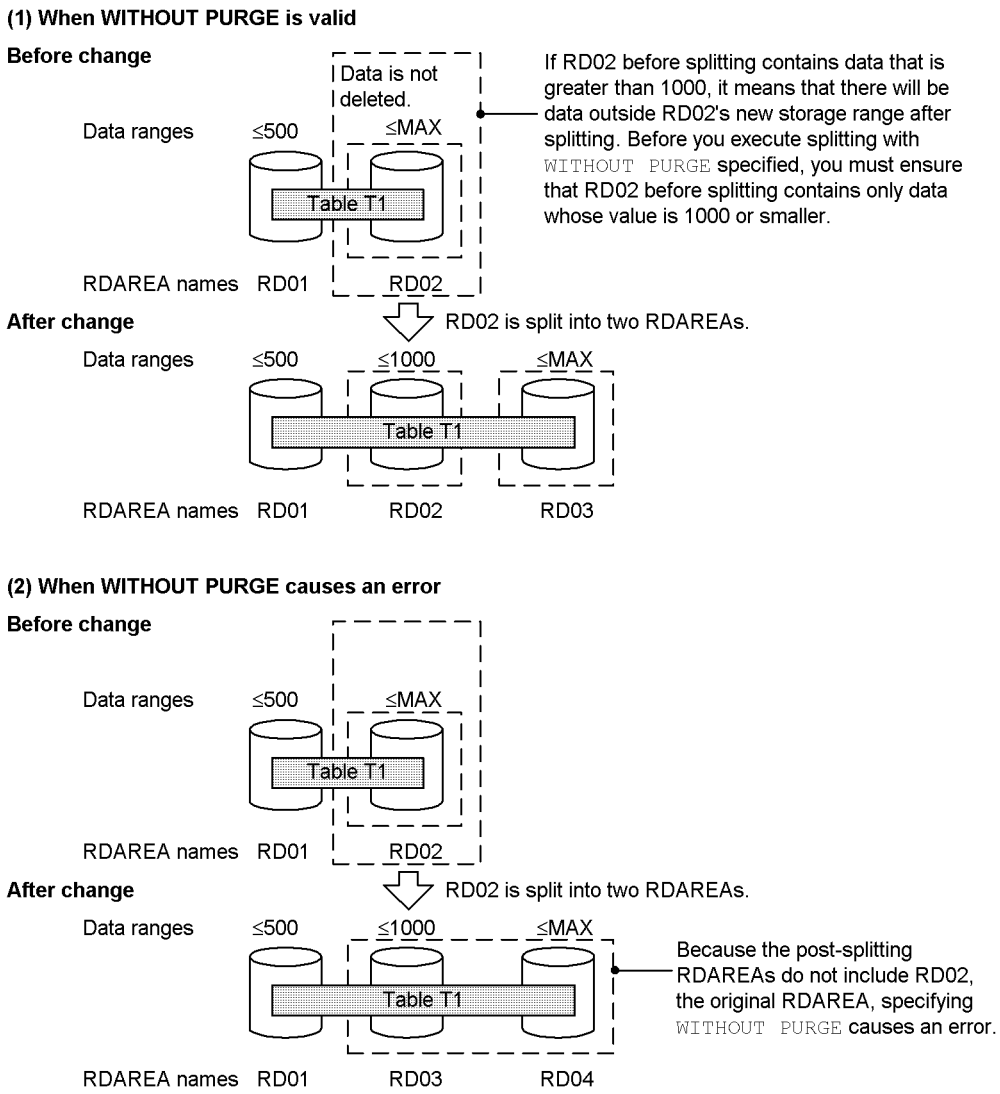
3. Notes on cases in which data is not deleted

   There may be data in the pre-splitting RDAREA that will not be within the applicable storage range after splitting (this type of data validity is not checked during execution of `ALTER TABLE`). Consequently, as a result of a change in partitioning storage conditions, data outside the new storage range may exist in the post-splitting RDAREA. When this happens, HiRDB may not function correctly during SQL execution.

   For this reason, you must be careful about using the `WITHOUT PURGE` clause when you change partitioning storage conditions. If you cannot guarantee that all data in the pre-splitting RDAREA will be within the new storage range of that RDAREA after splitting, you should unload the data from the pre-splitting RDAREA, execute the change in partitioning storage conditions without specifying the `WITHOUT PURGE` clause, and then load the unloaded data into the split RDAREAs. For details about how to recover from a mistake in a splitting operation, see *13.13.9(2) Recovery procedure when data not satisfying the post-splitting storage condition remains*.

   Figure 13-27 shows examples of handling data in the post-splitting RDAREAs when `WITHOUT PURGE` is specified. These examples illustrate valid and invalid specifications of `WITHOUT PURGE`.

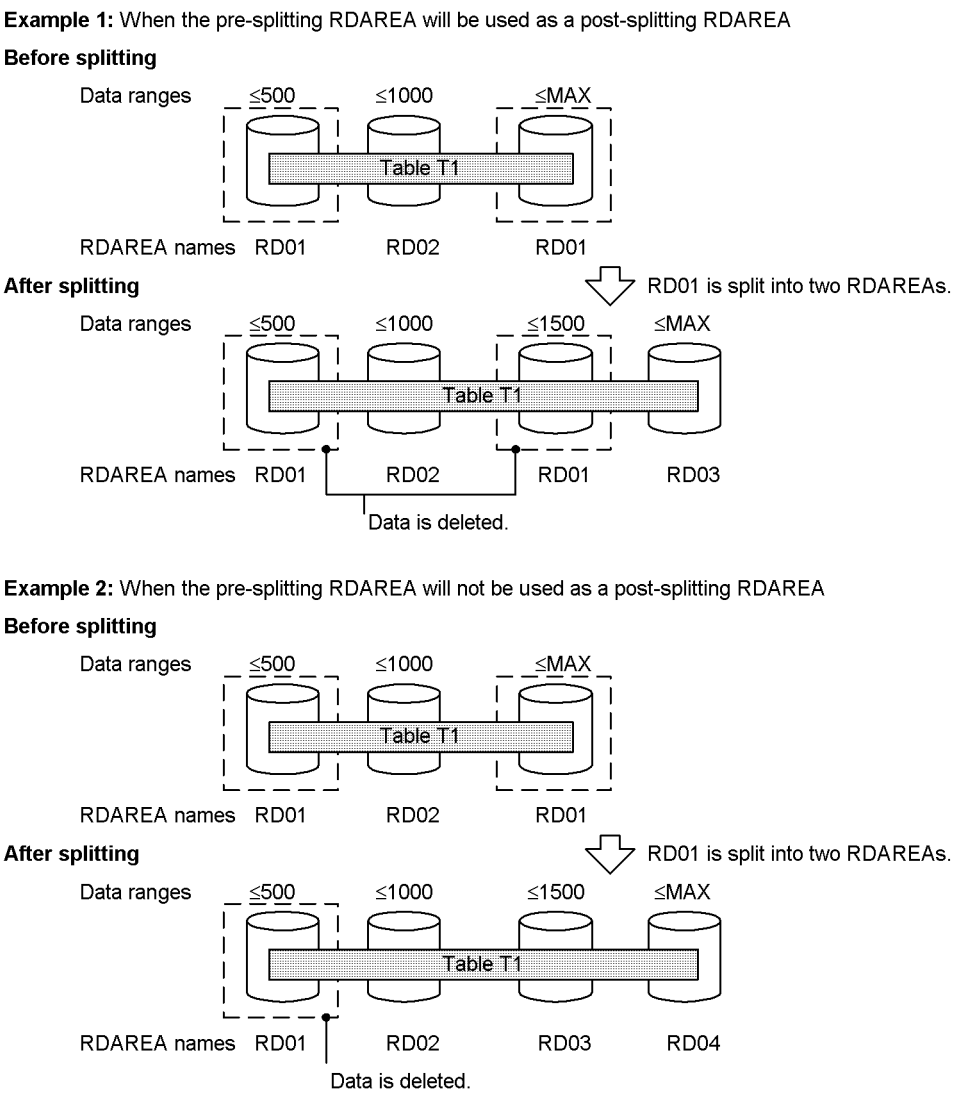
Figure 13-27: Examples of handling data in the post-splitting RDAREAs (valid and invalid specifications of WITHOUT PURGE)



4. Notes on deleting data

   If the pre-splitting RDAREA contains data that will be stored in another storage range, data in this other storage range will also be deleted. This holds true, regardless of whether or not the pre-splitting RDAREA will be used as a post-splitting RDAREA. Figure 13-28 shows examples of RDAREA data deletion.

Figure 13-28: Examples of RDAREA data deletion



13.12.6 Combining RDAREAs (in the case of boundary value specification)

(1) Maximum and minimum values

Table 13-13 shows the maximum and minimum values for the combine facility.

Table 13-13: Maximum and minimum values for the combine facility

Item	Maximum and minimum values	What happens when the maximum or minimum value is exceeded?
Number of RDAREAs that can be combined	2-16 (minimum is 2, maximum is 16)	Causes an error in `ALTER TABLE`.
Number of RDAREAs following a single combining operation	1 (fixed)	
Total number of RDAREAs in the combined table	2 (minimum)	

(2) Determining the RDAREAs for combining

All RDAREAs that satisfy specified storage conditions are selected for combining, based on multiple boundary values specified in `CHANGE RDAREA` of `ALTER TABLE`. Because specification of boundary values identifies the target RDAREAs, there is no need to specify the RDAREAs themselves. The multiple boundary values must be specified in ascending order; they must also be specified to account for all the contiguous storage conditions defined for the table. For example, if boundary values 10, 20, 30, and 40 are defined in a table definition, specifying `10`, `30`, and `40` (and skipping specification of `20`) will cause an error in `ALTER TABLE`.

Table 13-14 shows the specification of `ALTER TABLE` and how the RDAREAs to be combined are determined.

Table 13-14: ALTER TABLE specification and determination of RDAREAs to be combined

Specification	Condition 1	Condition 2	Action
Boundary values	Boundary values are specified in the table definition.	Boundary values are specified in the order in which they are defined.	RDAREAs that satisfy the specified storage conditions become the RDAREAs that will be combined.
		Boundary values are not specified in their definition order.	Causes an error in `ALTER TABLE`.
	No boundary values are specified in the table definition.	None	Causes an error in `ALTER TABLE`.
`'MAX'`	Boundary value specified immediately before is the maximum boundary value in the table definition.	None	RDAREA storing data with partitioning key values greater than the maximum boundary value will be combined.

Specification	Condition 1	Condition 2	Action
	Boundary value specified immediately before is not the maximum boundary value in the table definition.	None	Causes an error in `ALTER TABLE`.

(3) Determining the RDAREA to be used after combining

The post-combination RDAREA specified in `CHANGE RDAREA` of `ALTER TABLE` becomes the RDAREA that stores all the specified pre-combination storage conditions. Because the boundary value after the combining operation is a merger of the boundary values that were combined, there is no need to specify it. For example, if the boundary values are 10, 20, 30, and 40, and a combining operation combines 20 and 30, the new boundary values become 10, 30, and 40. In this case, all data that is greater than 10 but equal to or smaller than 30 satisfies the storage condition for the combined RDAREA.

The post-combination RDAREA may be one of the pre-combination RDAREAs or may be a different RDAREA. Table 13-15 shows the RDAREAs that can be specified as the post-combination RDAREA.

Table 13-15: RDAREAs that can be specified as the post-combination RDAREA

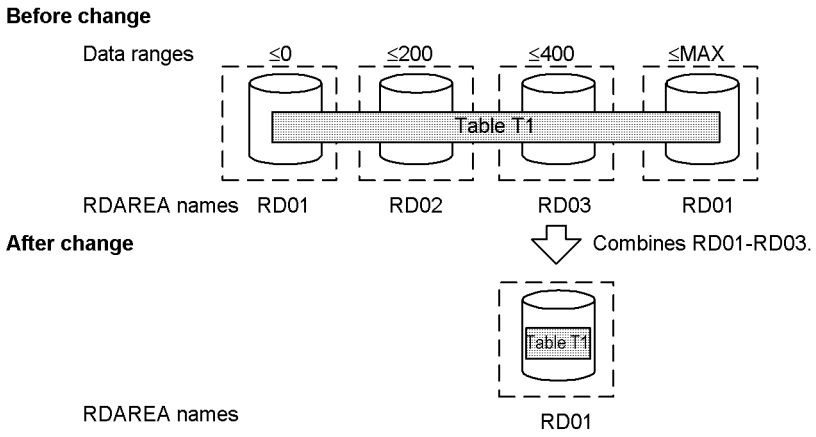
RDAREA		Can be used as the post-combination RDAREA?
Any one of the pre-combination RDAREAs	—	Yes
None of the pre-combination RDAREAs	RDAREA that is already being used for a boundary value of the same table (but not a boundary value that is being combined).	Yes
	RDAREA that is not being used for this table (a newly created RDAREA).	Yes

Legend:

—: Not applicable

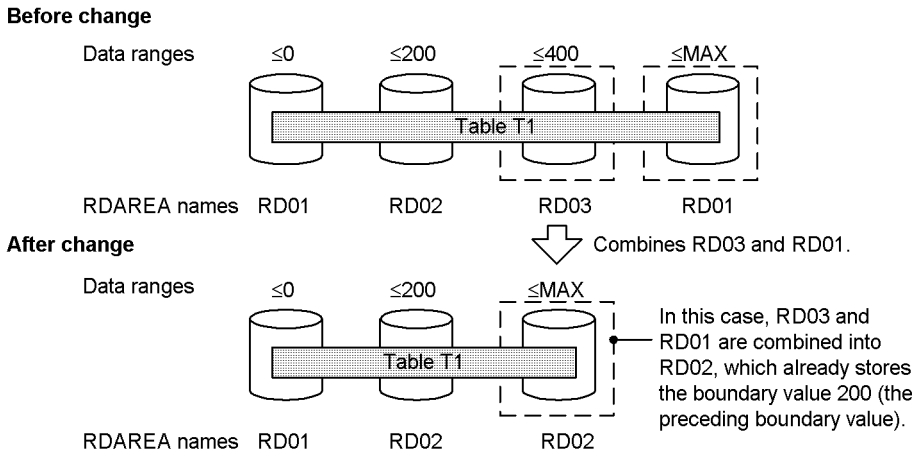
You can combine RDAREAs so that multiple storage conditions are stored in one RDAREA after combining. However, you cannot combine RDAREAs such that all data is stored in a single RDAREA after combining. Figure 13-29 shows an example of a combining operation that is not allowed (which combines all data into a single RDAREA).

Figure 13-29: Example of a combining operation that is not allowed (which combines all data into a single RDAREA)



You cannot combine RDAREAs such that the post-combination RDAREA also stores the preceding or succeeding storage range. In this case, you must combine the RDAREAs by also including the preceding or succeeding RDAREA. Figure 13-30 shows an example of a combining operation that is not allowed (where the post-combination RDAREA is the same RDAREA that stores the preceding storage range).

Figure 13-30: Example of a combining operation that is not allowed (the post-combination RDAREA also stores the preceding storage range)



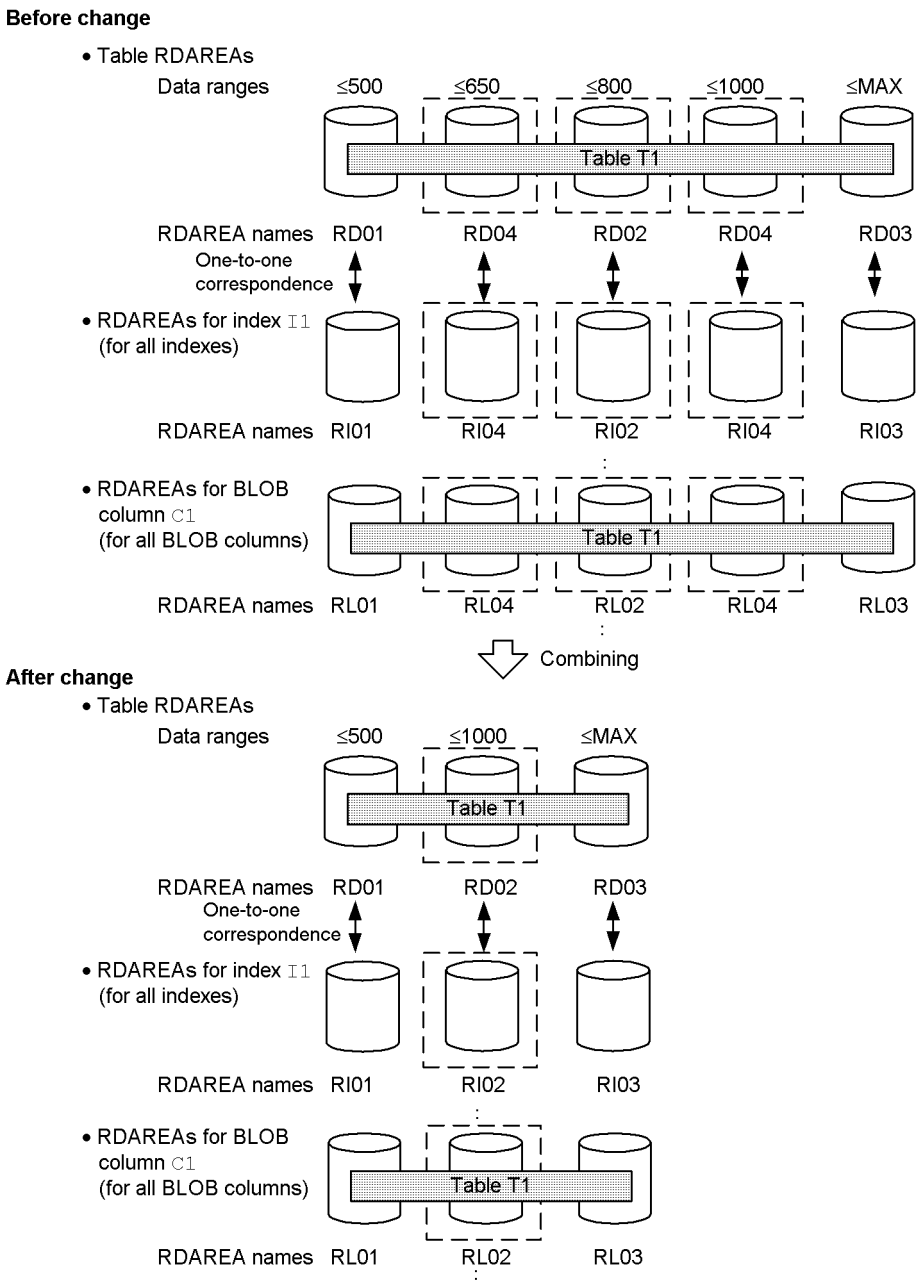
(4) Correspondence between a table and RDAREAs other than for the table

When, for example, a partitioning key index is defined for a table, the index data also must be stored in RDAREAs. If the partitioning storage conditions are changed for the table, the index must be partitioned in the same way that the table RDAREAs are partitioned. Table 13-16 shows how to specify a table and RDAREAs other than for the table (combining storage conditions when partitioning boundary values). If more than one unit of a resource such as is shown in Table 13-16 is defined, the specification method must be applied to each of them. If any specification is incorrect, the system causes an error in `ALTER TABLE`. Figure 13-31 shows an example of the correspondence between a table and RDAREAs other than for the table.

Table 13-16: Specifying a table and RDAREAs other than for the table (combining storage conditions when partitioning boundary values)

Resource name		Specification method
Column	BLOB column	Specify these resources so that they correspond to the table RDAREAs on a one-to-one basis. If duplicate table RDAREAs are specified, specify duplicate resources so that they correspond to the table. If an existing table RDAREA is to be used after the change, you must specify RDAREAs for storing the existing indexes and LOB data so that they correspond to the same boundary values.
Index	Cluster key index Primary key index (including an index for which the primary key and a cluster key are defined) B-tree index	

Figure 13-31: Example of the correspondence between a table and RDAREAs other than for the table



(5) Handling the data in the RDAREAs to be combined

The general rule is that when storage ranges are combined based on boundary values, the system automatically deletes the applicable table's existing table data from the RDAREAs. However, under some conditions, you can specify that data is to be retained.

1. Deleting data

   When storage ranges are combined based on boundary values, pre-combination RDAREAs may no longer be used by the table after combining. Therefore, the system automatically deletes the data in those RDAREAs. Note that only the data in the table whose partitioning storage conditions are to be changed is deleted; data of other tables contained in the RDAREA is not deleted. One of the following methods is used to delete the data from the RDAREA:

   - Deleting all definition information

     If a pre-combination RDAREA will not be used for the table after combining, all information about the pre-combination RDAREA is deleted from the RDAREA information in the dictionary table (`SQL_DIV_TABLE`) used for the table's applicable storage condition. Table information that is managed within the RDAREA is also deleted. As a result, all data from the combination-target table that existed in the RDAREA is deleted. In concept, this is equivalent to executing `DROP TABLE` for the RDAREA.

   - Deleting data only

     If a pre-combination RDAREA is to be used as the RDAREA for the combined storage ranges, the dictionary information and information managed within the RDAREA is not deleted, and only the table's data in the RDAREA is deleted. If the RDAREA to be used for the combined storage ranges is already being used for another storage range, the data in this other storage range is also deleted. In concept, this is equivalent to executing `PURGE TABLE` for that RDAREA.

   Note that when data in an RDAREA is deleted, all data in the following corresponding RDAREAs is also deleted:

   - Index keys in an index RDAREA
   - Data in a BLOB column RDAREA

   Additionally, if the inner replica facility is being used, all generation data is deleted.

2. Saving data

   As explained in *1. Deleting data*, when storage ranges are combined based on boundary values, the general rule is that the table's data in the pre-combination RDAREAs is deleted. However, the data in an RDAREA can be used as is if the

condition listed below is satisfied, and therefore it is possible to avoid deletion of such data:

- The pre-combination RDAREA is to be used without modification as the post-combination RDAREA.

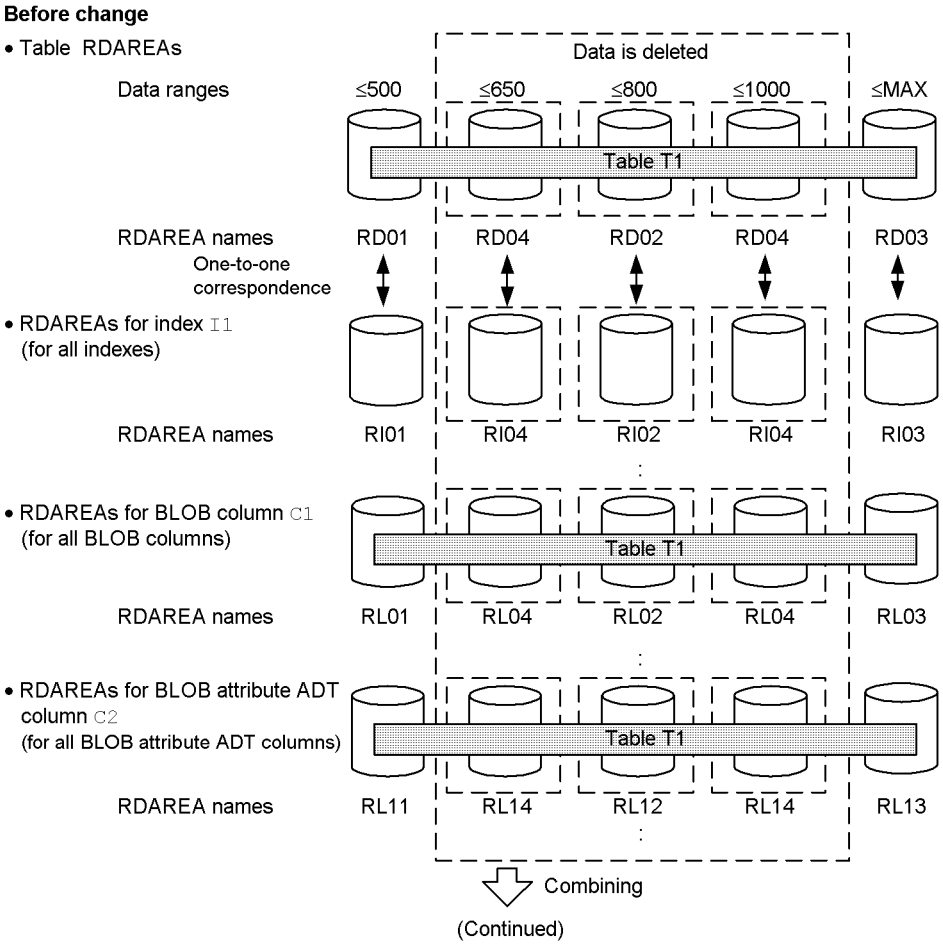
To instruct that the data in the RDAREA is not to be deleted, you must specify the `WITHOUT PURGE` clause in `ALTER TABLE`. Table 13-17 shows whether or not the date is deleted depending on the specification of the `WITHOUT PURGE` clause.

Table 13-17: WITHOUT PURGE clause specification and data handling

Use of the RDAREA after combining	Can the WITHOUT PURGE clause be specified?	Action when the WITHOUT PURGE clause is specified.	Action when the WITHOUT PURGE clause is not specified.
One of the pre-combination RDAREAs will be used as the post-combination RDAREA	Yes	Does not delete the data from the RDAREA to be re-used after combining; deletes data from the other pre-combination RDAREAs.	Deletes the data from all pre-combination RDAREAs.
No pre-combination RDAREA will be used as the post-combination RDAREA (a different RDAREA from any of the pre-combination RDAREAs is to be used as the post-combination RDAREA)	No	Causes an error in `ALTER TABLE`.	

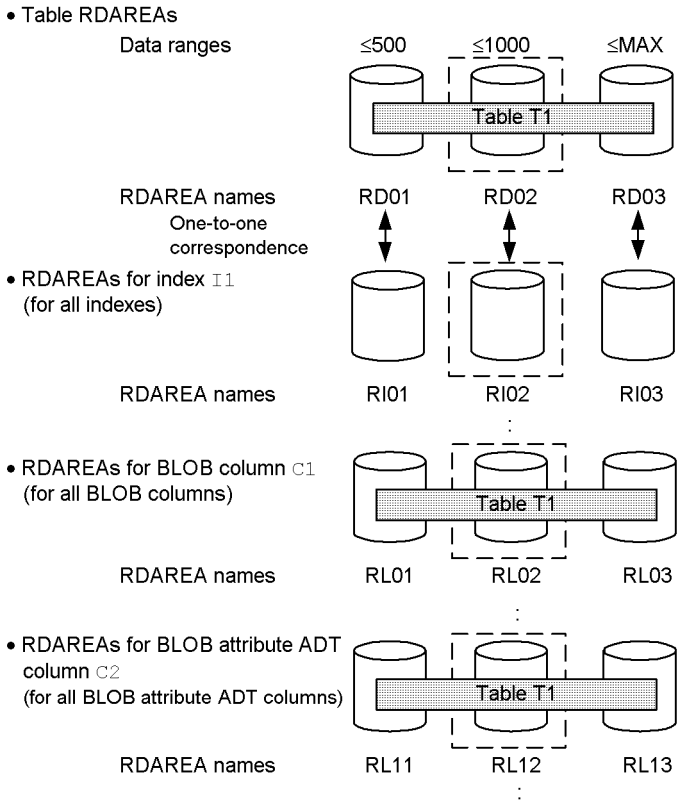
Figure 13-32 shows the RDAREAs from which data is deleted during combining.

Figure 13-32: RDAREAs from which data is deleted during combining



(Continued)

After change



3. Notes on cases in which data is not deleted

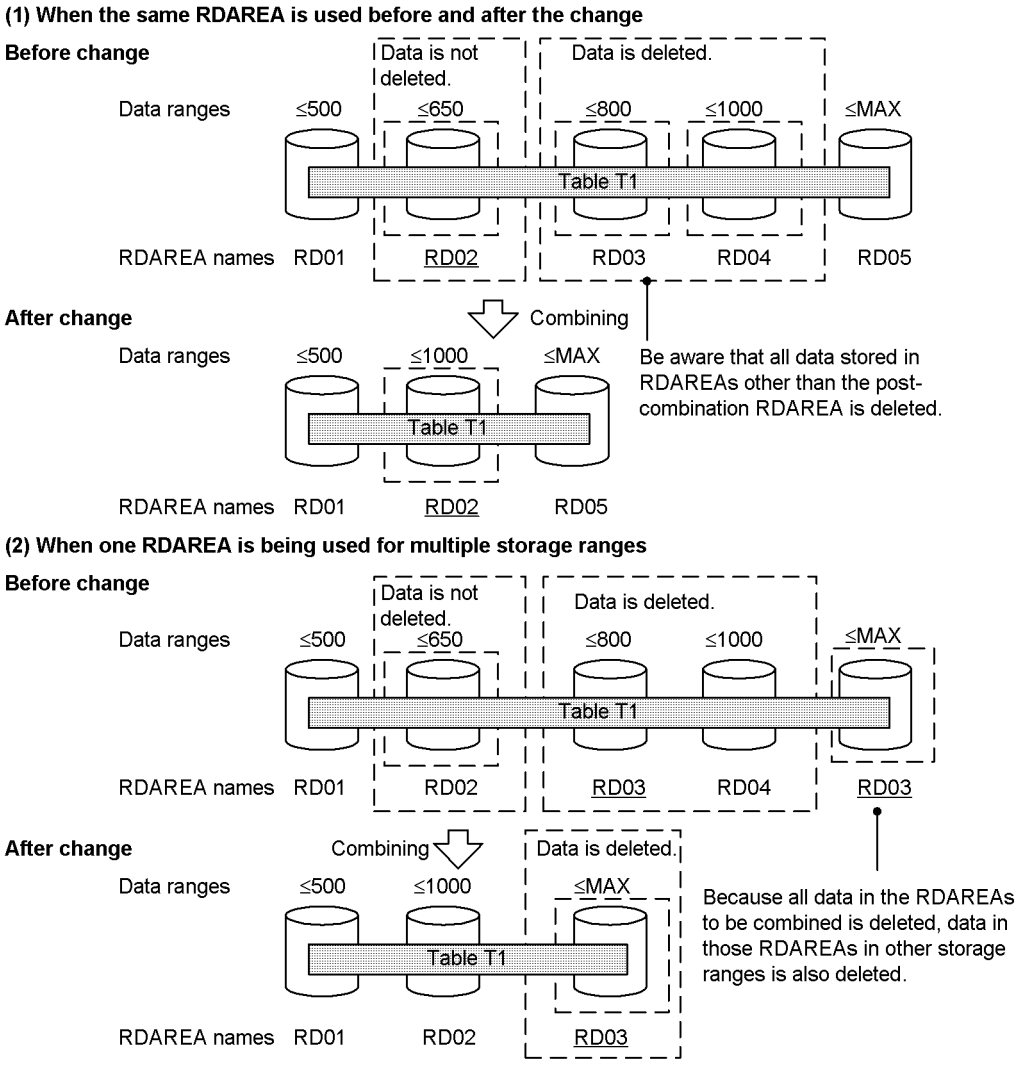
   If you specify `WITHOUT PURGE` to save the data, then you unload all the data before combining and reload it into the post-combination RDAREA, the data saved by the `WITHOUT PURGE` specification will have been registered twice. For this reason, you must not execute unloading and post-combination reloading of the data of an RDAREA if that RDAREA's data is being saved by specification of `WITHOUT PURGE`.

   Even if you specify `WITHOUT PURGE`, the data in any pre-combination RDAREA that is not used as the post-combination RDAREA is deleted. Also, if an RDAREA containing a storage range to be combined is not to be used as the post-combination RDAREA and that pre-combination RDAREA also contains a storage range that is not included in the combining operation, the data in this other storage range will also be deleted. Therefore, to combine one or more selected

storage ranges in an RDAREA that stores multiple storage ranges, it is necessary to change the partitioning storage conditions without specifying `WITHOUT PURGE`, and then reload the combined RDAREA with the data that has been unloaded.

Figure 13-33 shows examples of handling data in the RDAREAs to be combined when `WITHOUT PURGE` is specified.

Figure 13-33: Examples of handling data in the RDAREAs to be combined (when WITHOUT PURGE is specified)



4. Notes on deleting data

   As explained in the notes on not deleting data, if there is an RDAREA that is being used for a storage range that is not to be included in the combining operation, as well as for a storage range that will be included in the combining operation, and this RDAREA will not become the post-combination RDAREA, the data in the other storage range is also deleted. For this reason, Hitachi recommends in this case that you unload all data from the RDAREAs to be combined, change the partitioning storage conditions without specifying `WITHOUT PURGE`, and then reload the combined RDAREA with the data that has been unloaded.

13.12.7 How to change partitioning storage conditions (in the case of storage condition specification)

To change the partitioning storage conditions, specify `CHANGE RDAREA` in the `ALTER TABLE` statement. Splitting and combining cannot be executed at the same time. To execute both splitting and combining, use two separate `ALTER TABLE` statements.

(1) Partitioning storage conditions before combining

The following shows an example of partitioning storage conditions before combining or splitting:

```
CREATE FIX TABLE "T1"("C1" CHAR(10),"C2"...
 IN(("TA1")"C1"='Berkeley branch',("TA2")"C1"='Oakland branch',("TA3")"C1"='Sausalito
branch',
    ("TA4")"C1"=('San Francisco branch','San Jose branch'),("TA5"))
CREATE INDEX "I1" ON "T1"("C1")
 IN(("IA1"),("IA2"),("IA3"),("IA4"),("IA5"))
```

Key condition	C1='Berkeley branch'	C1='Oakland branch'	C1='Sausalito branch'	C1='San Francisco branch' or C1='San Jose branch'	Other
RDAREA	TA1	TA2	TA3	TA4	TA5
	IA1	IA2	IA3	IA4	IA5

(2) Combining partitioning storage conditions

The following example combines data from the Berkeley and Oakland branches into the same RDAREA:

```
ALTER TABLE "T1" CHANGE RDAREA PARTITIONED CONDITION (("TA1"),("TA2")) INTO ("TA1")
 FOR INDEX "I1" INTO "IA1"
```

Key condition	C1='Berkeley branch' or C1='Oakland branch'	C1= 'Sausalito branch'	C1='San Francisco branch' or C1='San Jose branch'	Other
RDAREA	TA1	TA3	TA4	TA5
	IA1	IA3	IA4	IA5

(3) Splitting a partitioning storage condition

The following example splits the data for the San Francisco and San Jose branches into separate RDAREAs:

```
ALTER TABLE "T1" CHANGE RDAREA PARTITIONED CONDITION
 (("TA4")) INTO (("TA4")"C1"=('San Francisco branch'),("TA6")"C1"=('San Jose branch'))
 FOR INDEX "I1" INTO (("IA4"),("IA6"))
```

Key condition	C1='Berkeley branch' or C1='Oakland branch'	C1= 'Sausalito branch'	C1= 'San Francisco branch'	C1= 'San Jose branch'	Other
RDAREA	TA1	TA3	TA4	TA6	TA5
	IA1	IA3	IA4	IA6	IA5

13.12.8 Splitting an RDAREA (in the case of storage condition specification)

(1) Rules for changing storage conditions

This subsection describes the rules for changing storage conditions.

(a) Maximum value

When you change storage conditions, you must observe the maximum values shown in Table 13-18.

Table 13-18: Maximum values for the split facility (in the case of storage condition specification)

Item	Maximum value	What happens when the maximum value is exceeded
Number of RDAREAs that can be split	1	Causes an error in `ALTER TABLE`.
Number of RDAREAs into which an RDAREA can be split in a single operation	16	

Item	Maximum value	What happens when the maximum value is exceeded
Total number of RDAREAs following splitting	1,024	
Total number of storage conditions after splitting (including RDAREAs with no storage condition specified)	15,000	

(b) Rules for splitting

The RDAREA to be split must satisfy specific conditions. Table 13-19 describes whether or not an RDAREA can be split.

Table 13-19: Determination of whether or not an RDAREA can be split

RDAREA to be split#1	Storage condition defined for table	Whether or not splittable	Description
RDAREA for which storage condition is specified	Only one storage condition is specified for the RDAREA.	N	Causes in an error in `ALTER TABLE`.
	Multiple storage conditions are specified for the RDAREA.	Y	An example of splitting an RDAREA with storage conditions specified is shown in Figure *13-34 Example of splitting an RDAREA with storage conditions specified*.
RDAREA for which no storage condition is specified	--	Y	An example of splitting an RDAREA with no storage condition specified is shown in Figure *13-35 Example of splitting an RDAREA with no storage condition specified*.
RDAREA with `OTHERS` specified#2	Storage conditions are specified for all RDAREAs.	Y	An example of splitting an RDAREA with `OTHERS` specified is shown in Figure *13-36 Example of splitting an RDAREA with OTHERS specified*.

RDAREA to be split[#1]	Storage condition defined for table	Whether or not splittable	Description
	There is an RDAREA with no storage condition specified.	N	An example where an RDAREA with OTHERS specified cannot be split is shown in Figure *13-37 Example where an RDAREA with OTHERS specified cannot be split*.

Legend:

Y: Can be split.

N: Cannot be split.

--: Not applicable.

#1

The RDAREA to be split is the one specified in the pre-change RDAREA information list for CHANGE RDAREA in the ALTER TABLE statement.

#2

For details about an RDAREA with OTHERS specified, see *13.12.8(2) RDAREA with OTHERS specified*.

Figure 13-34: Example of splitting an RDAREA with storage conditions specified

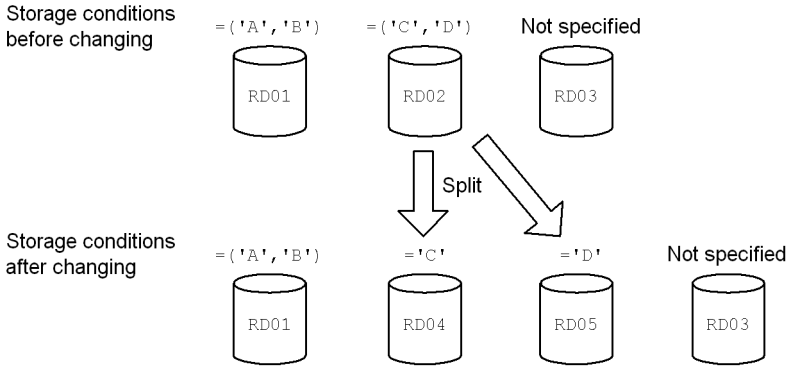


Figure 13-35: Example of splitting an RDAREA with no storage condition specified

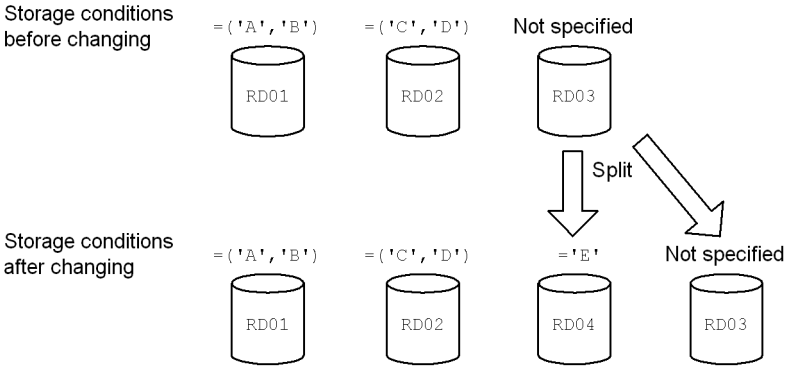


Figure 13-36: Example of splitting an RDAREA with OTHERS specified

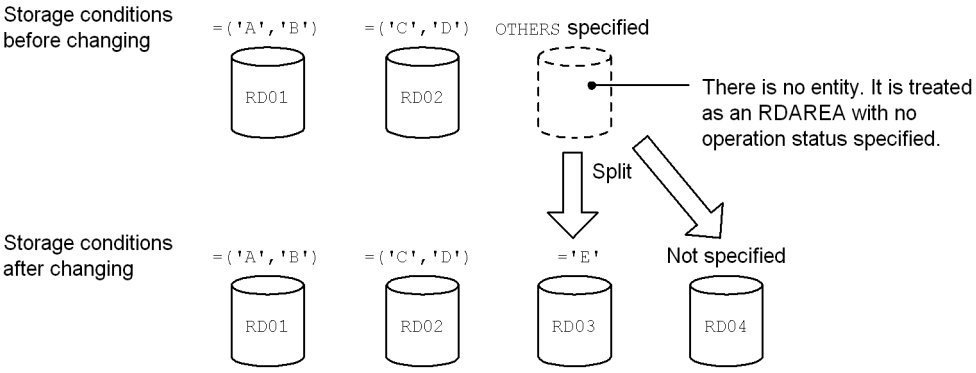
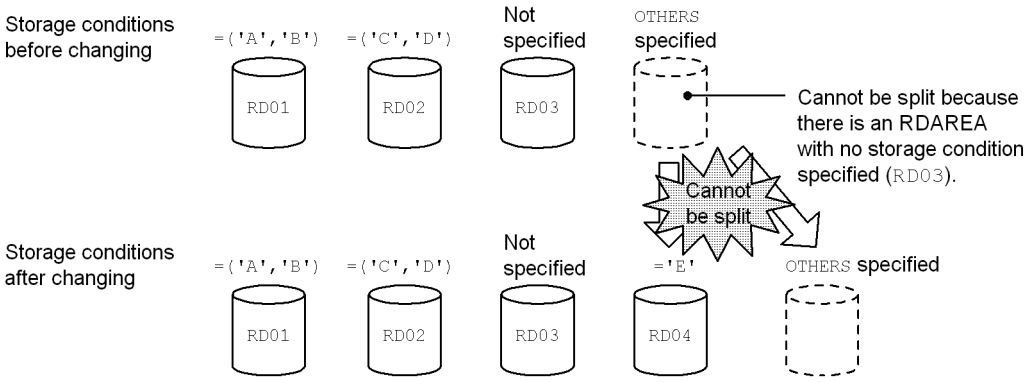


Figure 13-37: Example where an RDAREA with OTHERS specified cannot be split



(2) RDAREA with OTHERS specified

When storage conditions are specified, you can split a dummy RDAREA with no entity. You can also split an RDAREA into an RDAREA with entity and a dummy RDAREA. Such a dummy RDAREA with no entity is called an RDAREA with OTHERS specified. RDAREAs with OTHERS specified are treated as RDAREAs with no storage condition specified during split or combine processing. The following describes how to use an RDAREA with OTHERS specified.

- You can add an RDAREA with new storage conditions specified as well as an RDAREA with no storage condition specified

  The system performs split processing treating an RDAREA with OTHERS specified as an RDAREA with no storage condition specified. This makes it possible to add an RDAREA with new storage conditions specified as well as an RDAREA with no storage condition specified. For an example, see Figure *13-36 Example of splitting an RDAREA with OTHERS specified*.

- You can delete an RDAREA with no storage condition specified and add new storage conditions

  If you have split an RDAREA with no storage condition specified or an RDAREA with OTHERS specified, you can specify an RDAREA with OTHERS specified as one of the RDAREAs resulting after splitting. Because an RDAREA with OTHERS specified is treated as an RDAREA with no storage condition specified, you can delete the RDAREA with no storage condition specified or add new storage conditions by specifying an RDAREA with OTHERS specified together with an RDAREA with storage conditions specified. For an example, see *13.12.8(3) Determining the RDAREAs to be used after splitting*.

(3) Determining the RDAREAs to be used after splitting

The RDAREAs to be used to store data are determined by the RDAREAs and the post-split storage conditions specified in CHANGE RDAREA in the ALTER TABLE statement. The RDAREAs obtained after splitting may include the RDAREA subject to splitting or new RDAREAs. However, the same name cannot be specified for an RDAREA obtained after splitting.

The following describes whether or not the split-target RDAREA can be included in the post-split RDAREAs.

(a) When multiple storage conditions have been specified for the split-target RDAREA

Table 13-20 shows whether or not the split-target RDAREA (for which multiple storage conditions have been specified) can be included in the post-split RDAREAs.

Table 13-20: Whether or not the split-target RDAREA (for which multiple storage conditions have been specified) can be included in the post-split RDAREAs

Conditions for the post-split RDAREAs that are specified in CHANGE RDAREA in ALTER TABLE		Whether or not post-split RDAREAs can be specified	
		When there is an RDAREA with no storage condition specified	When storage conditions have been specified for all RDAREAs
When the split-target RDAREA is not specified	Only new RDAREAs are specified.	Y (see Figure *13-38 Case where splitting is supported (part 1)*)	
	An RDAREA with a storage condition specified is included.	N (see Figure *13-40 Case where splitting is not supported (part 1)*)	
	An RDAREA with no storage condition specified is included.	N (see Figure *13-40 Case where splitting is not supported (part 1)*)	--
	An RDAREA with OTHERS specified is specified.	N (see Figure *13-41 Case where splitting is not supported (part 2)*)	
When the split-target RDAREA is specified	All RDAREAs are new other than the split-target RDAREA.	Y (see Figure *13-39 Case where splitting is supported (part 2)*)	
	An RDAREA with a storage condition specified is included.	N (see Figure *13-40 Case where splitting is not supported (part 1)*)	
	An RDAREA with no storage condition specified is included.	N (see Figure *13-40 Case where splitting is not supported (part 1)*)	--
	An RDAREA with OTHERS specified is specified.	N (see Figure *13-41 Case where splitting is not supported (part 2)*)	

Legend:

Y: Can be split.

N: Cannot be split.

--: Not applicable.

Figure 13-38: Case where splitting is supported (part 1)

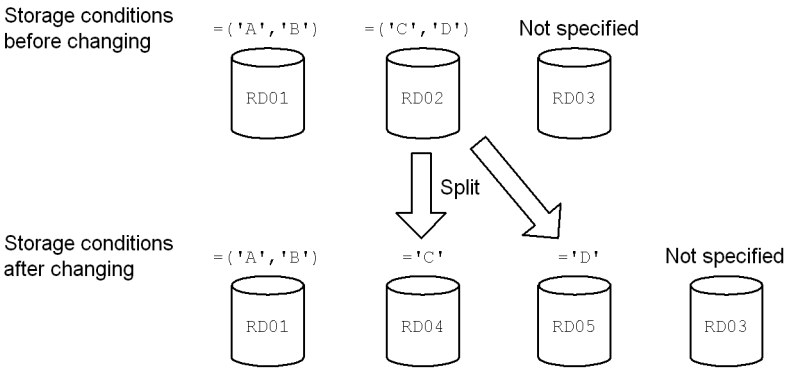


Figure 13-39: Case where splitting is supported (part 2)

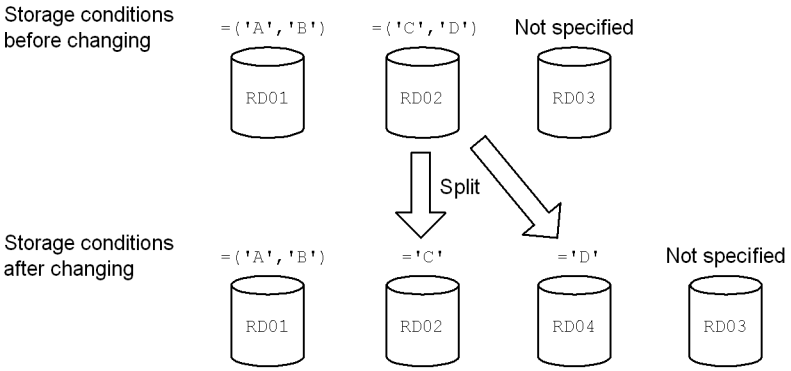
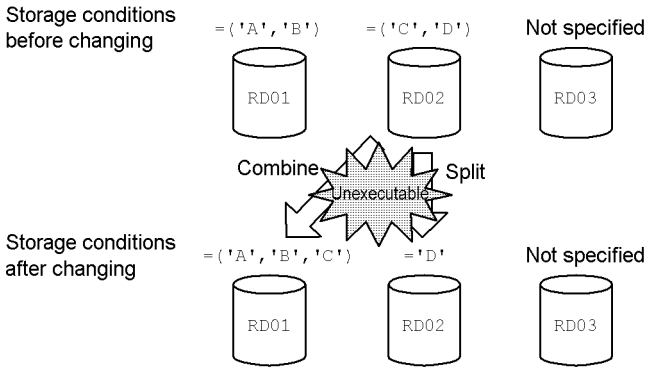
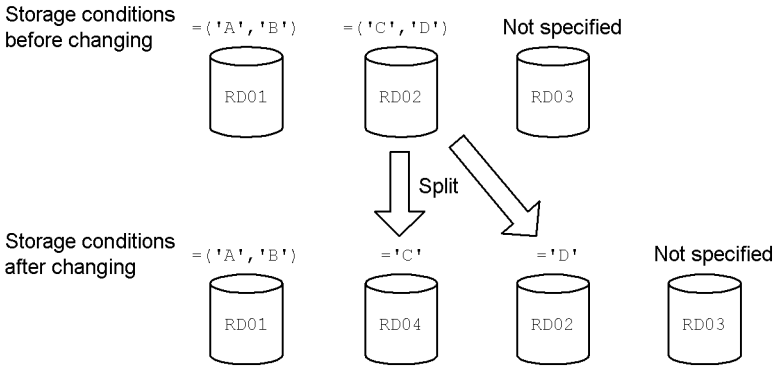


Figure 13-40: Case where splitting is not supported (part 1)



Execution of `ALTER TABLE` results in an error because an attempt was made to perform split processing and combine processing at the same time. You must perform split processing and combine processing in separate steps:

1. The first `ALTER TABLE` splits the storage conditions for `RD02`.



2. The second `ALTER TABLE` combines `RD01` and `RD04`.

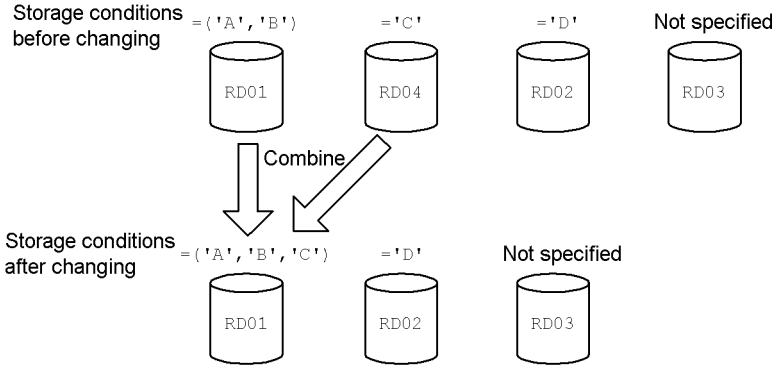
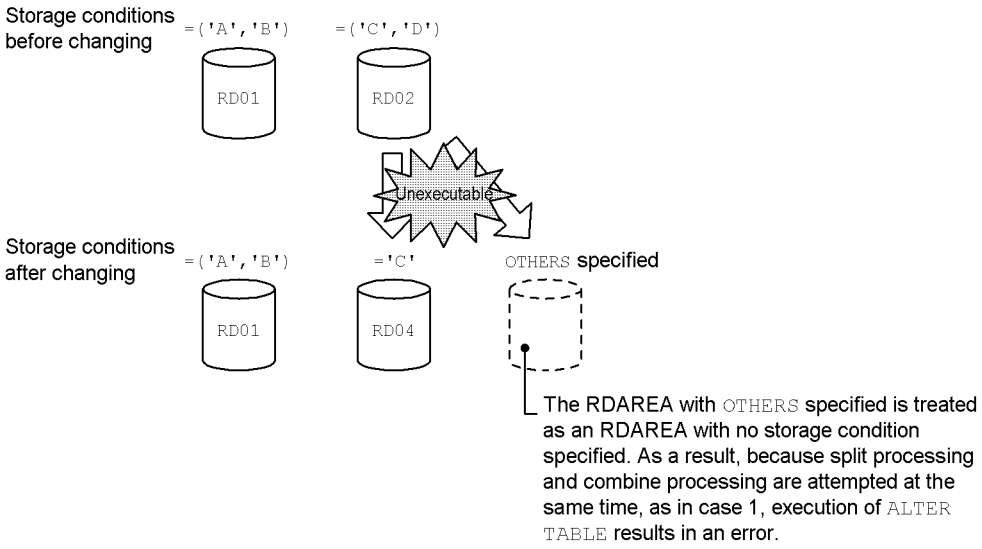


Figure 13-41: Case where splitting is not supported (part 2)



(b) When no storage condition has been specified for the split-target RDAREA

Table 13-21 shows whether or not the split-target RDAREA (for which no storage condition has been specified) can be included in the post-split RDAREAs.

Table 13-21: Whether or not the split-target RDAREA (for which no storage condition has been specified) can be included in the post-split RDAREAs

Conditions for the post-split RDAREAs that are specified in CHANGE RDAREA in ALTER TABLE			Whether or not post-split RDAREAs can be specified	
			When there is an RDAREA with no storage condition specified	When storage conditions have been specified for all RDAREAs
When the split-target RDAREA is not specified	Only new RDAREAs are specified.	No RDAREA with OTHERS specified is included.	Y (see Figure *13-42 Case where splitting is supported (part 3)*)	--
		An RDAREA with OTHERS specified is included.	Y (see Figure *13-43 Case where splitting is supported (part 4)*)	--
	An RDAREA with a storage condition specified is included.		N (see Figure *13-46 Case where splitting is not supported (part 3)*)	--
When the split-target RDAREA is specified	All RDAREAs are new other than the split-target RDAREA.	No RDAREA with OTHERS specified is included.	Y (see Figure *13-44 Case where splitting is supported (part 5)*)	--
		An RDAREA with OTHERS specified is included.	Y (see Figure *13-45 Case where splitting is supported (part 6)*)	--
	An RDAREA with a storage condition specified is included.		N (see Figure *13-46 Case where splitting is not supported (part 3)*)	--

Legend:

Y: Can be split.

N: Cannot be split.

--: Not applicable.

Figure 13-42: Case where splitting is supported (part 3)

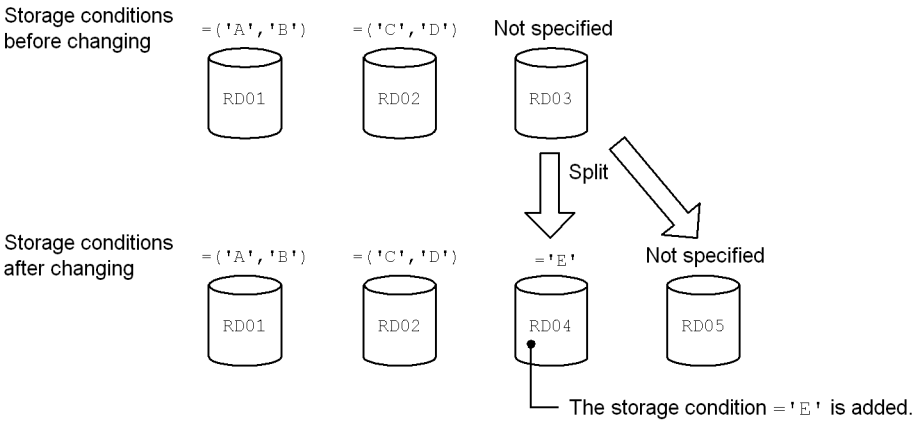
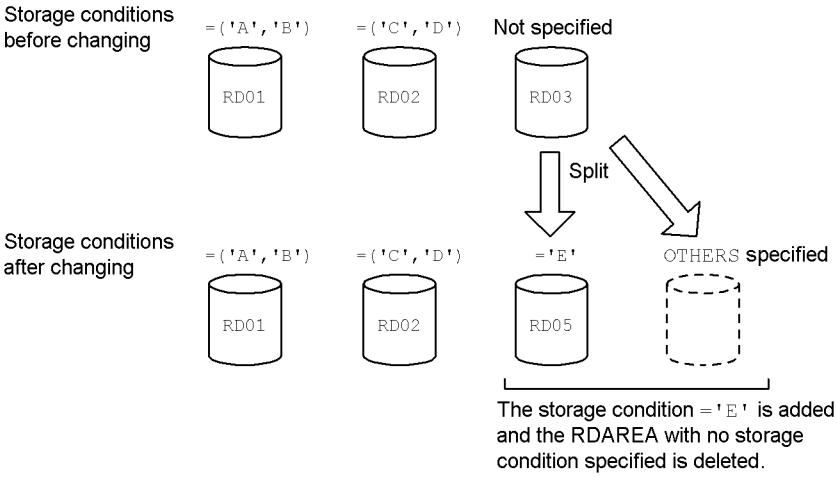


Figure 13-43: Case where splitting is supported (part 4)



Note:

In this example, data other than 'A', 'B', 'C', 'D', and 'E' can no longer be referenced or inserted after splitting. Therefore, caution is required if you specify OTHERS for a post-split RDAREA.

Figure 13-44: Case where splitting is supported (part 5)

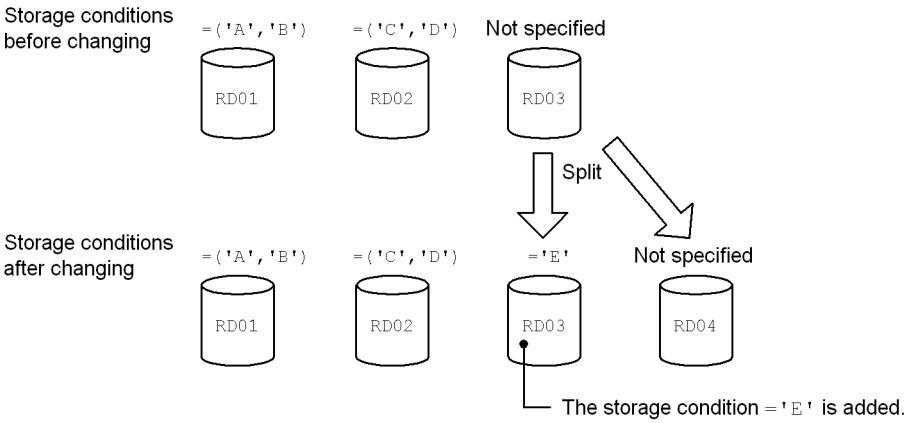
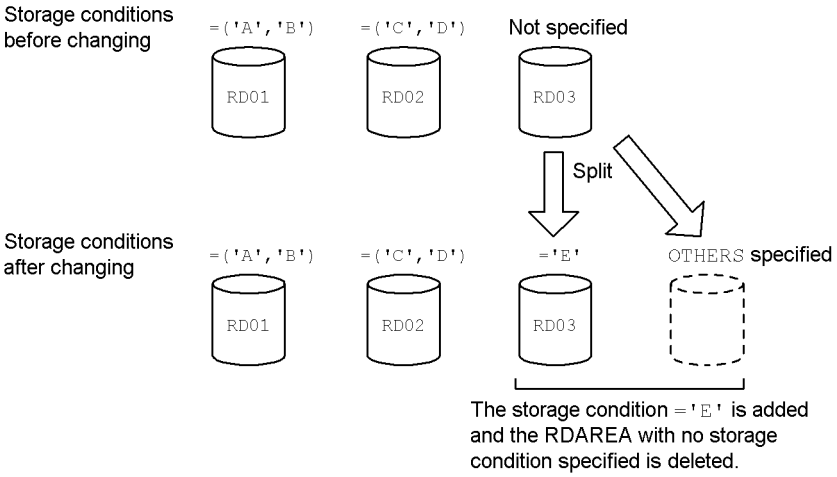


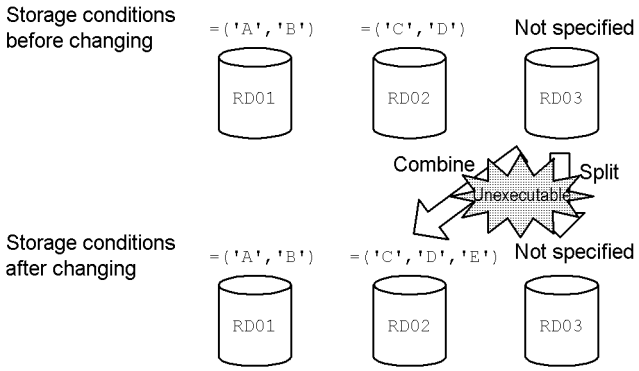
Figure 13-45: Case where splitting is supported (part 6)



Note:

In this example, data other than 'A', 'B', 'C', 'D', and 'E' can no longer be referenced or inserted after splitting. Therefore, caution is required if you specify OTHERS for a post-split RDAREA.

Figure 13-46: Case where splitting is not supported (part 3)



Execution of ALTER TABLE results in an error because an attempt was made to perform split processing and combine processing at the same time. You must perform split processing and combine processing in separate steps.

(c) When OTHERS is specified for the split-target RDAREA

Table 13-22 shows whether or not the split-target RDAREA (with OTHERS specified) can be included in the post-split RDAREAs.

Table 13-22: Whether or not the split-target RDAREA (with OTHERS specified) can be included in the post-split RDAREAs

Conditions for the post-split RDAREAs that are specified in CHANGE RDAREA in ALTER TABLE		Whether or not post-split RDAREAs can be specified	
		When there is an RDAREA with no storage condition specified	When storage conditions have been specified for all RDAREAs
The post-split RDAREAs include an RDAREA with OTHERS specified	Only new RDAREAs are specified.	N	Y (see Figure *13-47 Case where splitting is supported (part 7)*)
	An RDAREA with a storage condition specified is included.	N	N (see Figure *13-49 Case where splitting is not supported (part 4)*)
The post-split RDAREAs include no RDAREA with OTHERS specified	Only new RDAREAs are specified.	N	Y (see Figure *13-48 Case where splitting is supported (part 8)*)
	An RDAREA with a storage condition specified is included.	N	N (see Figure *13-49 Case where splitting is not supported (part 4)*)

Legend:

Y: Can be split.

N: Cannot be split.

--: Not applicable.

Figure 13-47: Case where splitting is supported (part 7)

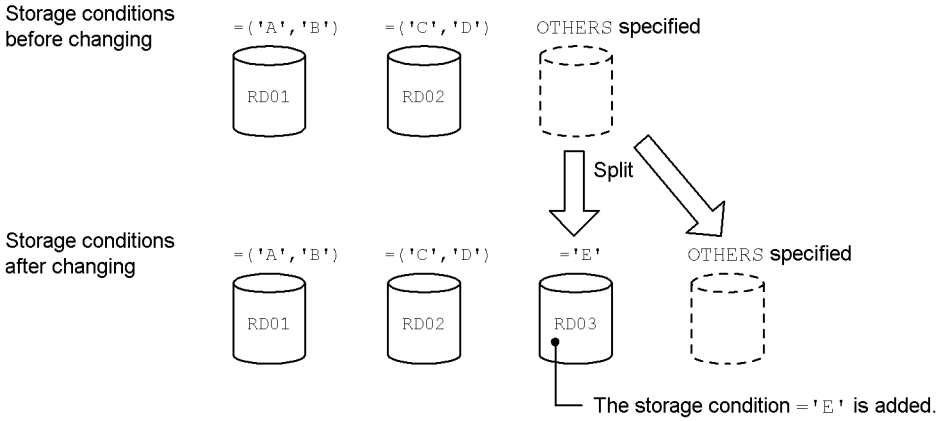


Figure 13-48: Case where splitting is supported (part 8)

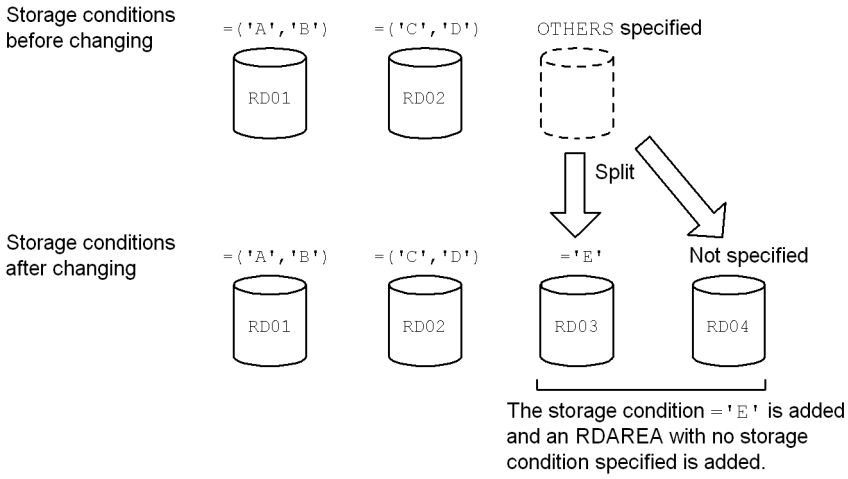
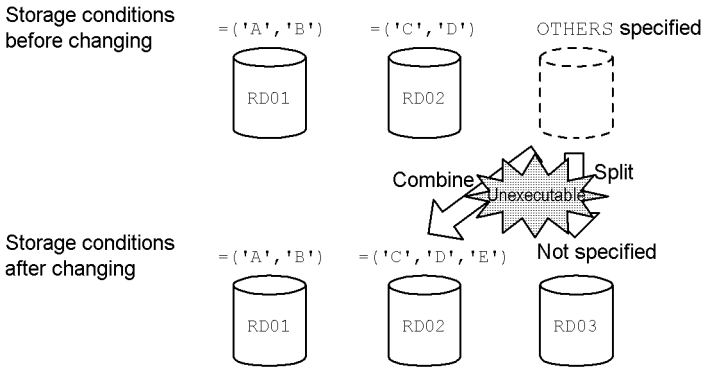


Figure 13-49: Case where splitting is not supported (part 4)



Execution of ALTER TABLE results in an error because an attempt was made to perform split processing and combine processing at the same time. You must perform split processing and combine processing in separate steps.

(4) How to specify a post-split storage condition

You must specify a post-split storage condition for the split-target RDAREA, and the specified storage condition must satisfy all the following conditions:

- The specified partitioning key value exists in the table definition.
- The data type of the column specified in a single storage condition can be compared with the data type of a literal.
- There is duplication of post-split storage conditions.

Table 13-23 shows whether or not a post-split storage condition can be specified.

Table 13-23: Whether or not a post-split storage condition can be specified

Split-target RDAREA	Post-split storage condition		Specifiable
RDAREA for which storage conditions are specified	A storage condition applicable to the split-target RDAREA is specified.	All storage conditions are specified.	Y
		At least one storage condition is not specified.	N
	Storage conditions that are not included in the split-target RDAREA are specified.		N
	No storage conditions are specified.		N

Split-target RDAREA	Post-split storage condition		Specifiable
RDAREA for which no storage condition is specified	A storage condition already specified in the table definitions is included.		N
	The storage conditions already specified in the table definitions are not included.	There is an RDAREA with no storage condition specified.	Y
		There is no RDAREA with no storage condition specified.	N
RDAREA with OTHERS specified	The storage condition already specified in the table definitions is included.		N
	The storage conditions already specified in the table definitions are not included.		Y

Legend:

Y: Can be specified.

N: Cannot be specified.

You can specify one or more post-split storage conditions for each post-split RDAREA.

(5) When a resource such as a partitioning key index has been defined for the table

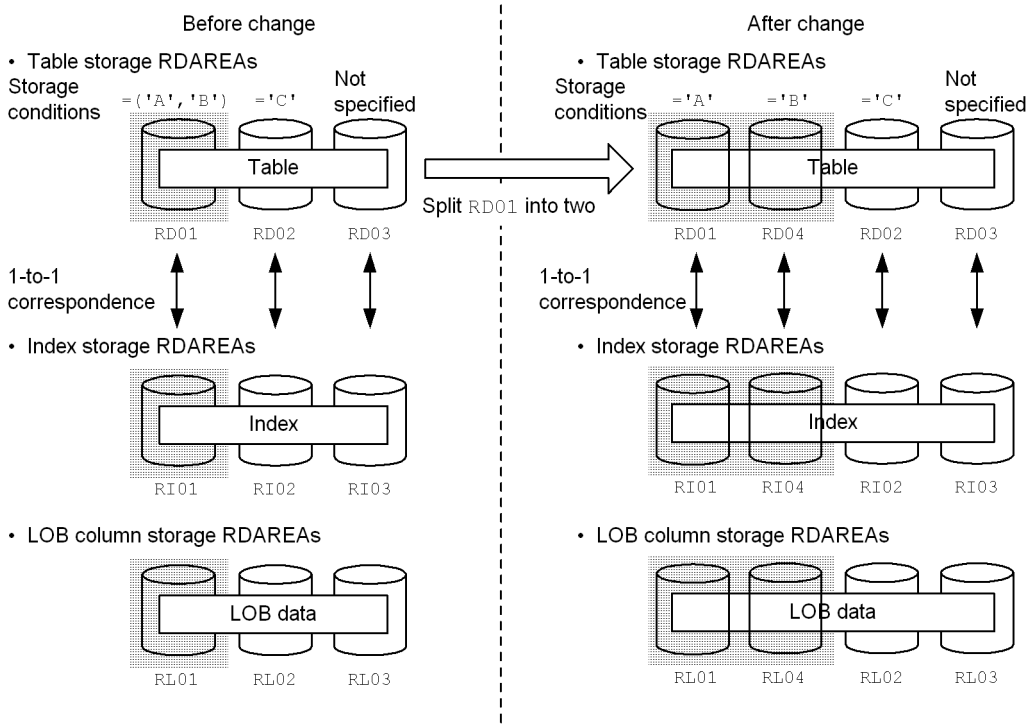
If a resource such as a partitioning key index has been defined for a table whose storage conditions are to be changed, you must also split the index storage RDAREAs when you split table storage RDAREAs. Table 13-24 shows the resources that are subject to splitting when storage conditions are changed.

Figure 13-50 shows an example of splitting when a resource such as a partitioning key index has been defined for the table.

Table 13-24: Resources subject to splitting when storage conditions are changed

Resource defined for the table	Splitting method
Index	Split in such a manner that the index storage RDAREAs have a 1-to-1 correspondence with the table storage RDAREAs.
Cluster key	
Primary key	
LOB column	

Figure 13-50: Example of splitting when a resource such as a partitioning key index has been defined for the table



During split processing, the following rules apply:

- If multiple resources have been defined, all resources are subject to splitting.
- If the specification for splitting is invalid, `ALTER TABLE` results in an error.
- There can be no duplication of names among the index storage RDAREAs and LOB column storage RDAREAs obtained after storage conditions have been changed.

(6) Handling of data in the split-target RDAREA

When `ALTER TABLE` is executed, data is normally deleted from the split-target RDAREA. Data is also deleted from the following corresponding RDAREAs:

- Index data in the index storage RDAREA
- Lob data in the LOB column storage RDAREA

Reference note:

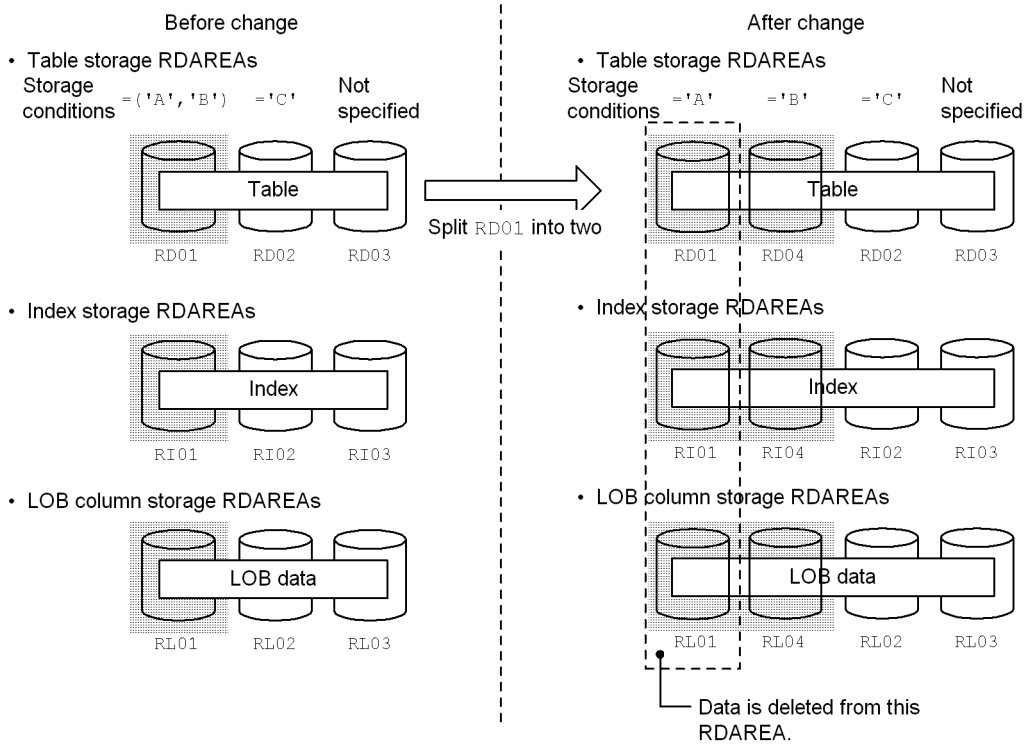
The reason for deleting data from the split-target RDAREA is as follows:

- Part of the data in the split-target RDAREA may no longer satisfy the storage conditions for the RDAREA obtained after splitting.

Note that data is not deleted from RDAREAs that are not the split target.

Figure 13-51 shows the data that is deleted during split processing.

Figure 13-51: Data that is deleted during split processing



(a) Case in which data is not deleted

If the following conditions are all satisfied, data can be retained because data in the split-target RDAREA may be usable as is:

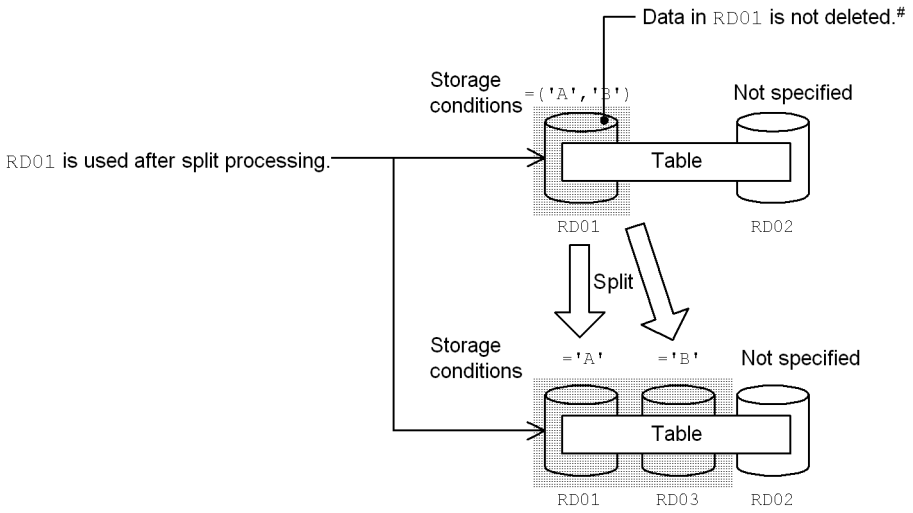
1. The split-target RDAREA is to be used after split processing.
2. The split-target RDAREA contains only data that satisfies the post-splitting storage conditions.
3. The split-target index storage RDAREA or LOB column storage RDAREA also

satisfies conditions 1 and 2.

If data is not to be deleted, specify `WITHOUT PURGE` in `ALTER TABLE`. If use of the split-target RDAREA after split processing is not specified but `WITHOUT PURGE` is specified, `ALTER TABLE` results in an error.

Figure 13-52 shows the case where `WITHOUT PURGE` takes effect, and Figure 13-53 shows the case where `WITHOUT PURGE` results in an error.

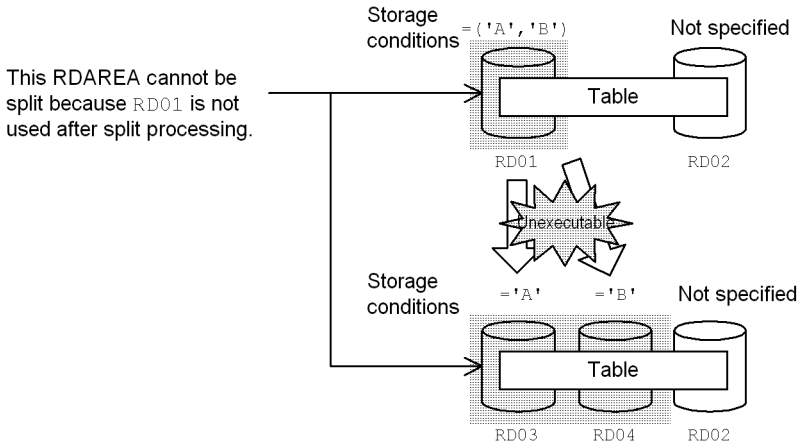
Figure 13-52: Case where WITHOUT PURGE takes effect



#

If `RD01` contains data that is `='B'` before split processing, the data that does not satisfy the storage condition remains in `RD01` after split processing. Therefore, you must make sure that `RD01` does not contain data that is `='A'` before split processing.

Figure 13-53: Case where WITHOUT PURGE results in an error



(b) Notes about specifying WITHOUT PURGE

If split processing is performed with `WITHOUT PURGE` specified, HiRDB does not check whether or not all the data in the RDAREA satisfies the post-splitting storage conditions. If the RDAREA contains data that does not satisfy the storage conditions, HiRDB does not function correctly during SQL execution. Therefore, when you change the storage conditions specifying `WITHOUT PURGE`, check the data in the split-target RDAREA.

Hint:

Normally, do not specify `WITHOUT PURGE` when you change storage conditions. For details about how to change storage conditions, see *13.13 Changing a table's partitioning storage conditions*.

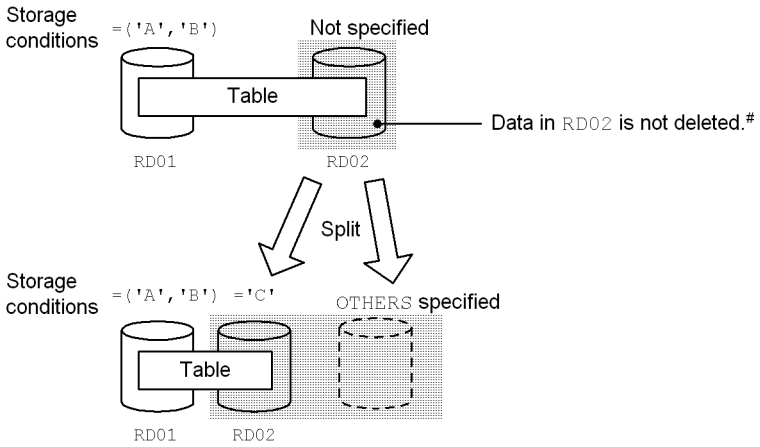
(c) Notes about specifying OTHERS for post-split RDAREAs

If both of the conditions listed below are satisfied, storage conditions are specified for all post-split RDAREAs (there will be no RDAREA with no storage condition specified after split processing):

- An RDAREA with no storage condition specified is to be split.
- An RDAREA with `OTHERS` specified is specified as a post-split RDAREA.

Any data that does not satisfy the post-splitting storage conditions will have no storage RDAREA. Figure 13-54 shows an example where there is no storage RDAREA for data after split processing.

Figure 13-54: Example where there is no storage RDAREA for data after split processing



#

Data whose storage condition is not `='C'` will have no storage RDAREA after split processing.

13.12.9 Combining RDAREAs (in the case of storage condition specification)

(1) Rules for changing storage conditions

This subsection describes the rules for changing storage conditions.

(a) Maximum and minimum values

When you change storage conditions, you must observe the maximum and minimum values shown in Table 13-25.

Table 13-25: Maximum and minimum values for the combine facility

Item	Maximum value	Minimum value	Result when the value is not within the range of the minimum and maximum values
Number of RDAREAs that can be combined in a single operation	16	2	Causes an error in `ALTER TABLE`.
Number of RDAREAs that are combined after execution of `ALTER TABLE`	1 (fixed)		

Item	Maximum value	Minimum value	Result when the value is not within the range of the minimum and maximum values
Total number of RDAREAs resulting from combining	--	1	

Legend:

--: Not applicable.

(b) Rules for combining

The RDAREAs to be combined must satisfy specific conditions. Table 13-26 describes whether or not RDAREAs can be combined.

Table 13-26: Determination of whether or not RDAREAs can be combined

RDAREAs to be combined		Whether or not combinable	Description
The combination-target RDAREAs are specified in the table definition.		Y	RDAREAs can be combined.
The combination-target RDAREAs contain an RDAREA that is not specified in the table definition.		N	Causes an error in `ALTER TABLE`.
RDAREA with `OTHERS` specified#	Storage conditions have been specified for all RDAREAs.	Y	An example of combining an RDAREA with `OTHERS` specified is shown in Figure *13-55 Example of combining an RDAREA with OTHERS specified*.
	There is an RDAREA with no storage condition specified.	N	Causes an error in `ALTER TABLE`. An example in which an RDAREA with `OTHERS` specified cannot be combined is shown in Figure *13-56 Example where an RDAREA with OTHERS specified cannot be combined*.

Legend:

Y: Can be combined.

N: Cannot be combined.

#

For details about an RDAREA with OTHERS specified, see *13.12.9(2) RDAREA with OTHERS specified*.

Figure 13-55: Example of combining an RDAREA with OTHERS specified

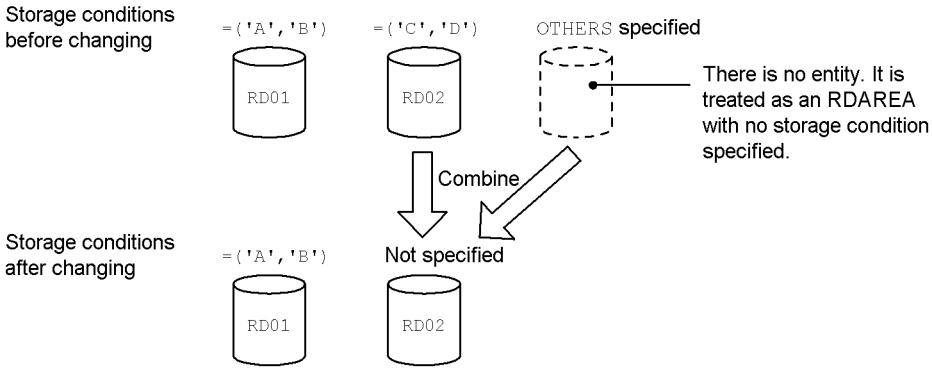
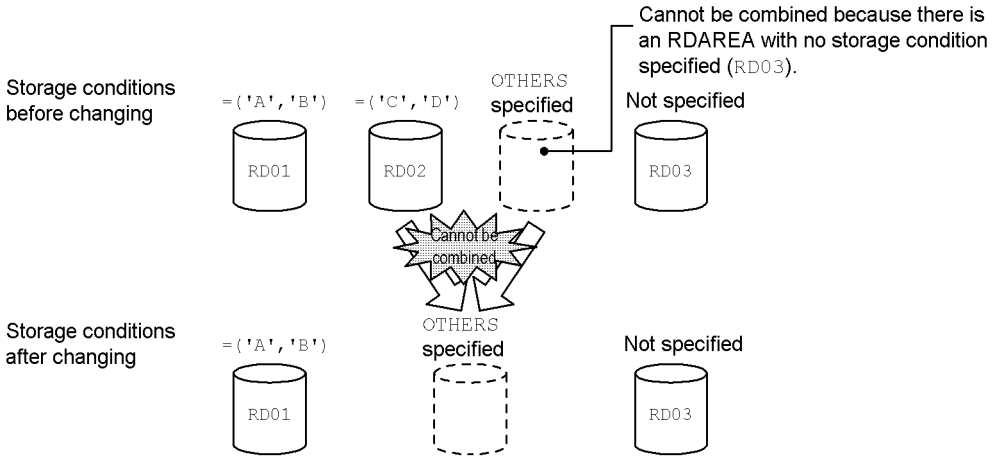


Figure 13-56: Example where an RDAREA with OTHERS specified cannot be combined



(2) RDAREA with OTHERS specified

When storage conditions are specified, you can combine dummy RDAREAs with no entity. You can also integrate an RDAREA in a dummy RDAREA. Such a dummy RDAREA with no entity is called an RDAREA with OTHERS specified. RDAREAs with OTHERS specified are treated as RDAREAs with no storage condition specified during split or combine processing. The following describes how to use RDAREAs with OTHERS specified.

- You can delete existing storage conditions and create an RDAREA with no storage condition specified.

By including an RDAREA with OTHERS specified in the combination-target RDAREAs, you can delete existing storage conditions and create an RDAREA with no storage condition specified. For an example, see Figure *13-55 Example of combining an RDAREA with OTHERS specified*.

- You can delete RDAREAs with storage conditions specified.

  If combination-target RDAREAs include an RDAREA with no storage condition specified or an RDAREA with OTHERS specified, you can specify the RDAREA with OTHERS specified as the RDAREA resulting after combine processing. Because an RDAREA with OTHERS specified is treated as an RDAREA with no storage condition specified, you can delete an RDAREA with storage conditions specified by combining an RDAREA with OTHERS specified and the RDAREA with storage conditions specified. For an example, see *13.12.9(3) How to determine the combined RDAREA*.

(3) How to determine the combined RDAREA

You specify the combined RDAREA in CHANGE RDAREA of ALTER TABLE. This RDAREA stores the data that satisfies all storage conditions for the combination-target RDAREAs. The resulting combined RDAREA may be one of the combination-target RDAREAs or a newly provided RDAREA.

Note that the RDAREAs may not be combined depending on the conditions during combine processing. This subsection describes the permitted and prohibited combinations of combination-target RDAREAs and post-combination RDAREA.

(a) Total number of post-combination RDAREAs

There is a condition on the total number of post-combination RDAREAs. Table 13-27 shows and describes the relationship between the conditions during combine processing and the total number of post-combination RDAREAs.

Table 13-27: Relationship between the conditions during combine processing and the total number of post-combination RDAREAs

Condition during combine processing		Total number of post-combination RDAREAs	
Condition for combination-target table	Condition for combination-target RDAREAs.	1	2 or more

Condition during combine processing			Total number of post-combination RDAREAs	
There is an RDAREA with no storage condition specified.	There is an RDAREA with OTHERS specified.		N#1	
	There is no RDAREA with OTHERS specified.	There is an RDAREA with no storage condition specified.	N (see Figure *13-62 Case where combine processing is not supported (part 1)*)	Y (see Figure *13-57 Case where combine processing is supported (part 1)*)
		There is no RDAREA with no storage condition specified.	--	Y (see Figure *13-58 Case where combine processing is supported (part 2)*)
There is no RDAREA with no storage condition specified.	There is an RDAREA with OTHERS specified.		N (see Figure *13-63 Case where combine processing is not supported (part 2)*)	Y (see Figure *13-59 Case where combine processing is supported (part 3)*)
	There is no RDAREA with OTHERS specified.		Y#2 (see Figure *13-60 Case where combine processing is supported (part 4)*)	Y (see Figure *13-61 Case where combine processing is supported (part 5)*)

Legend:

Y: Can be combined.

N: Cannot be combined.

--: Not applicable.

#1

If the combination-target table has an RDAREA with no storage condition specified, an RDAREA with OTHERS specified cannot be specified as a combination-target RDAREA.

#2

RDAREAs cannot be combined if an index has been defined for the combination-target table.

Figure 13-57: Case where combine processing is supported (part 1)

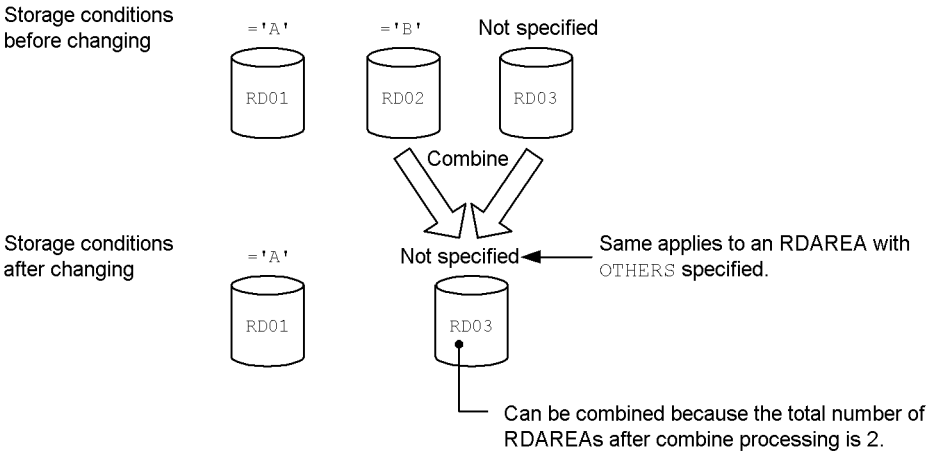


Figure 13-58: Case where combine processing is supported (part 2)

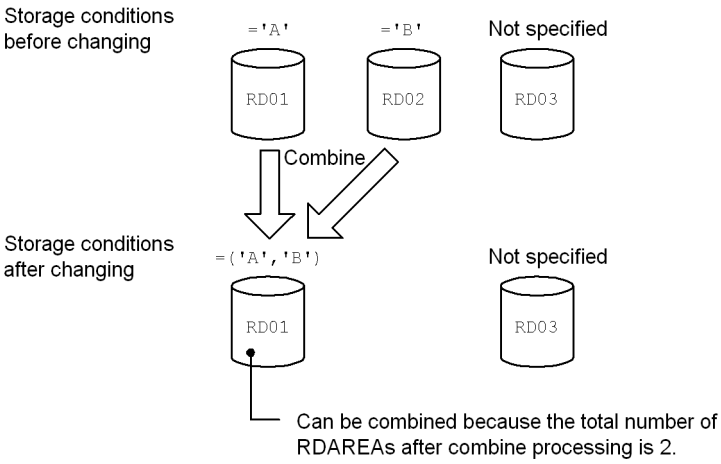


Figure 13-59: Case where combine processing is supported (part 3)

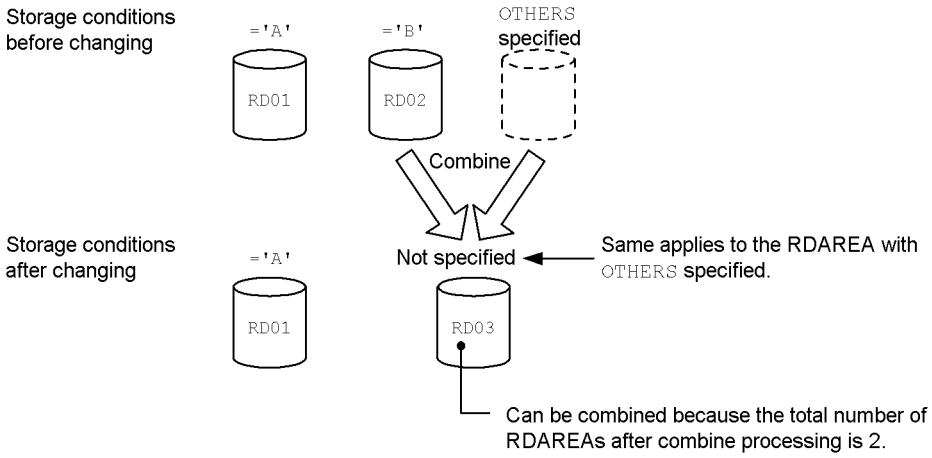


Figure 13-60: Case where combine processing is supported (part 4)

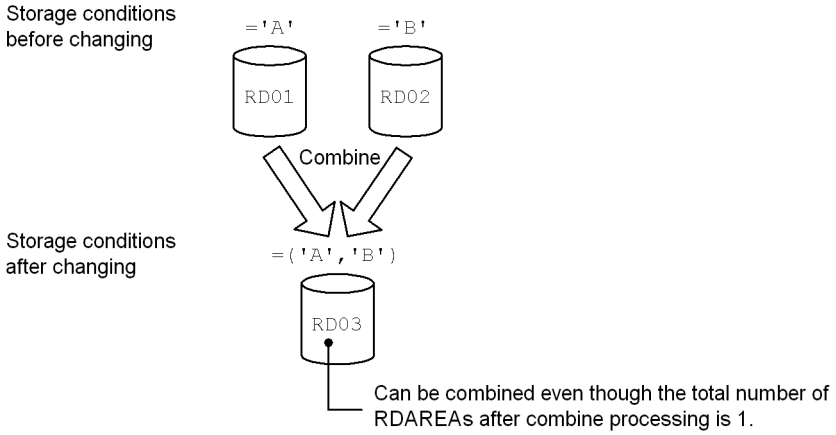


Figure 13-61: Case where combine processing is supported (part 5)

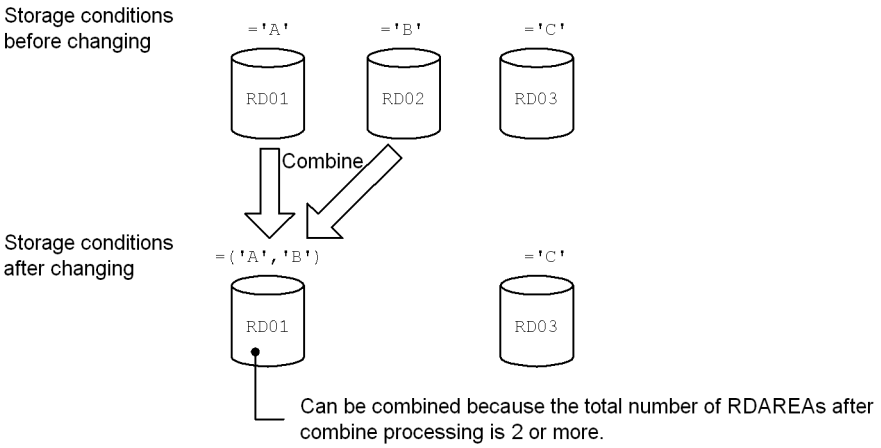


Figure 13-62: Case where combine processing is not supported (part 1)

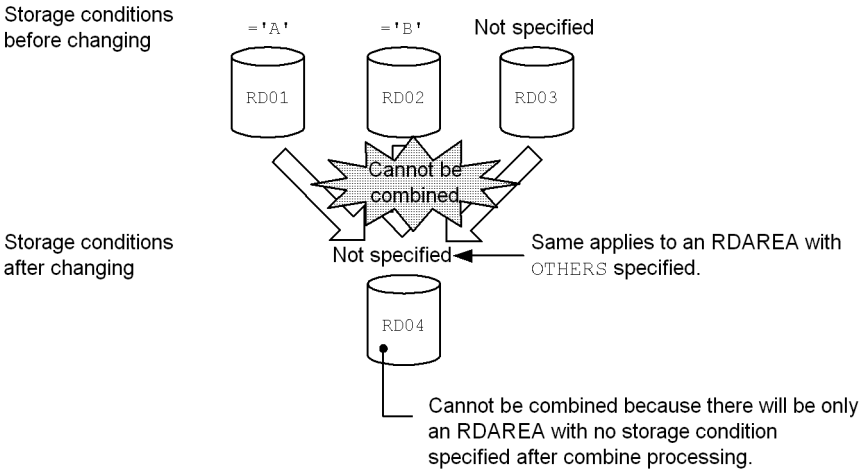
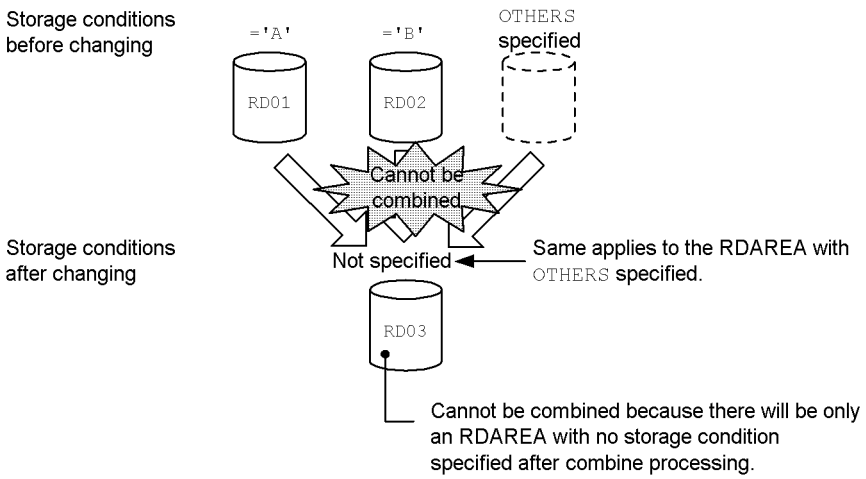


Figure 13-63: Case where combine processing is not supported (part 2)



(b) When combining only RDAREAs with storage conditions specified

Table 13-28 lists and describes the specification conditions for the post-combination RDAREA and whether or not combine processing is supported (when only RDAREAs with storage conditions specified are to be combined).

Table 13-28: Specification conditions for the post-combination RDAREA and whether or not combine processing is supported (when only RDAREAs with storage conditions specified are to be combined)

Specification condition for post-combination RDAREA		Whether or not combine processing is supported
No combination-target RDAREA is used after combine processing.	One of the table storage RDAREAs is specified as the post-combination RDAREA.	N (RDAREA specification error)
	An RDAREA other than a table storage RDAREAs is specified as the post-combination RDAREA.	Y (see Figure *13-64 Case where combine processing is supported (part 6)*)
	An RDAREA with OTHERS specified is specified as the post-combination RDAREA.	N (see Figure *13-66 Case where combine processing is not supported (part 3)*)
A combination-target RDAREA is used after combine processing.		Y (see Figure *13-65 Case where combine processing is supported (part 7)*)

Legend:

Y: Can be combined.

N: Cannot be combined.

Figure 13-64: Case where combine processing is supported (part 6)

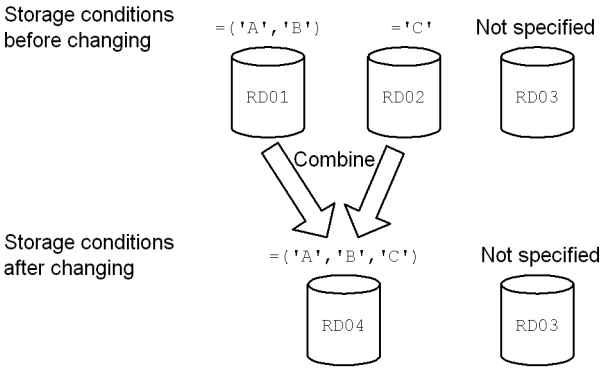


Figure 13-65: Case where combine processing is supported (part 7)

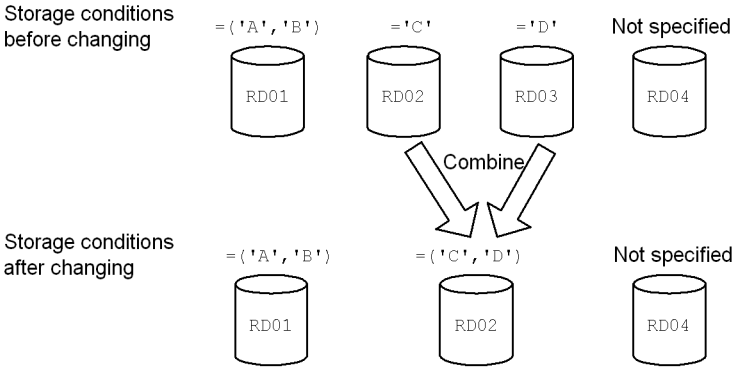
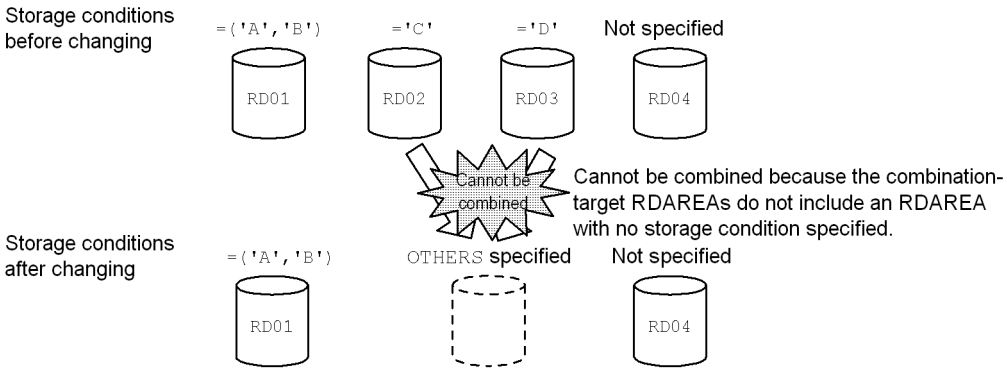


Figure 13-66: Case where combine processing is not supported (part 3)



(c) When an RDAREA with no storage condition specified is to be combined

Table 13-29 lists and describes the specification conditions for the post-combination RDAREA and whether or not combine processing is supported (when an RDAREA with no storage condition specified is to be combined).

Table 13-29: Specification conditions for the post-combination RDAREA and whether or not combine processing is supported (when an RDAREA with no storage condition specified is to be combined)

Specification condition for post-combination RDAREA		Conditions for table storage RDAREA after combine processing	
		When there is an RDAREA with no storage condition specified	When there is no RDAREA with no storage condition specified
No combination-target RDAREA is used after combine processing.	One of the table storage RDAREAs is specified as the post-combination RDAREA.	N (RDAREA specification error)	--
	An RDAREA other than one of the table storage RDAREAs is specified as the post-combination RDAREA.	Y (see Figure *13-67 Case where combine processing is supported (part 8)*)	--
	An RDAREA with OTHERS specified is specified as the post-combination RDAREA.	Y (see Figure *13-68 Case where combine processing is supported (part 9)*)	--

Specification condition for post-combination RDAREA	Conditions for table storage RDAREA after combine processing	
	When there is an RDAREA with no storage condition specified	When there is no RDAREA with no storage condition specified
A combination-target RDAREA is used after combine processing.	Y (see Figure *13-69 Case where combine processing is supported (part 10)*)	--

Legend:

Y: Can be combined.

N: Cannot be combined.

--: Not applicable.

Figure 13-67: Case where combine processing is supported (part 8)

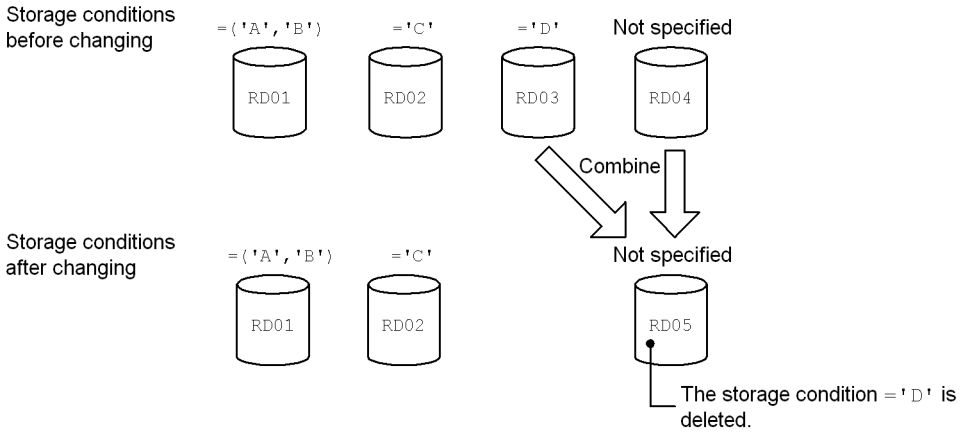


Figure 13-68: Case where combine processing is supported (part 9)

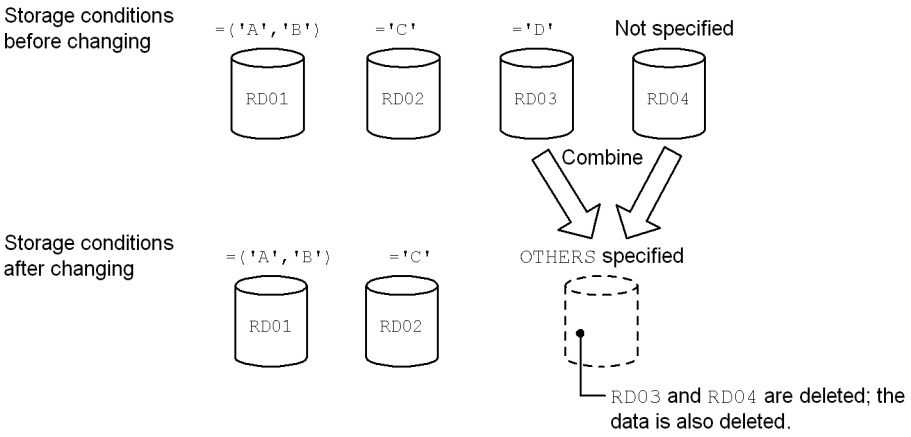
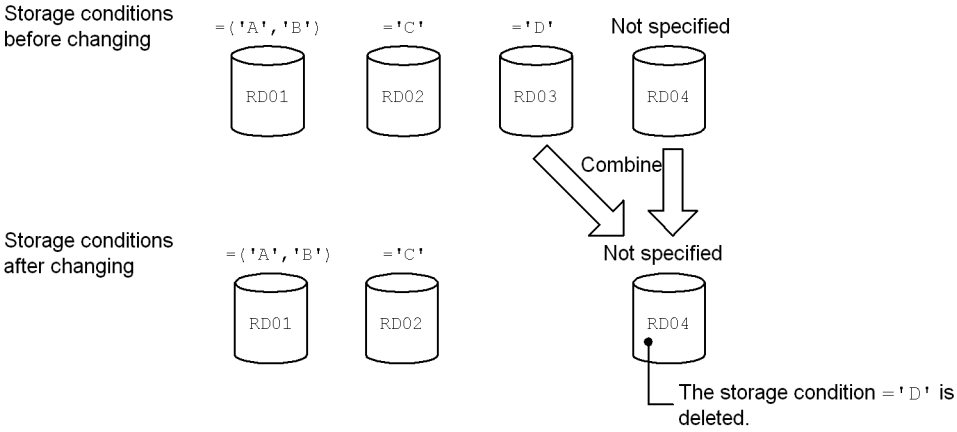


Figure 13-69: Case where combine processing is supported (part 10)



(d) When an RDAREA with OTHERS specified is to be combined

Table 13-30 lists and describes the specification conditions for the post-combination RDAREA and whether or not combine processing is supported (when an RDAREA with OTHERS specified is to be combined).

Table 13-30: Specification conditions for the post-combination RDAREA and whether or not combine processing is supported (when an RDAREA with OTHERS specified is to be combined)

Specification condition for post-combination RDAREA		Conditions for table storage RDAREA after combine processing	
		When there is an RDAREA with no storage condition specified	When there is no RDAREA with no storage condition specified
No combination-target RDAREA is used after combine processing.	An RDAREA with storage conditions specified is specified as the post-combination RDAREA.	N (OTHERS cannot be specified)	N (specified RDAREA is not a target of combine processing)
	An RDAREA other than one of the table storage RDAREAs is specified as the post-combination RDAREA.	N (OTHERS cannot be specified)	Y (see Figure *13-70 Case where combine processing is supported (part 11)*)
	An RDAREA with OTHERS specified is specified as the post-combination RDAREA.	N (OTHERS cannot be specified)	Y (see Figure *13-71 Case where combine processing is supported (part 12)*)
A combination-target RDAREA is used after combine processing.	An RDAREA with storage conditions specified is specified as the post-combination RDAREA.	N (OTHERS cannot be specified)	Y (see Figure *13-70 Case where combine processing is supported (part 11)*)
	An RDAREA with OTHERS specified is specified as the post-combination RDAREA.	N (OTHERS cannot be specified)	Y (see Figure *13-71 Case where combine processing is supported (part 12)*)

Legend:

Y: Can be combined.

N: Cannot be combined.

Figure 13-70: Case where combine processing is supported (part 11)

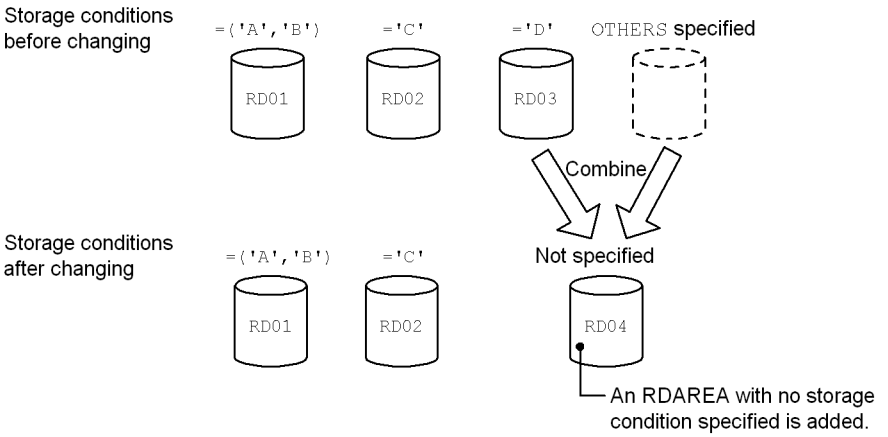
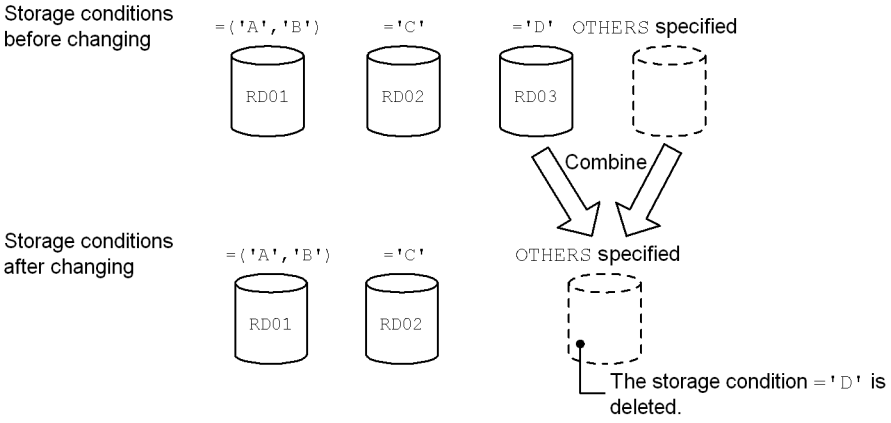


Figure 13-71: Case where combine processing is supported (part 12)



(4) When a resource such as a partitioning key index has been defined for the table

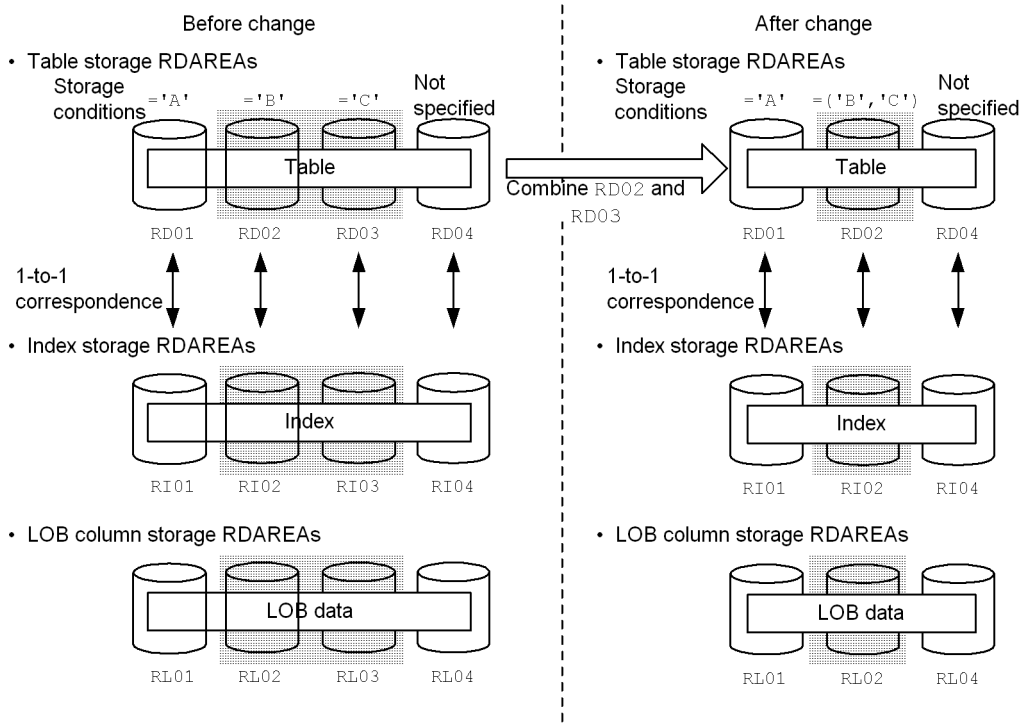
If a resource such as a partitioning key index has been defined for the combination-target table, you must combine not only the table storage RDAREAs but also the index storage RDAREAs. Table 13-31 shows the resources that must be combined at the same time.

Figure 13-72 shows an example of combining when a resource such as a partitioning key index has been defined for the table.

Table 13-31: Resources subject to combining when storage conditions are changed

Resource defined for the table	Combining method
Index	Combine so that the index storage RDAREAs have a 1-to-1 correspondence with the table storage RDAREAs.
Cluster key	
Primary key	
LOB column	

Figure 13-72: Example of combining when a resource such as a partitioning key index has been defined for the table



The following rules apply to combine processing:

- If multiple resources have been defined, all resources are subject to combining.
- If the specification for combining is invalid, `ALTER TABLE` results in an error.

(5) Handling of data in a combination-target RDAREA

When `ALTER TABLE` is executed, data is normally deleted from a combination-target

RDAREA. Data is also deleted from the following corresponding RDAREAs:

- Index data in index storage RDAREAs
- Lob data in LOB column storage RDAREAs

Reference note:

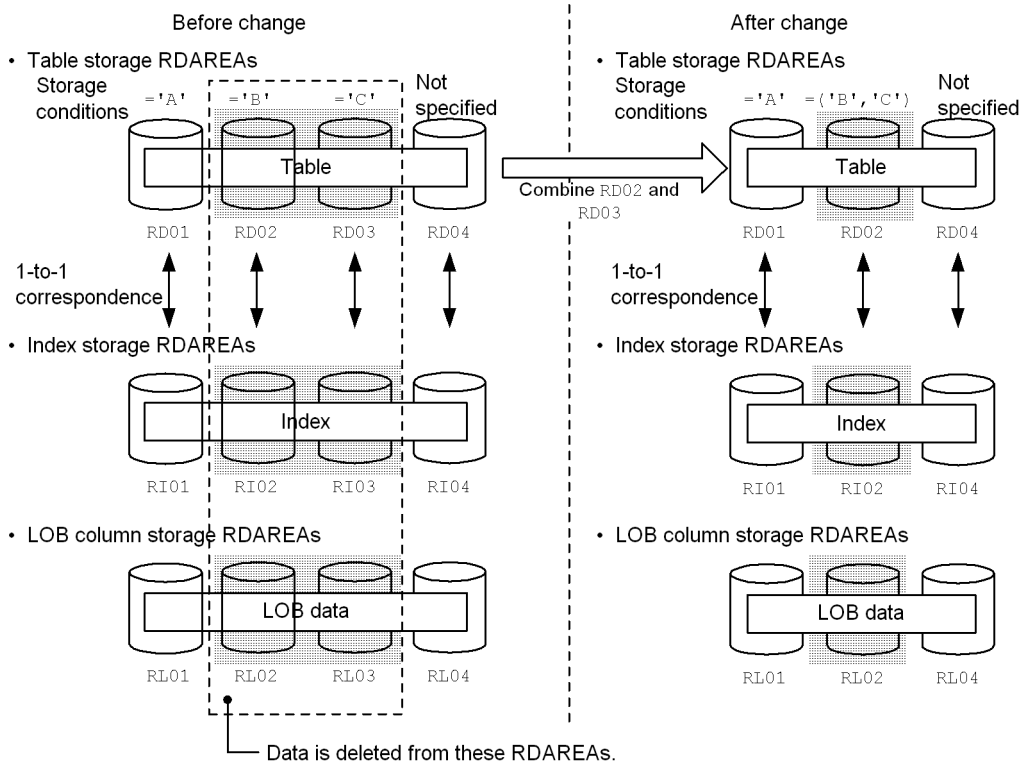
The reason for deleting data from combination-target RDAREAs is as follows:

- Combination-target RDAREAs may no longer be used after combine processing.

Note that data is not deleted from RDAREAs that are not combination targets.

Figure 13-73 shows the data that is deleted during combine processing.

Figure 13-73: Data that is deleted during combine processing



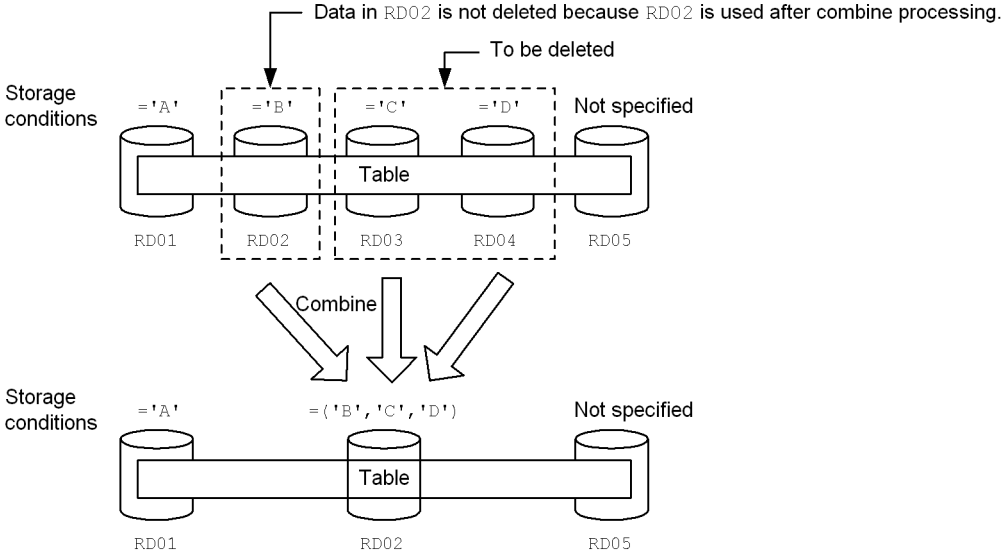
(a) Case in which data is not deleted

If the following conditions are all satisfied, data can be retained because data in the combination-target RDAREA may be usable as is:

1. The combination-target RDAREA is to be used after combine processing.
2. The combination-target index storage RDAREA or LOB column storage RDAREA also satisfies condition 1.

If data is not to be deleted, you must specify `WITHOUT PURGE` in `ALTER TABLE`. If use of a combination-target RDAREA after combine processing is not specified but `WITHOUT PURGE` is specified, `ALTER TABLE` results in an error. Figure 13-74 shows the processing when `WITHOUT PURGE` is specified.

Figure 13-74: Processing when WITHOUT PURGE is specified



Explanation:

Because RDAREAs are combined into `RD02`, the data in `RD02` (whose storage condition is `='B'`) is not deleted. The data in `RD03` and `RD04` (whose storage conditions are `='C'` and `='D'`) is deleted.

Note:

If data is saved with `WITHOUT PURGE` specified and the pre-combination data is unloaded and then loaded into the post-combination RDAREA, that data is registered twice. Therefore, if you save data with `WITHOUT PURGE` specified, do not perform data unloading or data loading.

In the case of Figure 13-74, if the data in `RD02` (whose storage condition is `='B'`) is unloaded and then loaded into the post-combination RDAREA, the data is registered twice.

(b) Notes about combining into an RDAREA with OTHERS specified

If combine processing is performed when both the following conditions are satisfied, there will be no RDAREA for storing pre-combination data:

- A combination-target RDAREA is an RDAREA with no storage condition specified or `OTHERS` specified.
- The data is to be combined into an RDAREA with `OTHERS` specified.

Figures 13-75 and 13-76 show the cases where there will be no RDAREA for storing data after combine processing.

Figure 13-75: Case where there will be no RDAREA for storing data after combine processing (part 1)

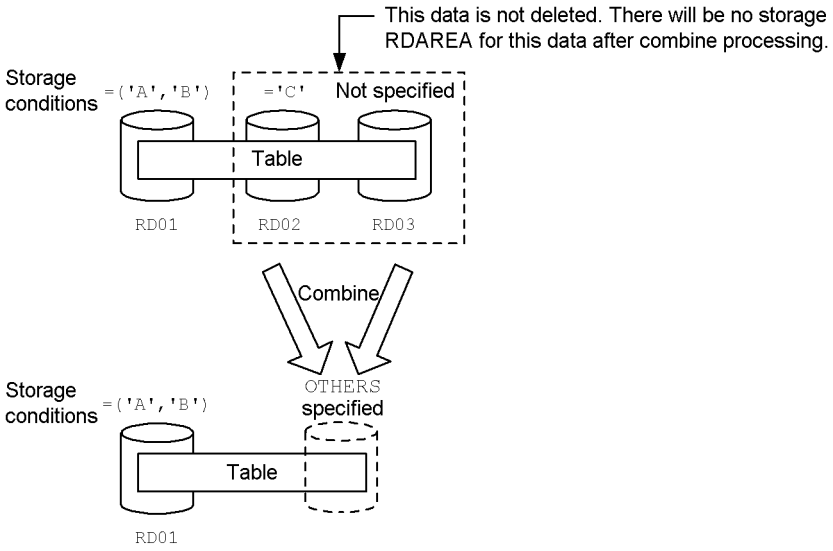
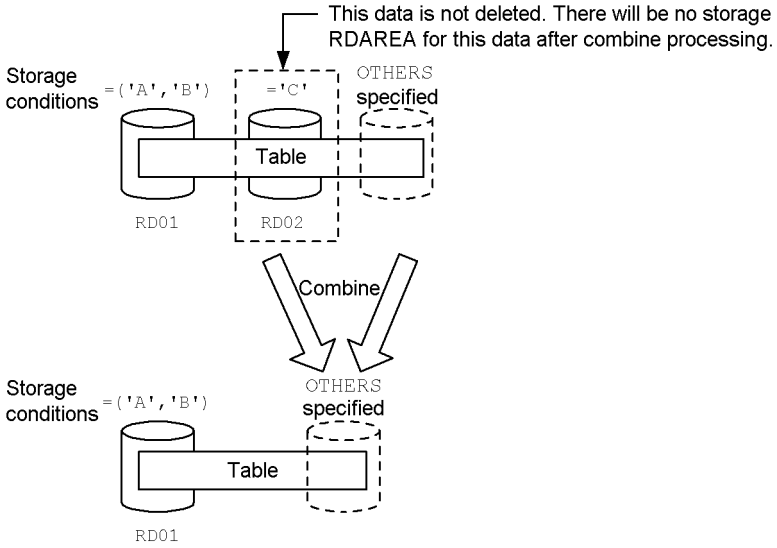


Figure 13-76: Case where there will be no RDAREA for storing data after combine processing (part 2)



13.12.10 Relationship with other facilities

(1) Invalidating related resources

If a routine has been defined for a table whose partitioning storage conditions are changed, the routine is invalidated. This also includes a routine created using a trigger definition (including a trigger routine created internally by HiRDB in a referential constraint operation).

You must use `ALTER ROUTINE` to re-create the invalidated routine.

(2) Referential constraint and check constraint

For details about the handling when partitioning storage conditions are changed for a referenced table, a referencing table, and a check constraint table, see *13.13.10 Handling when referential constraint and check constraint are used*.

(3) Inner replica facility

When the inner replica facility is used, you must synchronize the generation numbers of all the tables whose partitioning storage conditions are to be changed and the RDAREAs storing the related resources. If referential constraint is used, you must synchronize the generation numbers of all RDAREAs storing referencing tables and referenced tables. Note that when partitioning storage conditions are changed, data is deleted from all generations of RDAREAs.

(4) Concurrent execution with a utility

You cannot change partitioning storage conditions while a utility that updates the database is executing (such as the database load utility or the database reorganization utility). If you change partitioning storage conditions at such a time, data that does not satisfy the partitioning storage conditions may be registered in the database.

13.13 Changing a table's partitioning storage conditions

This section describes how to change a table's partitioning storage conditions.

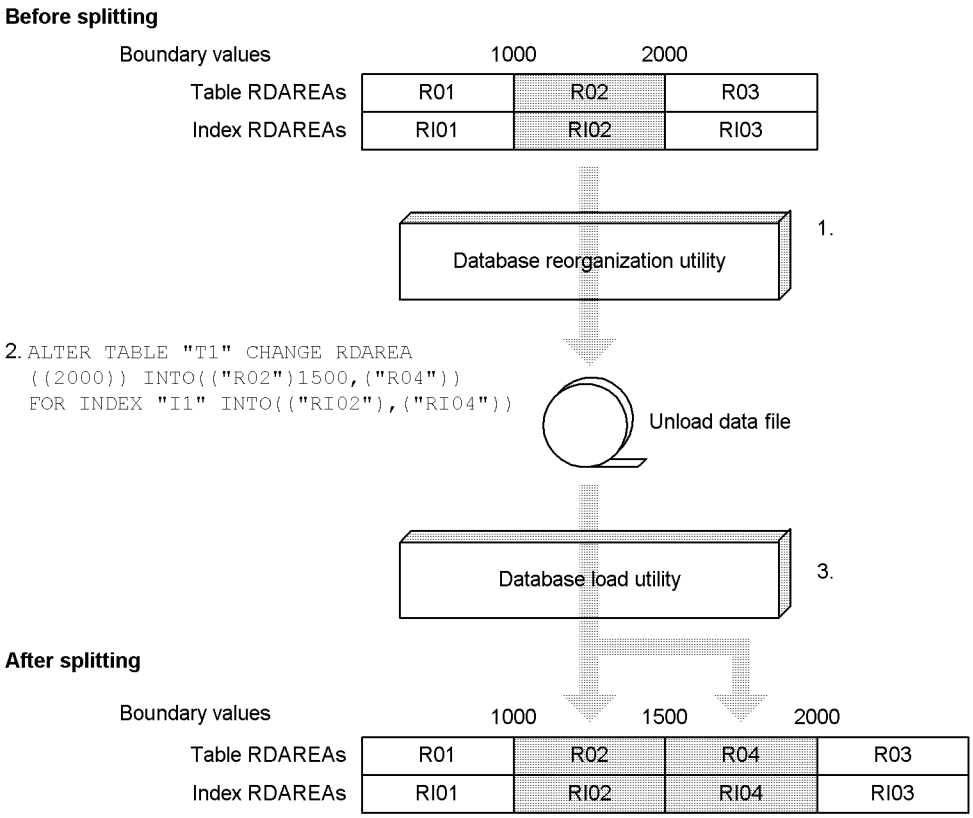
13.13.1 Examples (in the case of boundary value specification)

This subsection describes how to change a table's partitioning storage conditions (in the case of boundary value specification).

You should always make a backup before you change a table's partitioning storage conditions.

(1) Example 1 (basic split operation)

This section explains how to split a data storage range by creating new RDAREAs when a space shortage occurs in the RDAREA for a given storage range. You may use the original RDAREAs as post-splitting RDAREAs or you may create completely new RDAREAs. This subsection describes an example of the basic split operation (in which the original RDAREA is used).



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the procedure below.

Procedure

1. Use the database reorganization utility (`pdrorg`) to unload the table data from the RDAREA (`R02`) whose storage condition is to be changed and save it in a format that can be used as the input to the database load utility (`pdload`).
2. Use `ALTER TABLE` to specify the new partitioning boundary values. During this process, data in the splitting-target RDAREAs (`R02` and `RI02`) is deleted.
3. Using the unload data file created in step 1 as the input file, execute the database load utility (`pdload`) to load data in the addition mode into the split RDAREAs for storing the partitioned boundary values on an RDAREA-by-RDAREA basis.

Because the unload data file contains data that does not match the split storage conditions, data error information is output. If you do not need the data error information, you can suppress its output by specifying `divermsg=off` in the `option` statement.

4. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
5. Use `ALTER ROUTINE` to re-create the routines and triggers that were invalidated in step 2.

(2) Example 2 (splitting an RDAREA without deleting its data and using the pre-split RDAREA after splitting)

This section explains how to create new RDAREAs for future use for data in RDAREAs that increase in size over time. In this case, it is assumed that the RDAREA to be split does not store data with a larger value than the post-splitting boundary value.

This subsection describes an example of splitting an RDAREA without deleting its data and using the pre-split RDAREA after splitting. In this example, the registration date is used as the partitioning key, and it is assumed that no data for 2005 and beyond is stored in `R03`. The example creates in the table a new RDAREA for storing data for 2005 and beyond.

Before splitting

1. `SELECT MAX("C1") FROM "T1"`
   `WHERE "C1">'2004-12-31'`
   This confirms that there are no data hits (no data exists for 2005 and beyond)

Boundary values (partitioning key = "C1")

	'2002-12-31'	'2003-12-31'	
Table RDAREAs	R01	R02	R03
Index RDAREAs	RI01	RI02	RI03

2. `ALTER TABLE "T1" CHANGE RDAREA`
   `((MAX)) INTO(("R03")'2004-12-31',("R04"))`
   `FOR INDEX "I1" INTO(("RI03"),("RI04"))`
   `WITHOUT PURGE`

After splitting

Boundary values

	'2002-12-31'	'2003-12-31'	'2004-12-31'	
Table RDAREAs	R01	R02	R03	R04
Index RDAREAs	RI01	RI02	RI03	RI04

Note: The numbers in the figure (1. and 2.) correspond to the numbers in the procedure below.

Procedure

1. Confirm that the RDAREA containing the splitting-target boundary value contains only data for the post-splitting storage condition (`SELECT MAX(`*partitioning-key-column-name*`) FROM` *table-name* `WHERE` *partitioning-key-column-name* > *post-splitting-boundary-value*). If data other than data that satisfies the post-splitting storage condition (data for 2005 and beyond) is present, use the procedure in *13.13.1(1) Example 1 (basic split operation)* above.

   If you do not execute this step and data not satisfying the post-*splitt*ing storage condition remains, you can implement a recovery procedure. For details about this procedure, see *13.13.9(2) Recovery procedure when data not satisfying the post-splitting storage condition remains*.

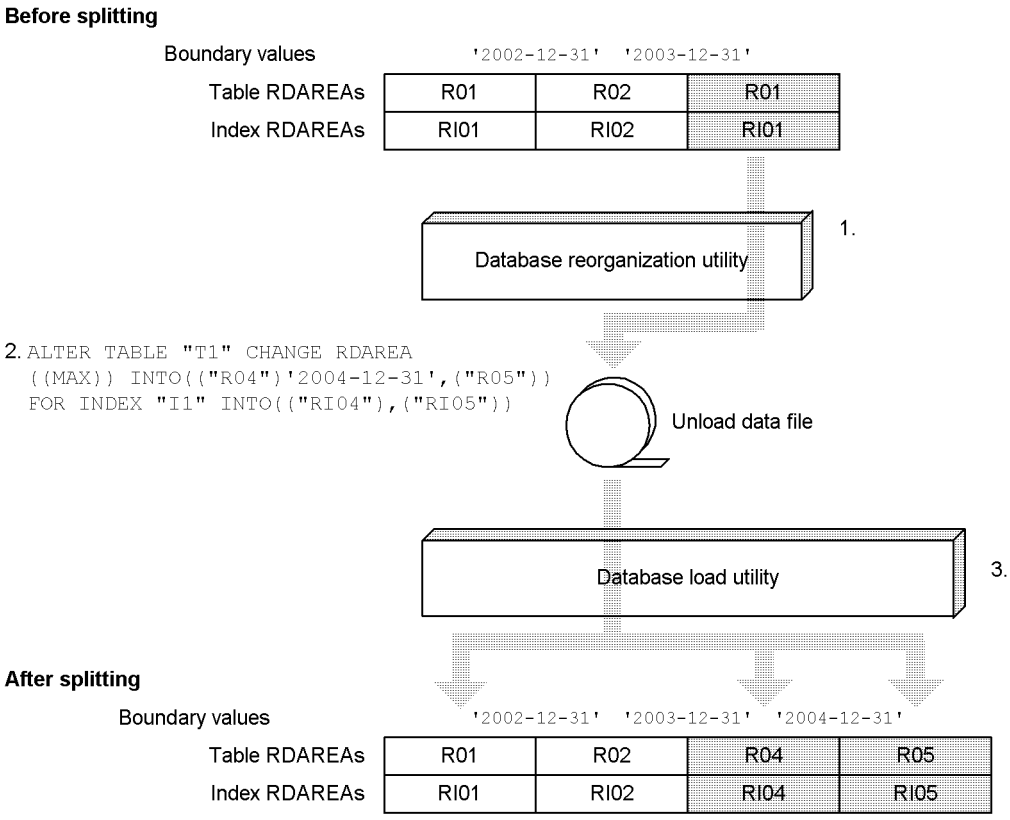
2. Use `ALTER TABLE` to specify the new partitioning boundary values. For this step, specify `WITHOUT PURGE`.

3. Use `ALTER ROUTINE` to re-create the routines and triggers that were invalidated in step 2.

(3) Example 3 (operation in which the split-target RDAREA is being used for another boundary value)

If the splitting-target RDAREA is being used for another boundary value (storage range), the data for the other boundary value is also deleted. For this reason, when the splitting-target RDAREA is being used for another boundary value, you must be careful when you use the database load utility (`pdload`) to load the data that was unloaded by the database reorganization utility (`pdrorg`) before splitting.

This subsection describes an example of the operation when the split-target RDAREA is being used for another boundary value.



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the procedure below.

Procedure

1. Use the database reorganization utility (`pdrorg`) to unload the table data from the RDAREA whose storage condition is to be changed and save it in a format that can be used as the input to the database load utility (`pdload`). During this process, data for `'2002-12-31'` or earlier is also unloaded.
2. Use `ALTER TABLE` to specify the new partitioning boundary values. During this process, data in the splitting-target RDAREAs (`R01` and `RI01`) (including data for `'2002-12-31'` or earlier) is deleted.
3. Using the unload data file created in step 1 as the input file, execute the database load utility (`pdload`) to load data in the addition mode into the split RDAREAs for storing the partitioned boundary values on an RDAREA-by-RDAREA basis. During this process, you must load data into `R01` as well because the data in `R01` was also deleted in step 2.

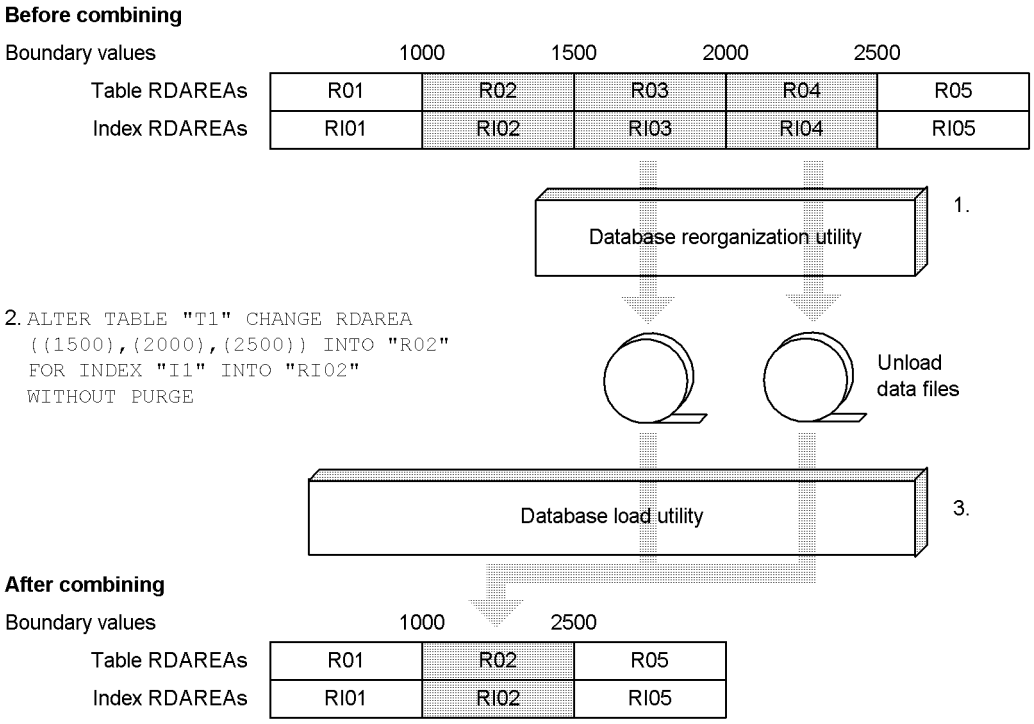
Because the unload data file contains data that does not match the split storage conditions, data error information is output. If you do not need the data error information, you can suppress its output by specifying `divermsg=off` in the `option` statement.

4. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
5. Use `ALTER ROUTINE` to re-create the routines and triggers that were invalidated in step 2.

(4) Example 4 (operation in which a pre-combination RDAREA is used as the post-combination RDAREA)

This section explains how to combine data storage ranges into a single RDAREA when RDAREAs in a contiguous storage have much free space. The post-combination RDAREA may use one of the pre-combination RDAREAs or some other RDAREA.

This subsection describes an example of the operation when one of the pre-combination RDAREAs is used as the post-combination RDAREA.



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the

procedure below.

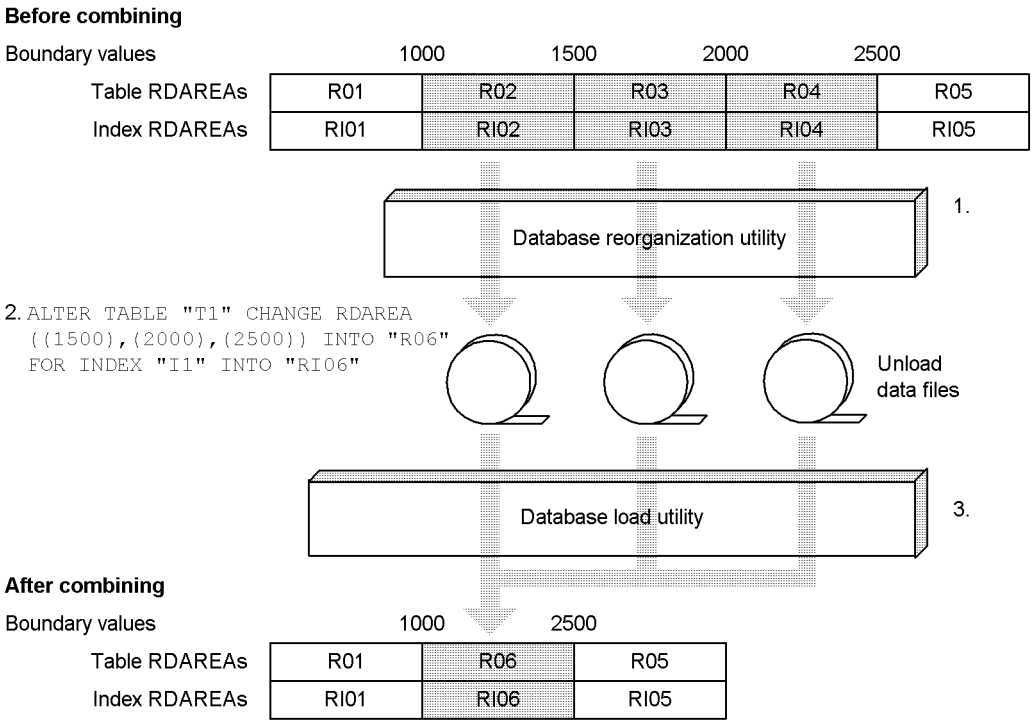
Procedure

1. Use the database reorganization utility (pdrorg) to unload the table data from all combination-target RDAREAs that will not be used after combining (R03 and R04) and save it in a format that can be used as the input to the database load utility (pdload).
2. Use ALTER TABLE to specify the boundary values to be combined. Specify WITHOUT PURGE for R02 because this is the RDAREA that will be used again after combining.
3. Using all unload data files created in step 1 as the input files, execute the database load utility (pdload) to load data in the addition mode into the combined RDAREA for storing the combined boundary values. You must be careful during this process because if you load data in the creation mode, the data in R02 before combining will be deleted. If you load data in the creation mode by mistake, use a backup to restore the system to its status before you executed ALTER TABLE, then start over from the beginning.
4. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
5. Use ALTER ROUTINE to re-create the routines and triggers that were invalidated in step 2.

(5) Example 5 (operation in which no pre-combination RDAREA is used as the post-combination RDAREA)

This example combines a data storage range into a single RDAREA because there is a large amount of free space in the RDAREAs for the particular contiguous storage range. One of the pre-combination RDAREAs may or may not be used as the post-combination RDAREA.

This subsection describes an example of the operation when no pre-combination RDAREA is used as the post-combination RDAREA.



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the procedure below.

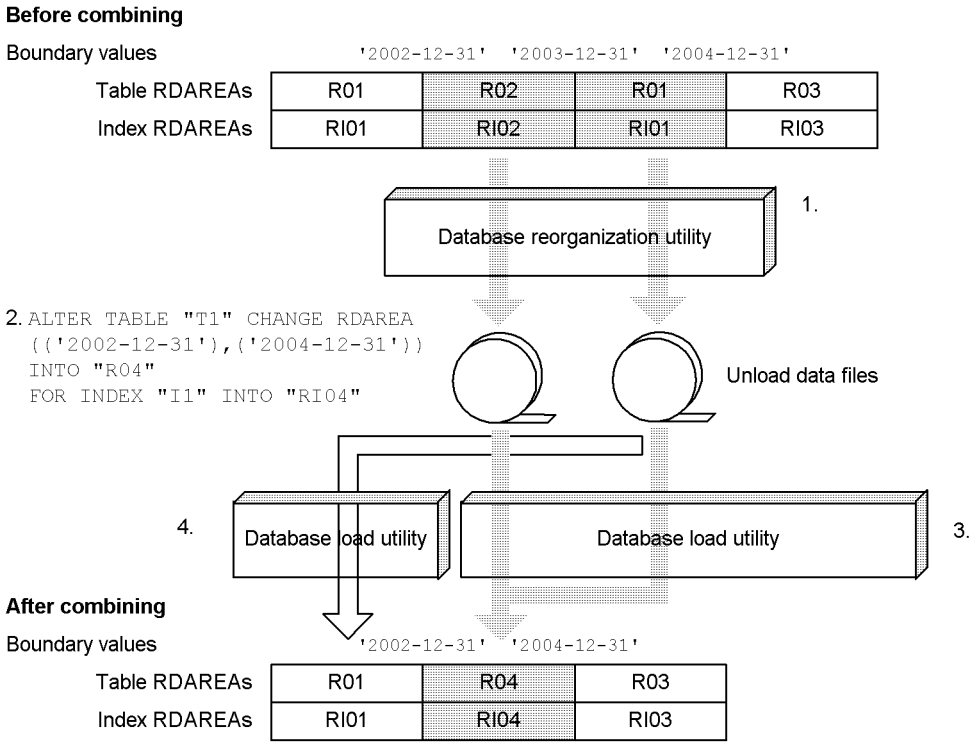
Procedure

1. Use the database reorganization utility (`pdrorg`) to unload the table data from all combination-target RDAREAs (`R02`, `R03`, and `R04`) and save it in a format that can be used as the input to the database load utility (`pdload`).
2. Use `ALTER TABLE` to specify the boundary values to be combined.
3. Using all the unload data files created in step 1 as the input files, execute the database load utility (`pdload`) to load data in the addition mode into the combined RDAREA for storing the combined boundary values, on an RDAREA-by-RDAREA basis.
4. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
5. Use `ALTER ROUTINE` to re-create the routines and triggers that were invalidated in step 2.

(6) Example 6 (operation in which a combination-target RDAREA is being used for another boundary value)

If a combination-target RDAREA is also being used for another boundary value (storage range), the data for the other boundary value is also deleted. Therefore, you must be careful to use the database load utility (`pdload`) after combining to load the data that was unloaded by the database reorganization utility (`pdrorg`) before combining into the RDAREA whose data was deleted.

This subsection describes an example of the operation when a combination-target RDAREA is being used for another boundary value.



Note: The numbers in the figure (1. through 4.) correspond to the numbers in the procedure below.

Procedure

1. Use the database reorganization utility (`pdrorg`) to unload the table data from all combination-target RDAREAs and save it in a format that can be used as the input to the database load utility (`pdload`). During this process, data for `'2002-12-31'` or earlier is also unloaded.

2. Use `ALTER TABLE` to specify the boundary values to be combined. During this process, data in the RDAREA (`R01`) being used for the other boundary value is also deleted.
3. Using all unload data files created in step 1 as the input files, execute the database load utility (`pdload`) to load data in the addition mode into the combined RDAREA for storing the combined boundary values, on an RDAREA-by-RDAREA basis.
4. The data in `R01`, which is being used for the other boundary value, has been deleted. Therefore, using the unload data file containing the data from `R01` as the input, use the database load utility (`pdload`) to add the data in the addition mode on an RDAREA-by-RDAREA basis.
5. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
6. Use `ALTER ROUTINE` to re-create the routines and triggers that were invalidated in step 2.

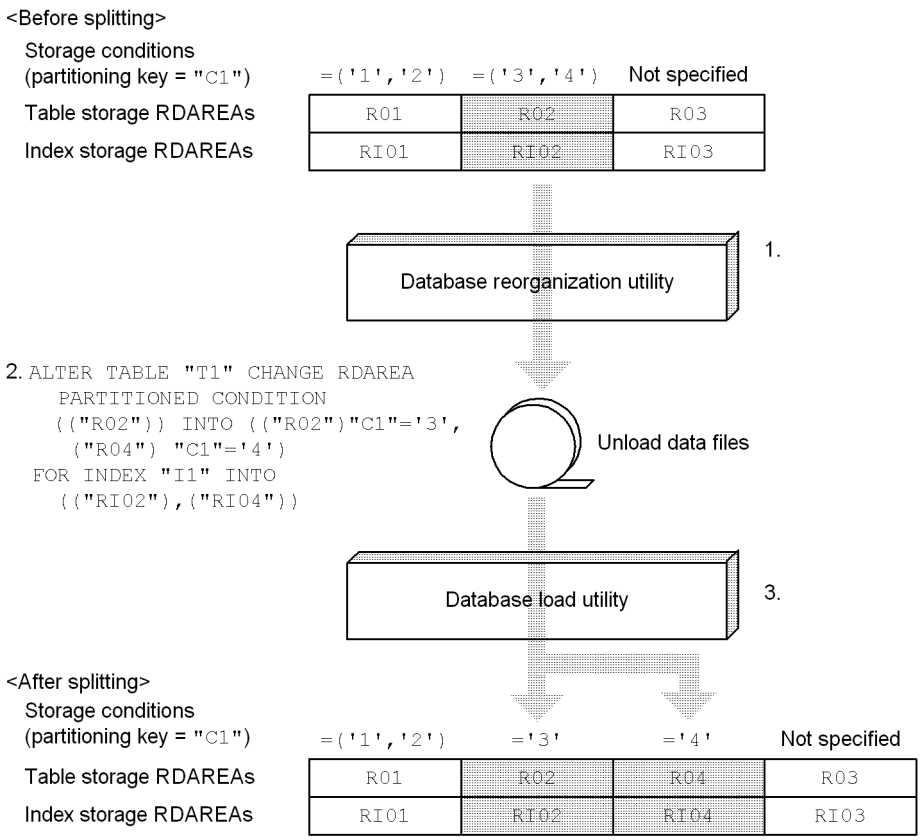
13.13.2 Examples (in the case of storage condition specification)

This subsection describes how to change a table's partitioning storage conditions (in the case of storage condition specification).

You should always make a backup before you change a table's partitioning storage conditions.

(1) Example 1 (splitting an RDAREA for which storage conditions with multiple values have been specified)

This example changes the partitioning storage conditions and splits RDAREA `R02` because it does not have sufficient free space.



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the procedure below.

1. Use the database reorganization utility (`pdrorg`) to unload the table data from the RDAREA (`R02`) whose storage conditions are to be changed, and save it in a format that can be used as the input to the database load utility (`pdload`).
2. Use `ALTER TABLE` to split the RDAREA. During this process, data in the split-target RDAREAs (`R02` and `RI02`) is deleted.
3. Perform data loading into the RDAREA (`R02`) using the addition mode of the database load utility (`pdload`). Perform this operation for each RDAREA using the unload data file created in step 1 as the input file.

   Because the unload data file contains data that does not match the split storage conditions, data error information is output. If you do not need the data error information, you can suppress its output by specifying

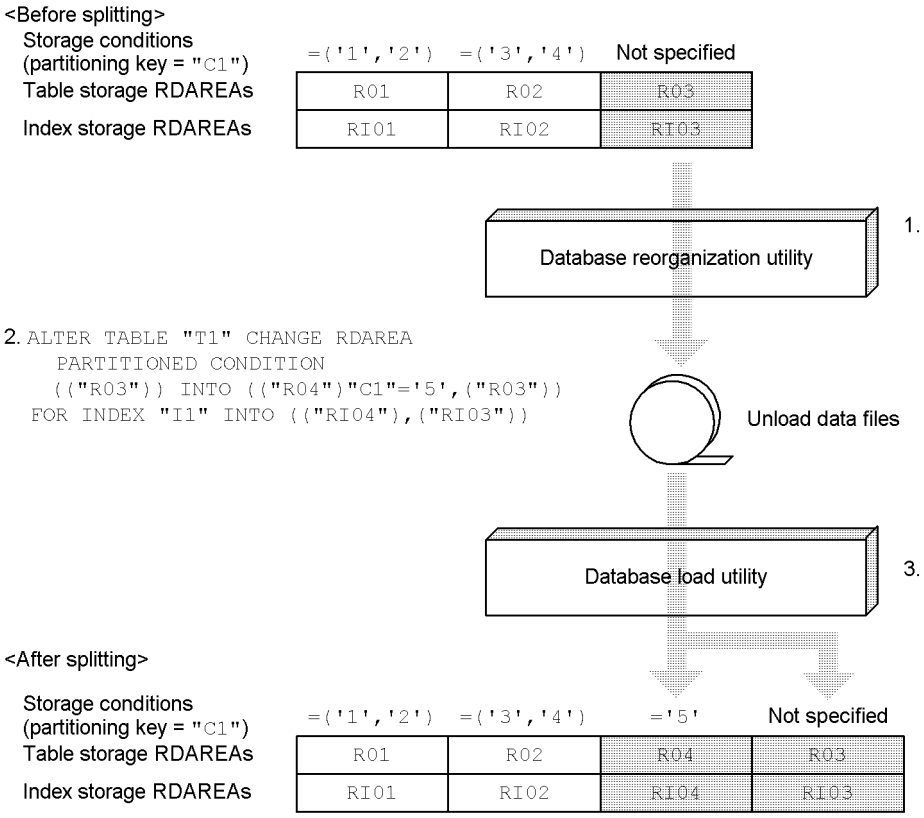
`divermsg=off` in the `option` statement.

4. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
5. Use `ALTER ROUTINE` to re-create the routines and triggers that were disabled in step 2.

(2) Example 2 (adding partitioning storage conditions)

This example adds a partitioning storage condition for a table that uses the branch code as the partitioning key (the partitioning storage condition will be branch code `'5'`).

- Provide new RDAREAs (`R04` and `RI04`) to store the data whose branch code is `'5'`.
- Extract the data whose branch code is `'5'` from the RDAREA with no storage condition specified (`R03`) and store it in the new RDAREA (`R04`).
- Use the pre-split RDAREA also after splitting.



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the procedure below.

1. Use the database reorganization utility (`pdrorg`) to unload the table data from the RDAREA (`R03`) whose storage conditions are to be changed and save it in a format that can be used as the input to the database load utility (`pdload`).
2. Use `ALTER TABLE` to split the RDAREA. During this process, data in the split-target RDAREAs (`R03` and `RI03`) is deleted.
3. Perform data loading into the RDAREAs (`R03` and `R04`) using the addition mode of the database load utility (`pdload`). Perform this operation for each RDAREA using the unload data file created in step 1 as the input file.

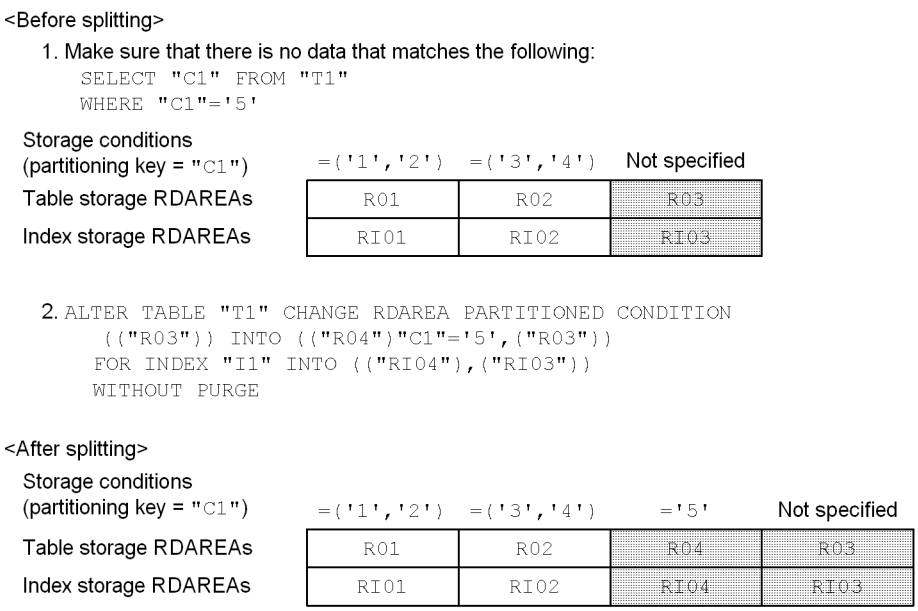
   Because the unload data file contains data that does not match the split storage conditions, data error information is output. If you do not need the data error information, you can suppress its output by specifying `divermsg=off` in the `option` statement.
4. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
5. Use `ALTER ROUTINE` to re-create the routines and triggers that were disabled in step 2.

(3) Example 3 (splitting an RDAREA without deleting its data and using the pre-split RDAREA after splitting)

This example changes the partitioning storage conditions for a table that uses the branch code as the partitioning key. It performs the splitting operation as follows:

- Splits the RDAREA without deleting its data.
- Provides new RDAREAs (`R04` and `RI04`) to store the data whose branch code is `'5'`.
- Use the pre-split RDAREA also after splitting.

This example assumes that RDAREA `R03` does not already contain data whose branch code is `'5'`.



Note: The numbers in the figure (1. and 2.) correspond to the numbers in the procedure below.

1. Make sure that the split-target table does not contain any data that matches the storage condition to be added (storage condition = `'5'`).

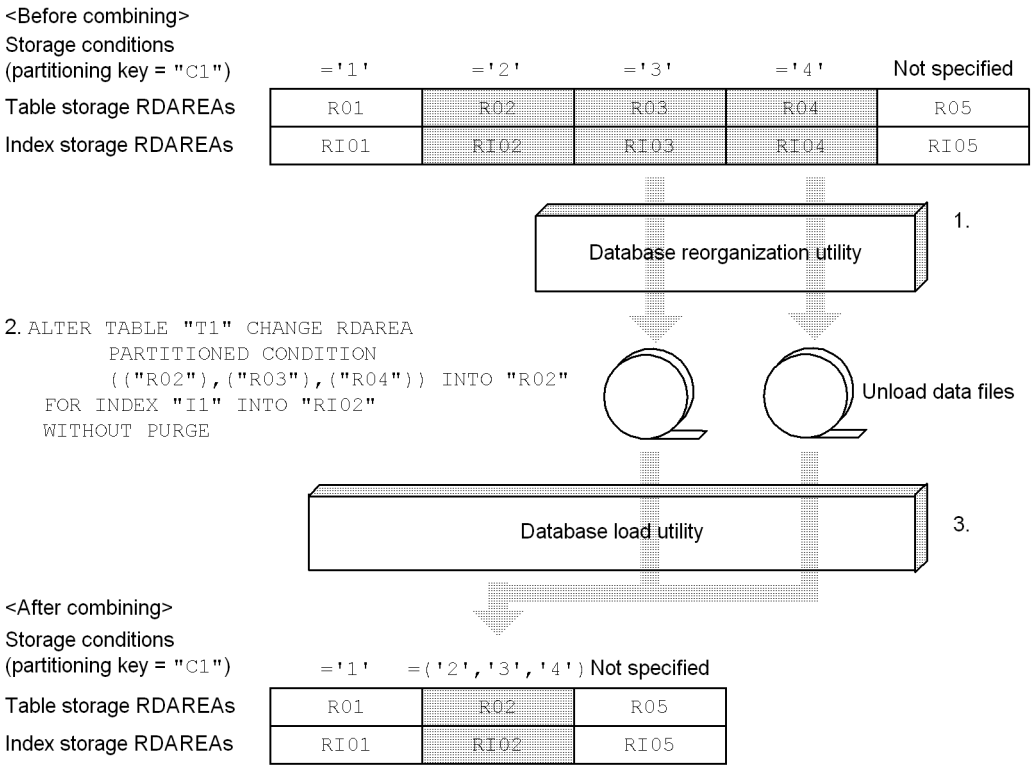
   If the table contains data whose storage condition = `'5'`, split it using the method described in *13.13.2(2) Example 2 (adding partitioning storage conditions)*. If you do not split the table with this method and the data that does not satisfy the storage condition remains after the splitting operation, you must recover the data using the method described in *13.13.9(2) Recovery procedure when data not satisfying the post-splitting storage condition remains*.

2. Use `ALTER TABLE` to split the RDAREA. In this case, specify `WITHOUT PURGE`.

3. Use `ALTER ROUTINE` to re-create the routines and triggers that were disabled in step 2.

(4) Example 4 (changing the partitioning storage conditions and then combining RDAREAs)

This example combines into `R02` the storage conditions that have been split into RDAREAs `R02`, `R03`, and `R04`.



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the procedure below.

1. Use the database reorganization utility (`pdrorg`) to unload data from the RDAREAs (`R03` and `R04`) in a format that can be used as the input to the database load utility (`pdload`).
2. Use `ALTER TABLE` to combine the RDAREAs. In this case, specify `WITHOUT PURGE` because `R02` will be used as is after the combine operation.
3. Perform data loading into RDAREA `R02` using the addition mode of the database load utility (`pdload`). Perform this operation for each RDAREA using the unload data file created in step 1 as the input file.

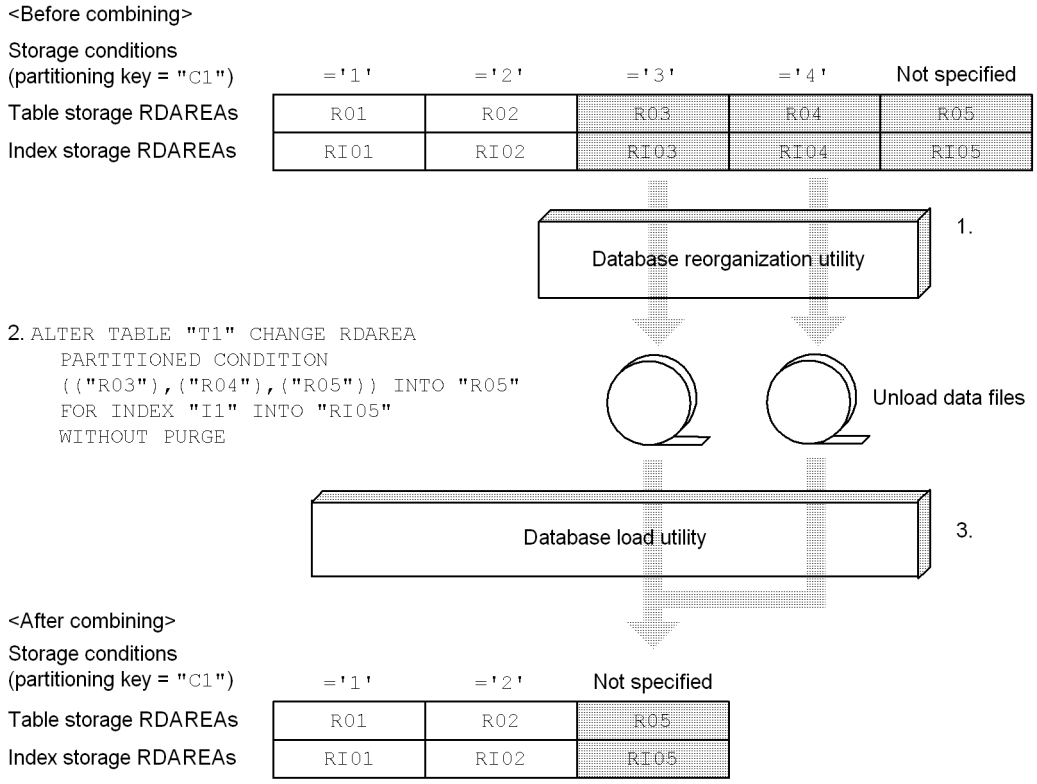
   Note that if data loading is performed in the creation mode, data is deleted from the pre-combination `R02`. If you have performed data loading in the creation mode by mistake, restore the data from a backup to its status immediately before execution of `ALTER TABLE`, and then perform the procedure again.
4. Check the number of items of data after the data has been loaded. For details,

see *13.13.8 Checking the number of items of data following splitting or combining*.

5. Use `ALTER ROUTINE` to re-create the routines and triggers that were disabled in step 2.

(5) Example 5 (deleting partitioning storage conditions)

This example combines into `R03` the storage conditions that have been split into RDAREAs `R03`, `R04`, and `R05`, and then deletes the storage conditions for `R03` and `R04`.



Note: The numbers in the figure (1. through 3.) correspond to the numbers in the procedure below.

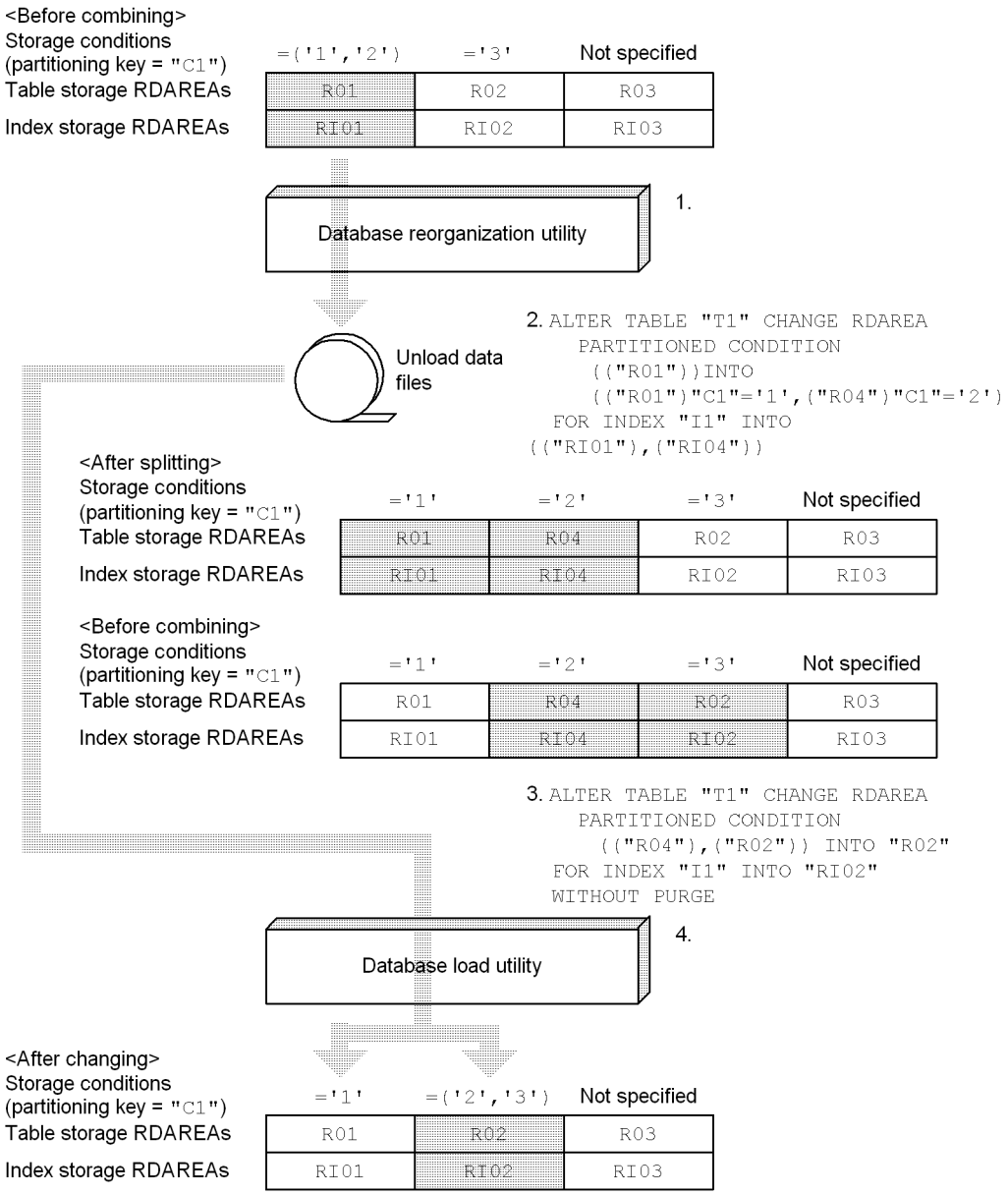
If you do not need the data in RDAREAs `R03` and `R04` whose storage conditions are to be deleted, you can skip steps 1 and 3 (unloading and data loading).

1. Use the database reorganization utility (`pdrorg`) to unload data from RDAREAs `R03` and `R04` in a format that can be used as the input to the database load utility (`pdload`).

2. Use `ALTER TABLE` to combine the RDAREAs.
3. Perform data loading into RDAREA `R05` using the addition mode of the database load utility (`pdload`). Perform this operation for each RDAREA using the unload data file created in step 1 as the input file.
4. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
5. Use `ALTER ROUTINE` to re-create the routines and triggers that were disabled in step 2.

(6) Example 6 (performing split and combine operations consecutively)

This example splits RDAREA `R01` into `R01` and `R04` and then combines RDAREAs `R02` and `R04` into `R02`.



Note: The numbers in the figure (1. through 4.) correspond to the numbers in the procedure below.

1. Use the database reorganization utility (`pdrorg`) to unload data from RDAREA `R01` in a format that can be used as the input to the database load

utility (pdload).

2. Use ALTER TABLE to split the RDAREA.
3. Use ALTER TABLE to combine the RDAREAs.
4. Perform data loading into RDAREAs R01 and R02 using the addition mode of the database load utility (pdload). Perform this operation for each RDAREA using the unload data file created in step 1 as the input file.

   Because the unload data file contains data that does not match the split storage conditions, data error information is output. If you do not need the data error information, you can suppress its output by specifying divermsg=off in the option statement.
5. Check the number of items of data after the data has been loaded. For details, see *13.13.8 Checking the number of items of data following splitting or combining*.
6. Use ALTER ROUTINE to re-create the routines and triggers that were disabled in steps 2 and 3.

Reference note:

Normally, it would be necessary to unload data from R02 and R04 before performing step 3. However, when two SQL statements are executed consecutively to perform split and combine operations, as in this example, unloading is not needed prior to step 3 because the unload data files created in step 1 can be used as input information for data loading in step 4.

13.13.3 Re-registering the data

It was explained in *13.13.1 Examples (in the case of boundary value specification)* and *13.13.2 Examples (in the case of storage condition specification)* that data that has been unloaded is loaded by executing the database load utility (pdload) for each RDAREA after a split operation and registration. This operation allows parallel execution of the database load utility (pdload) on multiple RDAREAs for data loading, thereby minimizing the impact of job stoppage. This operation also provides an advantage, in that the lock set during execution of the database load utility (pdload) is limited to the RDAREAs into which the data is to be loaded, and jobs that access other RDAREAs are not affected.

However, there are also disadvantages. Because the user must determine accurately the RDAREAs into which data is to be re-registered, the operation is prone to mistakes. Also, the database load utility (pdload) must be executed multiple times when it is necessary to register data into multiple RDAREAs.

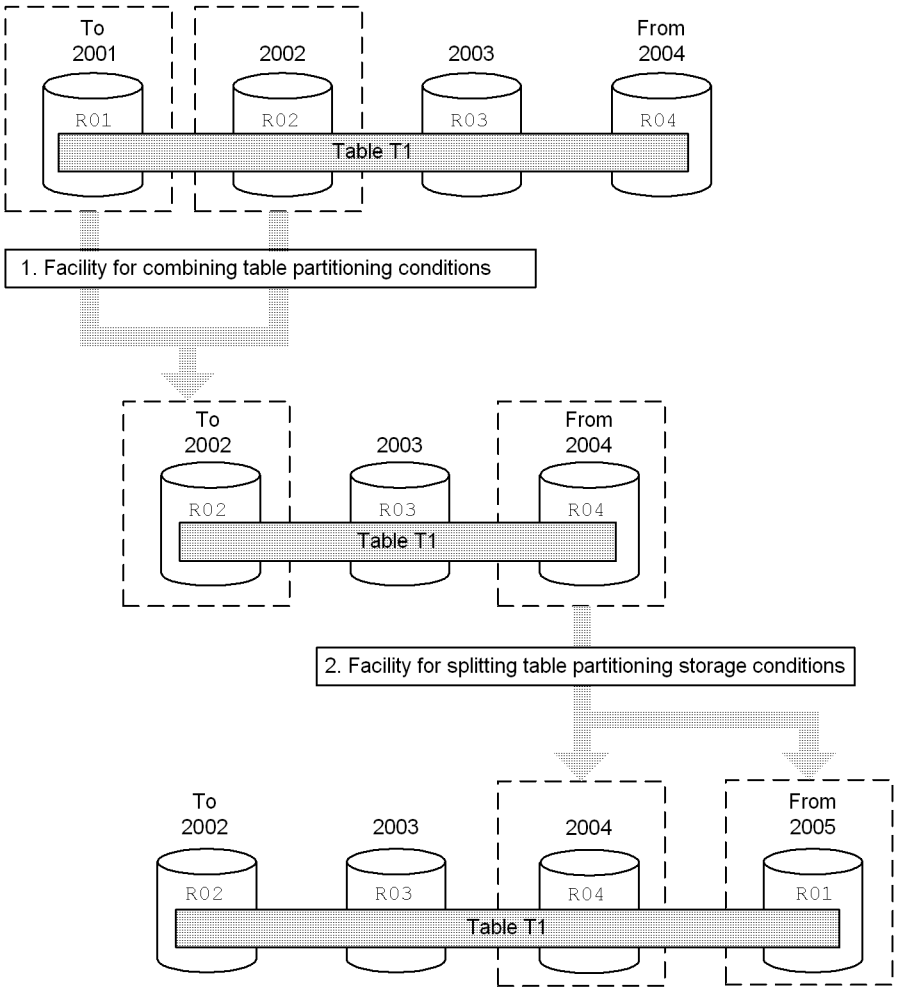
For this reason, if it is possible to allocate sufficient job stoppage time and to halt access to the tables that are to be split or combined for the duration of the stoppage,

Hitachi recommends that you load data with the database load utility (`pdload`) on a table-by-table basis, and not on an RDAREA-by-RDAREA basis.

13.13.4 Reusing RDAREAs

In the case of a database in which the volume of data increases over time, you can reuse RDAREAs by deleting data from RDAREAs storing old data and reusing those RDAREAs to store new data. Figure 13-77 shows an example of reusing an RDAREA.

Figure 13-77: Example of reusing an RDAREA



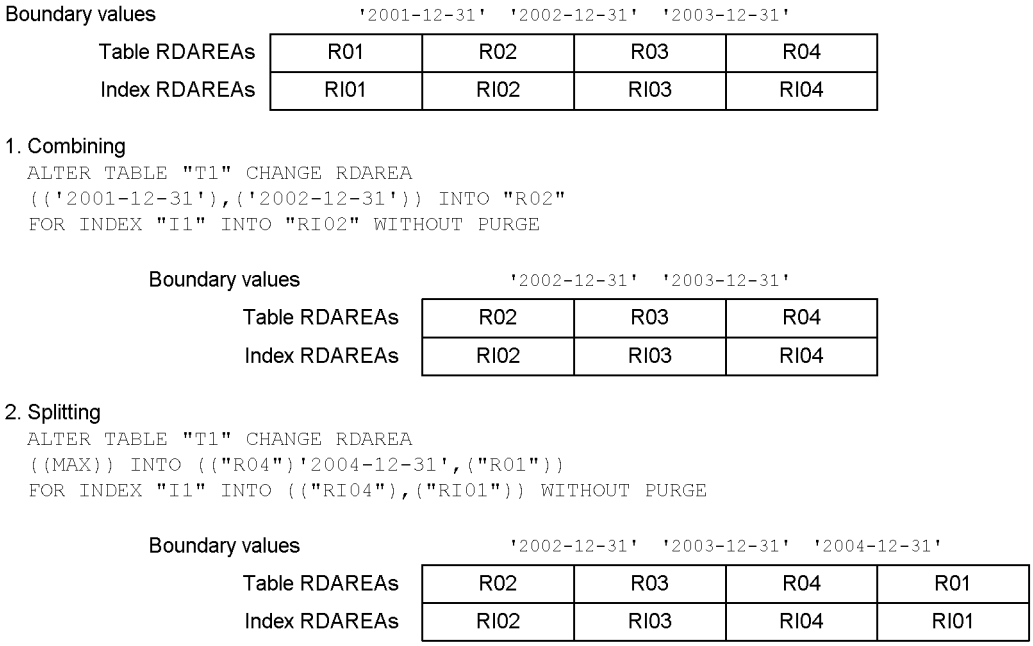
Note: The numbers in the figure (1. and 2.) correspond to the numbers in the procedure below.

Procedure

1. The combine facility for partitioning storage conditions is used to combine into RDAREA R02 the data for 2002 and earlier, which is stored in R01 and R02. During this process, the system deletes all data from R01.
2. The split facility for partitioning storage conditions can be used to split the data for 2004 and beyond, and to store the data for 2004 in R04 and the data for 2005 and beyond in R01; this means that R01, which had been used to store the data for 2001 and earlier, is reused.

Figure 13-78 shows the procedure for reusing RDAREAs.

Figure 13-78: Procedure for reusing RDAREAs



Note: The numbers in the figure (1. and 2.) correspond to the numbers in the procedure below.

Procedure

1. To separate R01 from the table, use ALTER TABLE to combine the minimum boundary value ('2001-12-31') into the boundary value ('2002-12-31') that is one increment greater than the minimum boundary value. For the post-combination RDAREAs, specify the RDAREAs (R02 and RI02) that will store the boundary value that is one increment greater than the minimum boundary value. Because the data in R02 must be retained, specify WITHOUT PURGE. As a result, R01 becomes separated from the table in terms of definition, and the data is deleted from the table in R01.

2. Use ALTER TABLE to split the RDAREA (R04) storing values greater than the maximum boundary value at a value ('2004-12-31') that is equal to or greater than the maximum value of the partitioning keys stored in that RDAREA. For the post-splitting RDAREAs, specify the RDAREAs (R04 and RI04) that used to store values greater than the maximum boundary value and the RDAREAs (R01 and RI01) that used to store values smaller than the minimum boundary value. Because the data in R04 must be retained, specify WITHOUT PURGE.

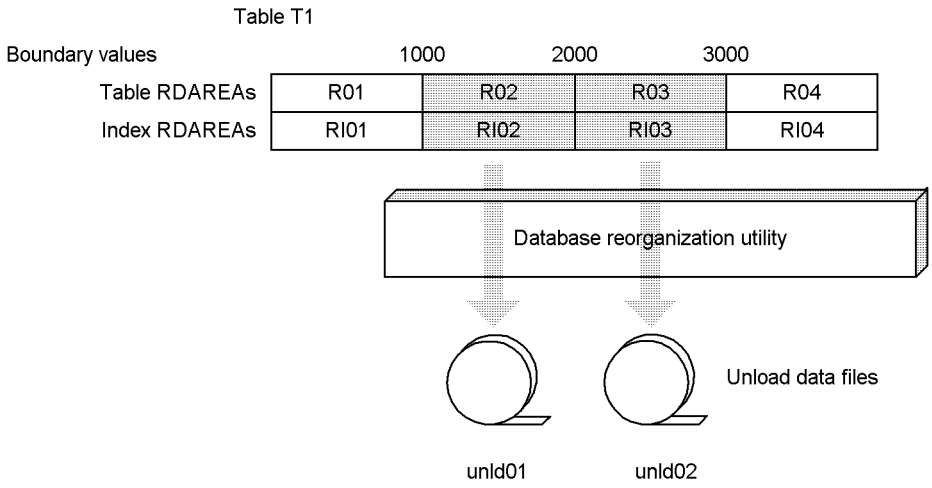
   Note that if there is data for '2005-1-1' and beyond in R04, there is a possibility that data not matching the storage conditions may remain in R04. For details, see *13.13.1(2) Example 2 (splitting an RDAREA without deleting its data and using the pre-split RDAREA after splitting)*.

13.13.5 Examples of the database reorganization utility and database load utility

(1) Example of the database reorganization utility (pdrorg)

The database reorganization utility (pdrorg) must unload data on an RDAREA-by-RDAREA basis and save it in a format that can be used as the input to the database load utility (pdload). Figure 13-79 shows an example of the database reorganization utility. In this example, the required unload data files are created before the table's partitioning storage conditions are changed.

Figure 13-79: Example of the database reorganization utility



(a) Unloading R02

The following is an example of the control statement for unloading R02, as well as the command line:

■ Control statement (control_file1)
```
unload /pdrorg/unld01
```

■ Command line
```
pdrorg -k unld -r R02 -W bin -j -t T1 control_file1
```

Explanation

-k: Specifies unld for unloading.

-r: Specifies the name of the RDAREA to be unloaded.

-W bin: Specifies unloading of data into a format that can be used by the database load utility. This option must be specified.

-j: Specifies that the table to be unloaded contains an abstract data type that has a BLOB column or LOB attribute.

-t: Specifies the table identifier of the table to be unloaded.

control_file1: Specifies the name the control statement file.

(b) Unloading R03

The following is an example of the control statement for unloading R03, as well as the command line:

■ Control statement (control_file2)
```
unload /pdrorg/unld02
```

■ Command line
```
pdrorg -k unld -r R03 -W bin -j -t T1 control_file2
```

Explanation

-k: Specifies unld for unloading.

-r: Specifies the name of the RDAREA to be unloaded.

-W bin: Specifies unloading of data into a format that can be used by the database load utility. This option must be specified.

-j: Specifies that the table to be unloaded contains an abstract data type that has a BLOB column or LOB attribute.

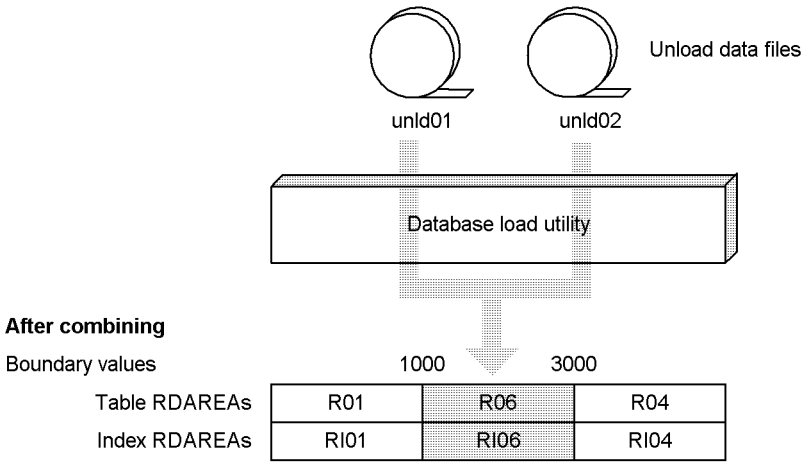
-t: Specifies the table identifier of the table to be unloaded.

control_file2: Specifies the name of the control statement file.

(2) Example of the database load utility (pdload)

The database load utility (pdload) must execute on the unload data files created by the database reorganization utility (pdrorg) on an RDAREA-by-RDAREA basis in the addition mode. Figure 13-80 shows an example of the database load utility. In this example, a database is created from a table whose partitioning storage conditions have been changed.

Figure 13-80: Example of the database load utility



(a) Combining R01 and R02 into R06

The following is an example of the control statement for combining R01 and R02 into R06, as well as the command line:

■ Control statement (control_file)

```
option divermsg=off
source R06 /pdrorg/unld01,/pdrorg/unld02
```

■ Command line

```
pdload -W -b T1 control_file
```

Explanation

-W: Specifies use of a file output by the database reorganization utility as the input file. This option must be specified.

`-b`: Specifies that the input file is binary. This must option be specified.

`T1`: Specifies the identifier of the target table.

`control_file`: Specifies the name of the control statement file.

13.13.6 Splitting or combining a table containing a non-partitioning key index

If a table contains a non-partitioning key index that does not correspond on a one-to-one basis to the table RDAREAs, that table cannot be split or combined.

Procedure

1. Delete the non-partitioning key index.
2. Unload the table data from the RDAREAs to be split or combined.
3. Use `ALTER TABLE` to specify boundary values or the RDAREA to be split.
4. Use the database load utility to load data from the unload data file created in step 2 into the split or combined RDAREAs.
5. Redefine the non-partitioning key index.

13.13.7 Splitting or combining when an index is incomplete

Once the database load utility has executed, you must not change partitioning storage conditions before indexes are re-created.

If you change partitioning storage conditions while indexes are incomplete, you must use the database reorganization utility to re-create the indexes.

13.13.8 Checking the number of items of data following splitting or combining

Before you can change partitioning storage conditions, you must unload data; after you have changed the partitioning storage conditions, you must reload the data. It is important that you check that the original number of items of data before the change matches the number of items of data after the data has been re-registered. The two methods described below are provided for this purpose. If you need a highly accurate check of the number of data items, Hitachi recommends that you use the second method.

- Using the number of data items output by the utility

  1. Record the number of items of data that were unloaded, which is indicated in the `KFPL00723-I` message that is output when the database reorganization utility has executed.
  2. After you change the partitioning storage conditions, record the number of items of data that are loaded, which is indicated in the `KFPL00723-I`

message that is output during data loading. If you load data into multiple RDAREAs, record the grand total.

3. Make sure that the number of data items in step 1 matches the number of data items in step 2.

- Executing the database condition analysis utility before and after a change in partitioning storage conditions

  You can check the number of items of data stored in RDAREAs by executing a condition analysis for each table with the database condition analysis utility (`pddbst`).

  1. Execute a condition analysis for each table with the database condition analysis utility (`pddbst`) and record the grand total of the values shown as the `Row Count` for each target RDAREA, as indicated in the example shown below.

```
pddbst -t table-name -s
```

The following are the execution results:

```
RD Area Name   : RD002
  Server        : bes1
  Status        :
  Original RD Area Name : RD002
  Generation Number :  0   Replica RD Area Count :  0
  History1 Hold Status :      Hold Code :     0  Hold Time :
  History2 Hold Status :      Hold Code :     0  Hold Time :
  Job Name      :         Line Count :               Index Method :
  Unused Segment:       3999
  Search Mode : INS   Segment Reuse :          - segments
  Reuse Search Failure :          0/         0
          Used(Full)       Used(      Full)        Sum
  Segment 100%(  0%)          1(         0)          1
  Page     30%(  0%)          3(         0)         10
  Row Count     :                  200

RD Area Name   : RD004
  Server        : bes2
  Status        :
  Original RD Area Name : RD004
  Generation Number :  0   Replica RD Area Count :  0
  History1 Hold Status :      Hold Code :     0  Hold Time :
  History2 Hold Status :      Hold Code :     0  Hold Time :
  Job Name      :         Line Count :               Index Method :
  Unused Segment:       3999
  Search Mode : INS   Segment Reuse :          - segments
  Reuse Search Failure :          0/         0
          Used(Full)       Used(      Full)        Sum
  Segment 100%(  0%)          1(         0)          1
  Page     20%(  0%)          2(         0)         10
  Row Count     :                  100
```

2. After you change the partitioning storage conditions and use the database load utility to load the data, use the same procedure as in step 1 and again record the grand total.
3. Check that the grand total obtained in step 1 matches the grand total obtained in step 2.

If the numbers of data items do not match, the following are possible causes and corrective measures:

Possible cause	Corrective measure
When the database load utility was executed to load data, the creation mode was used. You must use the addition mode.	You must use a backup to restore the system, then re-execute the utility.
Data loading was not executed for the RDAREA into which the data was supposed to be loaded.	Load data from the unload data file into the correct RDAREA.

13.13.9 Operation when an error occurs

(1) Rollback when an error occurs

If processing terminates abnormally during execution of `ALTER TABLE`, no recovery operation is required because the table changes are rolled back. You can simply re-execute `ALTER TABLE`.

However, once execution of `ALTER TABLE` has been completed, RDAREA data may have been deleted. Therefore, you cannot restore the system to its status before execution of `ALTER TABLE` if you did not make backup.

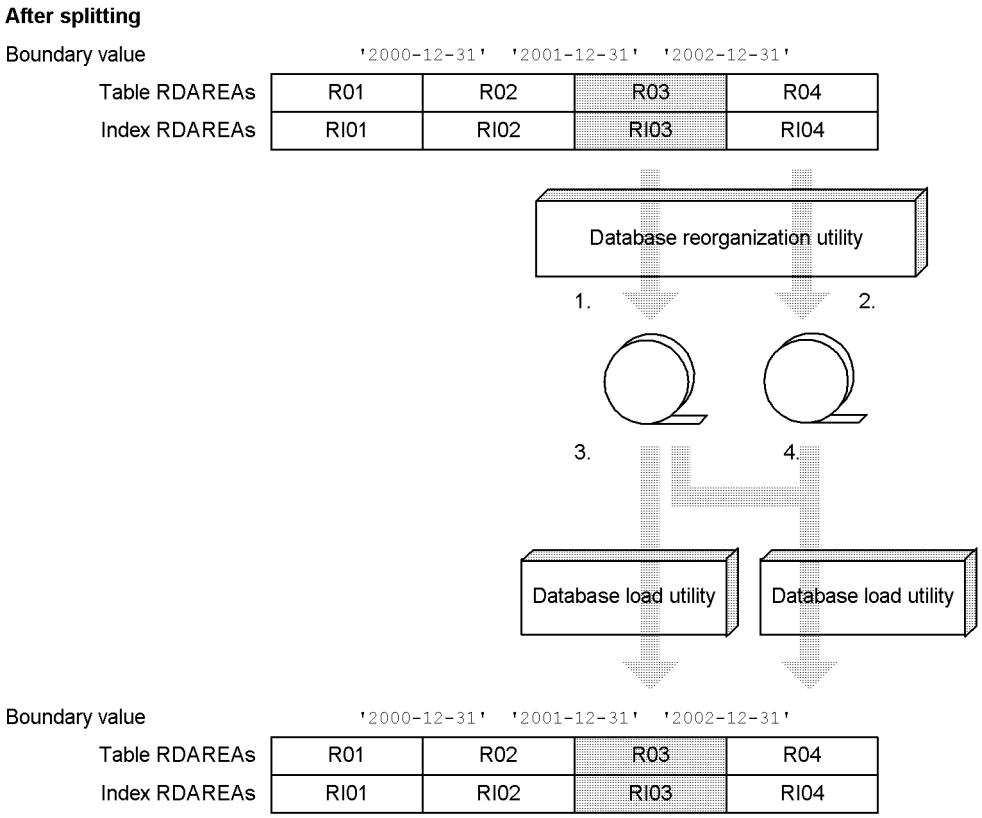
For this reason, it is important to make a backup in advance of performing any of these operations so that the system can be restored from the backup if necessary. You should make a backup of the following:

- Master directory
- Data directory
- Dictionary RDAREAs (including dictionary LOB RDAREAs)
- RDAREAs whose partitioning conditions are to be changed (including index RDAREAs and LOB RDAREAs). If the inner replica facility is used, you should back up all generations of RDAREAs whose partitioning conditions are to be changed.

(2) Recovery procedure when data not satisfying the post-splitting storage condition remains

If `WITHOUT PURGE` was wrongly specified, data may remain that does not match the storage conditions after splitting. In such a case, data retrieval operations on the basis of a partitioning key may not be possible and may terminate in an error. Figure 13-81 shows an example of the recovery procedure when data not satisfying the post-splitting storage condition remains. This example shows the recovery procedure when data that should have been stored in `R04` remains instead in `R03`.

Figure 13-81: Example of the recovery procedure when data not satisfying the post-splitting storage condition remains



Note: The numbers in the figure (1. through 4.) correspond to the numbers in the procedure below.

Procedure

1. Unload the table data from the RDAREA into which the incorrect data is stored, and save it in a format that can be used as the input to the database load utility.
2. Unload the table data from the RDAREA into which the data should have been stored, and save it in a format that can be used as the input to the database load utility.
3. Using the unload data file created in step 1 as the input file, execute the database load utility in the creation mode for the RDAREA that stores the wrong data on an RDAREA-by-RDAREA basis.

Because the unload data file contains data that does not match the split storage conditions, data error information is output. If you do not need the data error information, you can suppress its output by specifying `divermsg=off` in the `option` statement.

4. Using the unload data files created in steps 1 and 2 as the input files, execute the database load utility in the creation mode for the RDAREAs that should have stored the data on an RDAREA-by-RDAREA basis.

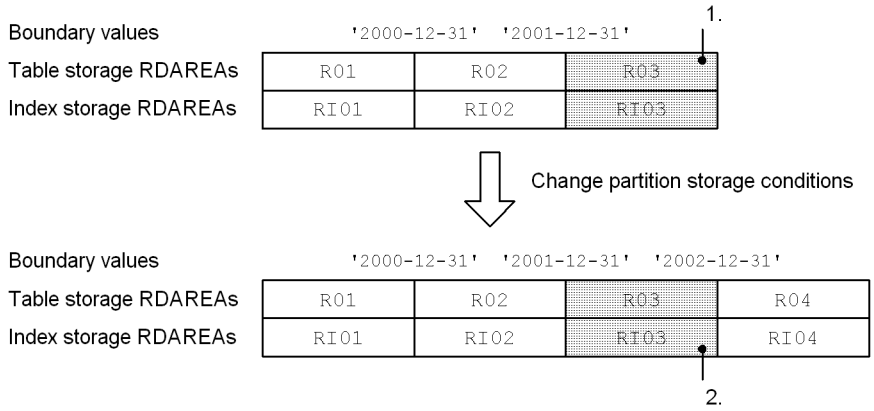
   Because the unload data files contain data that does not match the split storage conditions, data error information is output. If you do not need the data error information, you can suppress its output by specifying `divermsg=off` in the `option` statement.

13.13.10 Handling when referential constraint and check constraint are used

(1) Changing the partitioning storage conditions of a referenced table

When a partitioning storage condition is changed, table data may be deleted. Consequently, when the partitioning storage conditions of a referenced table are changed, matching between the referenced table and the referencing table cannot be guaranteed.

When `WITHOUT PURGE` can be used during splitting, table data will not be deleted. However, as explained in *13.13.9(2) Recovery procedure when data not satisfying the post-splitting storage condition remains*, if data remains that does not match the storage conditions after the partitioning storage conditions have been changed, that data cannot be retrieved. As a result, a mismatch occurs between this data and the data in the referencing table. The following shows an example.



Explanation:

1. In this example, R03 for the referenced table contains the data 2004-12-30 and this data also exists in the referencing table. Therefore, the referential constraint is satisfied.
2. If WITHOUT PURGE is specified, the data 2004-12-30 remains in R03 that is to store the data 2001-12-31 through 2002-12-31. In this case, the referenced table cannot be searched for the data 2004-12-30, and the referential constraint is not satisfied.

If pd_check_pending=USE is specified in the system definition and the partitioning storage condition is changed for a referenced table, the referencing table is placed in check pending status whether or not WITHOUT PURGE is specified. If pd_check_pending=NOUSE is specified, the referencing table is placed in non-check pending status. Therefore, if you have changed the partitioning storage conditions, you must check the table's integrity and release the table from check pending status. For details, see the manual *HiRDB Version 8 Installation and Design Guide*.

(2) Changing partitioning storage conditions for a referencing table

The following describes the operation when partitioning storage conditions are changed for a referencing table.

- When pd_check_pending=USE is specified in the system definition

  If you have changed the partitioning storage conditions for a referencing table, an RDAREA from which table data has been deleted is placed in non-check pending status. If all RDAREAs containing the subject table are placed in non-check pending status, the table is released from check pending status (the value of the CHECK_PEND columns in the SQL_TABLES and SQL_REFERENTIAL_CONSTRAINTS tables becomes the null value).

- When pd_check_pending=NOUSE is specified in the system definition

  If you have changed the partitioning storage conditions for a referencing table, an RDAREA whose table data has been deleted is placed in non-check pending status.

(3) Changing partitioning storage conditions for a table for which check constraint has been defined

The following describes the operation when partitioning storage conditions are changed for a table for which check constraint has been defined.

- When pd_check_pending=USE is specified in the system definition

  If you have changed the partitioning storage conditions for a table for which check constraint has been defined, an RDAREA from which table data has been deleted is placed in non-check pending status. If all RDAREAs containing the subject table are placed in non-check pending status, the table is released from

check pending status (the value of the CHECK_PEND2 columns in the SQL_TABLES and SQL_CHECKS tables becomes the null value).

- When pd_check_pending=NOUSE is specified in the system definition

  If you have changed the partitioning storage conditions for a table for which check constraint has been defined, an RDAREA whose table data has been deleted is placed in non-check pending status.

13.14 Changing the hash function

Executor: HiRDB administrator and table owner

When there are significant differences from one RDAREA to another in terms of the number of rows stored in the RDAREAs, processing efficiency for the table may degrade. In such a case, you can improve the table's processing efficiency by changing the hash function so as to equalize the numbers of rows stored in the RDAREAs.

Notes

1. Changing the hash function of a table invalidates any stored routines that use that table. If this happens, use the `ALTER PROCEDURE` or `ALTER ROUTINE` statement to re-create each stored routine.
2. Changing the hash function of a table specified in a trigger SQL statement invalidates the trigger. If this happens, use the `ALTER TRIGGER` or `ALTER ROUTINE` statement to re-create the trigger.
3. After the hash function has been changed, execute the optimizing information collection utility (`pdgetcst` command) if necessary. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.
4. The hash function can be changed for a table in which an abstract data type of a plug-in that has the constructor reverse generation function is defined. For example, the hash function of a table in which the following abstract data type is defined can be changed:

- `SGMLTEXT` (because the LOB attribute is present, the `-j` option must be specified)

13.14.1 Example 1: Flexible hash partitioning

This example changes to `HASH6` the hash function of table `TABLE01`. `TABLE01` is flexible hash partitioned.

(1) Use the ALTER TABLE statement to change the hash function

```
ALTER TABLE TABLE01 CHANGE HASH HASH6;
```

13.14.2 Example 2: FIX hash partitioning

This example changes to `HASH6` the hash function of table `TABLE01`. `TABLE01` is flexible hash partitioned.

(1) Use the pdhold command to shut down RDAREAs to be unloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -j -t TABLE01 -g /pdrorg/unld02
```

Explanation

`-k`: Specifies `unld` for unloading.

`-j`: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be unloaded.

`-t`: Specifies the name of the table that is to be unloaded.

`-g`: Specifies that `TABLE01` is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the `-g` option consolidates the unload data files (into a single file).

`/pdrorg/unld02`: Specifies the name of the control statements file for the `pdrorg` command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

(4) Use the PURGE TABLE statement to delete TABLE01's data

```
PURGE TABLE TABLE01;
```

When the hash function of a table with FIX hash partitioning is changed, the table data must be deleted.

(5) Use the ALTER TABLE statement to change the hash function

```
ALTER TABLE TABLE01 CHANGE HASH HASH6;
```

(6) Use the pdhold command to shut down RDAREAs to be reloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(7) Use the pdrorg command to reload data into TABLE01

```
pdrorg -k reld -j -t TABLE01 -g /pdrorg/reld02
```

Explanation

`-k`: Specifies `reld` for reloading.

`-j`: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be reloaded.

`-t`: Specifies the name of the table that is to be reloaded.

`-g`: If the `-g` option was specified in step (2), specify it here as well.

`/pdrorg/reld02`: Specifies the name of the control statements file for the `pdrorg` command.

(8) Back up the RDAREAs in which data was reloaded

Because you reloaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was reloaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(9) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.15 Changing a table's partitioning definition

Executor: HiRDB administrator and table owner (or user with DBA privilege)

This section explains how to change the partitioning definition of a table. The following partitioning definition changes are explained:

- Changing from key range partitioning to hash partitioning
- Changing the partitioning key column
- Changing from hash partitioning to key range partitioning
- Allocating a different RDAREA each month

Reference note:

After the table partitioning method has been changed, execute the optimizing information collection utility (`pdgetcst` command) if necessary. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.

13.15.1 Example 1 (changing from key range partitioning to hash partitioning and changing the partitioning key column)

- Changing the row partitioning method of a table (`TABLE01`) from key range partitioning to hash partitioning
- Changing the partitioning key column of a table (`TABLE01`)

(1) Use the pdhold command to shut down RDAREAs to be unloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -W bin -j -t TABLE01 -g /pdrorg/unld01
```

Explanation

`-k`: Specifies `unld` for unloading.

`-W bin`: Specifies that the unload data file is to be used as the input file (binary format) for the `pdload` command.

`-j`: Specifies that a LOB column or a column with the LOB attribute is defined

in the table that is to be unloaded.

`-t`: Specifies the name of the table that is to be unloaded.

`-g`: Specifies that `TABLE01` is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the `-g` option consolidates the unload data files (into a single file).

`/pdrorg/unld01`: Specifies the name of the control information file for the `pdrorg` command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

(4) Use the DROP TABLE statement to delete TABLE01

```
DROP TABLE TABLE01;
```

(5) Use the CREATE TABLE statement to redefine the partitioning method for TABLE01

```
CREATE TABLE TABLE01 ... ;
```

(6) Use the CREATE INDEX statement to redefine the index

```
CREATE INDEX INDX01 ... ;
```

(7) Use the pdhold command to shut down RDAREAs into which data is to be loaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(8) Use the pdload command to load data into TABLE01

```
pdload -b -W TABLE01 /pdload/load01
```

Explanation

-b: Specifies that data in binary format is to be loaded.

-W: Specifies that an input data file in binary format, created with the pdrorg command, is to be used.

TABLE01: Specifies the name of the table into which the data is to be loaded.

/pdload/load01: Specifies the name of the control information file for the pdload command.

(9) Back up the RDAREAs in which data was loaded

Because you loaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was loaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(10) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.15.2 Example 2 (changing from hash partitioning to key range partitioning)

The procedure described below can be used to change the row partitioning method from key range partitioning to hash partitioning. This procedure can be used only when no change is made to the structure of the columns containing the partitioning key.

(1) Use the pdhold command to shut down RDAREAs to be unloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -j -t TABLE01 -g /pdrorg/unld02
```

Explanation

-k: Specifies unld for unloading.

-j: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be unloaded.

-t: Specifies the name of the table that is to be unloaded.

-g: Specifies that TABLE01 is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the -g option consolidates the unload data files (into a single file).

/pdrorg/unld02: Specifies the name of the control information file for the pdrorg command.

(3) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

(4) Use the DROP TABLE statement to delete TABLE01

```
DROP TABLE TABLE01;
```

(5) Use the CREATE TABLE statement to redefine the partitioning method for TABLE01

```
CREATE TABLE TABLE01 ... ;
```

(6) Use the CREATE INDEX statement to redefine the index

```
CREATE INDEX INDX01 ... ;
```

(7) Use the pdhold command to shut down RDAREAs to be reloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(8) Use the pdrorg command to reload data into TABLE01

```
pdrorg -k reld -j -t TABLE01 -g /pdrorg/reld02
```

Explanation

-k: Specifies reld for reloading.

-j: Specifies that a LOB column or a column with the LOB attribute is defined

in the table that is to be reloaded.

`-t`: Specifies the name of the table that is to be reloaded.

`-g`: If the `-g` option was specified in step (2), specify it here as well.

`/pdrorg/reld02`: Specifies the name of the control information file for the `pdrorg` command.

(9) Back up the RDAREAs in which data was reloaded

Because you reloaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was reloaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(10) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

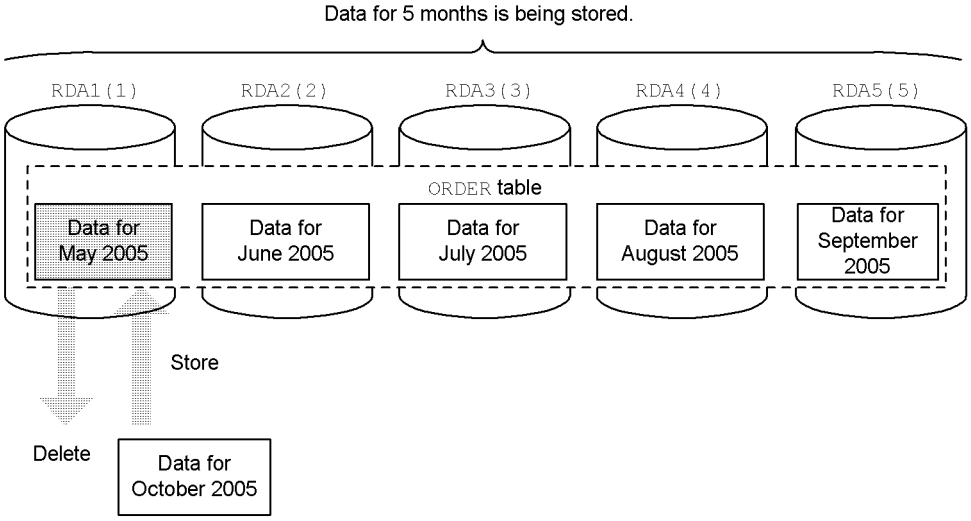
It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

13.15.3 Example 3 (allocating a different RDAREA each month)

This example assumes that data for five recent months (May through September 2005) is stored. The example uses the hash function `HASH0` to delete the data for the oldest month, May 2005, from the RDAREA and then stores the data for the most recent month, October 2005.

Figure 13-82 provides an overview of Example 3 (method for allocating a different RDAREA each month).

Figure 13-82: Overview of Example 3 (method for allocating a different RDAREA each month)



Notes:
1. (*m*) in the RDAREA names indicates the specification order in the allocation conditions.
2. The ORDER table has been defined as follows:

```
CREATE TABLE ORDER
  (DNO CHAR(6), CCODE CHAR(5), PCODE CHAR(4), ODATE CHAR(8) NOT NULL)
  FIX HASH HASH0 BY ODATE IN (RDA1,RDA2,RDA3,RDA4,RDA5)
```

(1) Identify the RDAREA that contains the data for May 2005, which is to be deleted

(a) Execute the hash function for table partitioning

Use the HiRDB-provided hash function for table partitioning to identify the RDAREA that contains the data for May 2005, which is to be deleted. The table below shows the arguments of the hash function for table partitioning. For details about the hash function for table partitioning, see the manual *HiRDB Version 8 UAP Development Guide*.

Argument	Specification value
`hashcode` (hash function code)	`p_rdb_HASH0`
`ncol` (number of partitioning key columns)	`1`
`collst` (partitioning key specification order and data type code, and data length code)	Data type code: `PDSQL_CHAR` Data length code: `8`
`dadlst` (data stored in the partitioning key)	Year, month, date

Argument	Specification value
ndiv (number of table partitions)	5
ncspace (national character for the double-byte space used at the HiRDB servers)	ncspace[0]: 0x81 ncspace[1]: 0x40
flags (space conversion level and whether or not the facility for conversion to a DECIMAL signed normalized number is to be used)	0
rdno (partitioning condition specification order or serial number in the partitioning key)	None

Note:

This example assumes that the character encoding type, space conversion level, and the facility for conversion to a DECIMAL signed normalized number are all omitted.

When the above hash function for table partitioning is executed, rdno=1 is returned as the partitioning condition specification order.

(b) Execute the SQL statement

Based on the result of the hash function for table partitioning, execute the following SQL statement to find the name of the RDAREA:

```
SELECT RDAREA_NAME
    FROM MASTER.SQL_DIV_TABLE
      WHERE TABLE_SCHEMA='USER1' /* user name       */
        AND TABLE_NAME='ORDER'   /* name of hash-partitioned table */
       AND DIV_NO=1              /* partitioning condition specification order */
```

When this SQL statement is executed, RDA1 is returned as the name of the RDAREA to be deleted.

(2) Use the pdhold command to shut down the RDAREA to be unloaded

```
pdhold -r RDA1
```

(3) Use the pdrorg command to unload data from RDA1 for the ORDER table

```
pdrorg -k unld -t ORDER -r RDA1 /pdrorg/unld03
```

Explanation

-k: Specifies unld for unloading.

-t: Specifies the name of the table to be unloaded.

-r: Specifies the name of the target RDAREA in order to unload only the specified RDAREA.

/pdrorg/unld03: Specifies the name of the control information file for the pdrorg command. The following shows the contents of the control information file:

```
unload bes1:/pdrorg/unload_file

/* bes1: Name of server that contains unload data file */
/* /pdrorg/unload_file: Name of unload data file */
```

(4) Use the pdload command to perform data loading on RDA1 with a data count of 0

To delete the data for May 2005 (the data in RDA1), which is the oldest data, execute data loading with a data count of 0. If a non-partitioning key index has been defined for the partitioned table, you must create the non-partitioning key index in the batch mode after executing pdload.

```
pdload -d ORDER /pdload/load03
```

Explanation

-d: Specifies that the existing data is to be deleted and then data loading is to be performed.

ORDER: Specifies the name of the data on which data loading is to be performed.

/pdload/load03: Specifies the name of the control information file for the pdload command. The following shows the contents of the control information file:

```
source RDA1 /pdload/load_file

/* RDA1: Name of RDAREA on which data loading is to be performed */
/* /pdload/load_file: Name of input data file */
```

(5) Use the pdrels command to release the RDAREA from shutdown status

```
pdrels -r RDA1
```

After executing the command, you should check the execution results for errors. For details about how to check the command's execution results, see the manual *HiRDB Version 8 Command Reference*.

After the command has executed, you can store data in `RDA1` by using a program such as a UAP to insert the data for October 2005.

13.16 Migrating data to another table

Executor: HiRDB administrator, DBA privilege holder, and table owner

Data can be moved to another table by executing the `pdrorg` or `pdload` command. The migration method depends on whether or not the migration source table and the migration target table have identical definitions. The table definitions are considered to be identical when all the following are true:

- Both are either FIX or non-FIX tables.
- They have the same number of columns.
- Their column definitions are the same (same column names, data types, `NULL` or `NOT NULL`, data sizes, sequence of column definitions, repetitions counts).

Figure 13-83 shows how to migrate data to a table with the same table definition as the source table. Figure 13-84 shows how to migrate data to a table with a different table definition.

Figure 13-83: Migrating data to a table with the same table definition

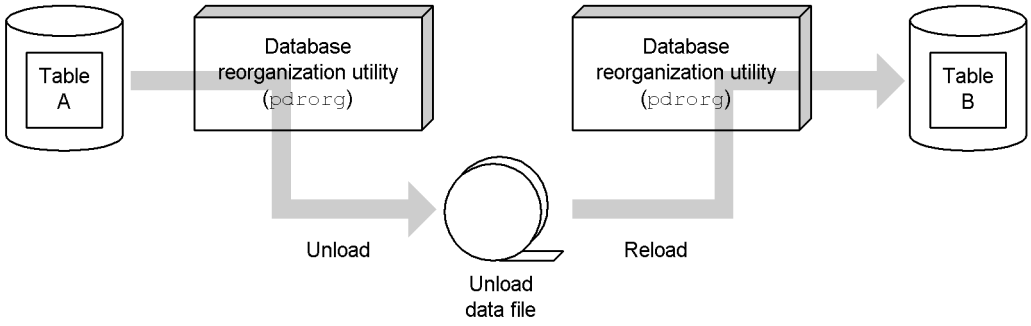
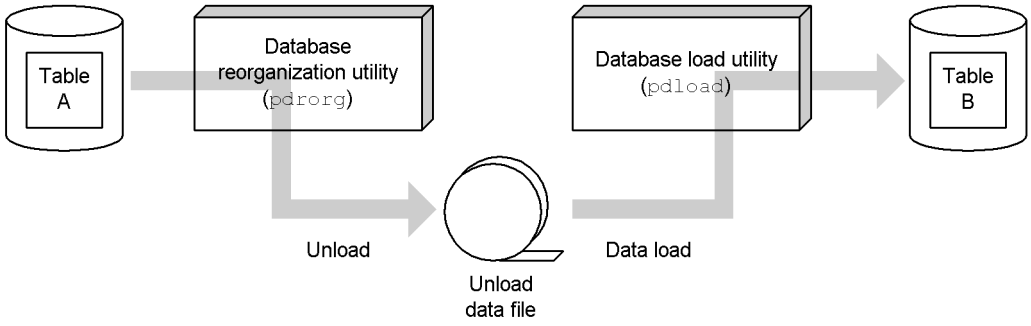


Figure 13-84: Migrating data to a table with a different table definition



Important

- After data migration, execute the optimizing information collection utility (`pdgetcst` command) for the migration destination table if necessary. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.
- If you wish to migrate to another HiRDB system a table that has columns of the abstract data type, you must apply the constructor parameter reverse generation function when you use the database reorganization utility (`pdrorg` command) to unload data of the abstract data type.

13.16.1 Example 1: Migrating data to a table with the same table definition

`TABLE1` and `TABLE2` have the same table definition. This example moves (copies) the data from `TABLE1` to `TABLE2`.

(1) Use the pdhold command to shut down RDAREAs to be unloaded and RDAREAs to be reloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -j -t TABLE01 -g /pdrorg/unld01
```

Explanation

`-k`: Specifies `unld` for unloading.

-j: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be unloaded.

-t: Specifies the name of the table that is to be unloaded.

-g: Specifies that TABLE01 is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the -g option consolidates the unload data files (into a single file).

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command.

(3) Use the pdrorg command to load data into TABLE02

```
pdrorg -k reld -j -t TABLE02 -g /pdrorg/reld01
```

Explanation

-k: Specifies reld for reloading.

-j: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be reloaded.

-t: Specifies the name of the table that is to be reloaded.

-g: If the -g option was specified in step (2), specify it here as well.

/pdrorg/reld01: Specifies the name of the control statements file for the pdrorg command. Because the data will be reloaded into another table, specify a tblname statement. The following is an example of specifying a tblname statement:

```
tblname TABLE01
```

(4) Back up the RDAREAs in which data was reloaded

Because you reloaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was reloaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

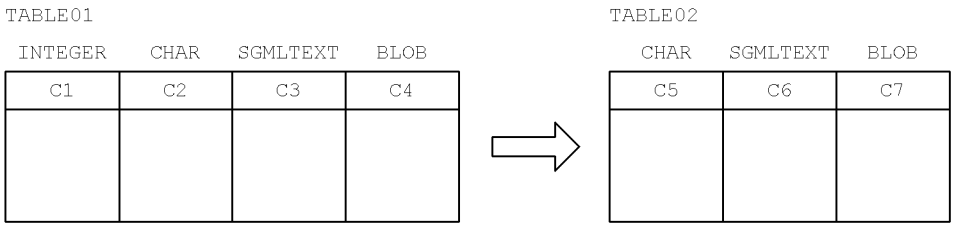
(5) Use the pdrels command to release RDAREAs from shutdown status

```
pdrels -r RDAREA01,RDAREA02,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.16.2 Example 2: Migrating data to a table with a different table definition

This example moves (copies) the data from TABLE1 to TABLE2. The table definitions of TABLE1 and TABLE2 differ as follows:



(1) Use the pdhold command to shut down RDAREAs to be unloaded and RDAREAs to be reloaded

```
pdhold -r RDAREA01,RDAREA02,...
```

(2) Create the control statements file for the pdrorg command

The contents of the control statements file (/pdrorg/rorg01) are shown below.

(a) HiRDB/Single Server

```
unload /pdrorg/unfile1                                  1
unld_func type=sgmltext,func=unsgmltext(sgmltext)       2
```

Explanation

1. Specifies the name of the unload data file.
2. Specifies information on the constructor parameter reverse generation function, because the SGMLTEXT type is defined for the table.

(b) HiRDB/Parallel Server

```
unload bes1:/pdrorg/unfile1                             1
unld_func type=sgmltext,func=unsgmltext(sgmltext)       2
```

Explanation

1. Specifies the name of the unload data file.

2. Specifies information on the constructor parameter reverse generation function, because the SGMLTEXT type is defined for the table.

(3) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -W bin -j -t TABLE01 -g /pdrorg/rorg01
```

Explanation

-k: Specifies unld for unloading.

-W bin: Specifies use of the unload data file as the input file (binary format) for the pdload command.

-j: Specifies that a LOB column or a column with the LOB attribute is defined in the table that is to be unloaded.

-t: Specifies the name of the table that is to be unloaded.

-g: Specifies that TABLE01 is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the -g option consolidates the unload data files (into a single file).

/pdrorg/rorg01: Specifies the name of the control statements file for the pdrorg command.

(4) Create the column structure information file (/pdload/column01)

```
*skipdata*,type=integer
C5
C6,func=(sgmltext,param=blob)
C7
```

Because the column structure of the input data file created in step (3) differs from the column structure of TABLE02, a column structure information file is needed for data loading. For an example of specifying a column structure information file, see *13.16.3 Specification examples of column structure information files*.

(5) Create the control statements file for the pdload command

The contents of the control statements file (/pdload/load01) are shown below.

(a) HiRDB/Single Server

```
source /pdrorg/unfile1
```

Explanation

Specifies the name of the unload data file created in step (3).

(b) HiRDB/Parallel Server

```
source bes1:/pdrorg/unfile1
```

Explanation

Specifies the name of the unload data file created in step (3).

(6) Use the pdload command to load data into TABLE02

```
pdload -k d -b -W -c /pdload/column01 TABLE02 /pdload/load01
```

Explanation

`-k d`: Specifies that the input data contains `BLOB` data.

`-b`: Specifies that data in binary format is to be loaded.

`-W`: Specifies that an input data file in binary format, created with the `pdrorg` command, is to be used.

`-c /pdload/column01`: Specifies the name of the column structure information file created in step (4).

`TABLE02`: Specifies the name of the table into which the data is to be loaded.

`/pdload/load01`: Specifies the name of the control statements file for the `pdload` command.

(7) Back up the RDAREAs in which data was loaded

Because you loaded data in the pre-update acquisition mode (default), back up the RDAREAs in which data was loaded. For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(8) Use the pdrels command to release RDAREAs from shutdown status

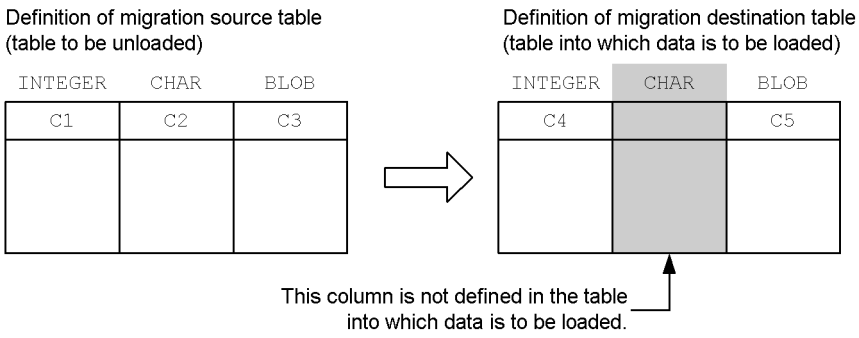
```
pdrels -r RDAREA01,RDAREA02,...
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

13.16.3 Specification examples of column structure information files

This section provides specification examples of column structure information files.

(1) Example 1: When a column is deleted



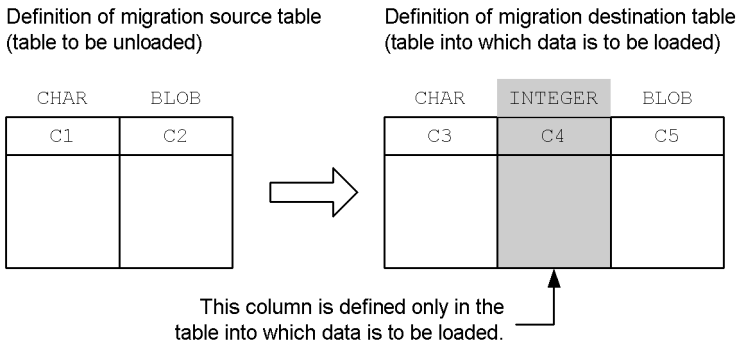
Specification example of a column structure information file

```
C4
*skipdata*,type=char(4)
C5
```

Explanation

The `skipdata` statement is specified because the migration destination table does not have the `CHAR`-type column.

(2) Example 2: When a column is added



Specification example of a column structure information file

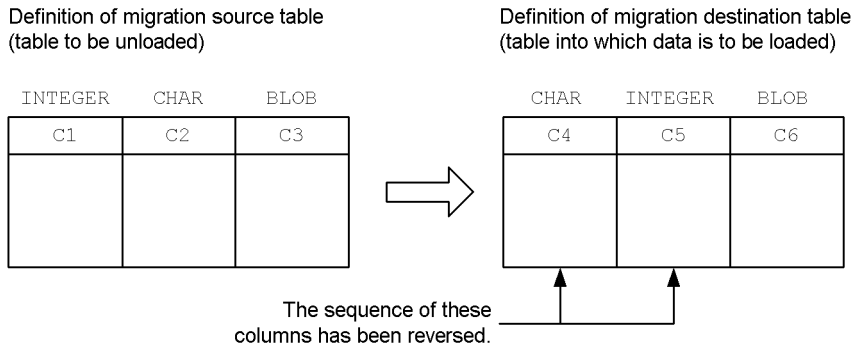
```
C3
C5
```

Explanation

Column C4 has been added to the migration destination table. The added column C4 is not described in the column structure information file.

When a column has been added, it is important that it not be specified in the column structure information file. That way, HiRDB detects that the input data does not contain data for that column, and it stores the default or NULL value in the added column. In the case of a table with the FIX attribute, the NULL value cannot be stored, so HiRDB stores the default value if the WITH DEFAULT operand was specified in the ALTER TABLE statement when the column was added.

(3) Example 3: When the column sequence has been changed



Specification example of a column structure information file

```
C5
C4
C6
```

Explanation

Because columns C4 and C5 have been reversed, C5 is specified before C4.

13.17 Deleting a table

Executor: Table owner or DBA privilege holder

`DROP TABLE` is used to delete an unneeded table. Because the data stored in the table is deleted, free space is created in the user RDAREAs, enabling the user RDAREAs to be used more efficiently.

Notes

1. When a table is deleted, the following resources are also deleted:
   - Indexes for the table
   - View tables created from the table
   - Access privileges to the table
   - Triggers defined for the table
2. Deleting a table invalidates any stored routines that use that table.
3. Deleting a table specified in a trigger SQL statement invalidates the trigger.
4. Falsification prevented tables containing data cannot be deleted.

13.18 Deleting a schema

Executor: Schema owner or DBA privilege holder

`DROP SCHEMA` is used to delete a schema.

(1) Resources that are deleted when a schema is deleted

When a schema is deleted, the schema owner's resources are also deleted. This means that the following resources are deleted:

- Tables
- Indexes
- View tables
- Comments
- Access privileges
- Abstract data types
- Index type
- Stored procedures
- Stored functions
- Trigger
- Foreign table
- Foreign index

(2) Other users' resources that are affected

View tables are deleted

Because the schema owner's tables and view tables are deleted, all view tables created from any of those tables or view tables are also deleted.

Access privileges are revoked

Because the schema owner's tables and view tables are deleted, all access privileges for those tables and view tables are revoked.

Stored routines and triggers become invalid

Because the schema owner's tables, view tables, indexes, abstract data types, stored routines, and triggers are deleted, stored routines and triggers that use any of these resources become invalid.

(3) When a schema cannot be deleted

- Your schema cannot be deleted if a user is accessing an abstract data type that is defined in a table you own.
- The schema of a user who owns a falsification prevented table that contains data cannot be deleted.

13.19 Deleting an abstract data type

Executor: Table owner or DBA privilege holder

`DROP DATA TYPE` is used to delete an abstract data type.

Note:

An abstract data type cannot be deleted while it is being used by any resource (table, index, or another abstract data type).

(1) When an abstract data type cannot be deleted

If a data type or its parent data type to be deleted satisfies any of the conditions listed below, `WITH PROGRAM` must be specified in `DROP DATA TYPE`; otherwise, the deletion processing will fail:

- The data type is specified in an SQL parameter in a procedure or function.
- The data type is specified for a function's return value.
- The data type is specified in an argument or return value of a function that is called from a procedure or another function.
- The data type is specified in a procedure, function, or trigger (if an abstract data type is being accessed by a component specification, this includes all data types used to build the abstract data type).

Deleting an abstract data type invalidates any stored routine or trigger that satisfies the above conditions. If this happens, use the `ALTER PROCEDURE`, `ALTER ROUTINE`, or `ALTER TRIGGER` statement to re-create the stored routines and stored triggers.

(2) Subtypes cannot be deleted

- A data type cannot be deleted if its parent data type is used in a table definition.
- A data type also cannot be deleted if the data type itself or its parent data type is used in another abstract data type.

13.20 Creating a definition SQL from an existing table

Executor: HiRDB administrator or DBA privilege holder

The `pddefrev` command can be used to create a definition SQL from an existing table.

This command is useful for creating a table that uses an existing table's definition. A definition SQL created with the `pddefrev` command is used as input information to the database definition utility (`pddef` command).

Reference note:

A table for which an abstract data type is defined cannot be processed by the `pddefrev` command (a definition SQL cannot be created from it).

13.21 Managing a list (narrowed search)

Executor: HiRDB administrator and table owner (or user with DBA privilege)

This section explains how to manage a list to be used for narrowed search. The following points should be noted when a list is used.

(1) All lists are deleted when HiRDB terminates

When HiRDB terminates (including abnormal termination), all lists are deleted, rendering a search using one of the lists impossible. If a list is needed, it must be re-created with the `ASSIGN LIST` statement.

HiRDB/Parallel Server

1. When a unit terminates, all lists in that unit are deleted.
2. When a server terminates, all lists in that server are deleted.

(2) When the base table of a list is recovered with the database recovery utility

When the base table of a list is recovered to the most recent synchronization point, that list can be used as is. However, if the base table is not recovered to the most recent synchronization point (e.g., it is recovered to the status when a backup was made), the base table may no longer match the list, and the list must be re-created with the `ASSIGN LIST` statement.

(3) When an RDAREA storing the base table of a list is re-initialized

When an RDAREA storing the base table of a list is re-initialized, you must do one of the following:

- Re-create the list with the `ASSIGN LIST` statement.
- Delete the list with the `DROP LIST` statement.

(4) Operations that invalidate a list

When any of the following processing is performed on the base table of a list, the retrieval results for the list become invalid, and the list must be re-created with the `ASSIGN LIST` statement:

- Table reorganization
- Data loading into the table in the creation mode
- Execution of the `PURGE TABLE` statement

(5) Some commands cannot be used for a list or list RDAREA.

The following commands cannot be used for a list or list RDAREA:

- Database load utility

- Database reorganization utility
- Database copy utility
- Database recovery utility
- Database condition analysis utility (only physical analysis of an RDAREA is possible)

(6) Notes on termination of the dictionary server

To reduce overhead during list creation and deletion, HiRDB maintains list management information in the memory of a dictionary server. Consequently, list management information is lost when the dictionary server or the unit in which the dictionary server is located is terminated. When this occurs, all lists that have been created become invalid and must be re-created with the `ASSIGN LIST` statement.

Once the dictionary server is restarted and until the transactions that use lists of other users that were being executed before the dictionary server was terminated have been completely recovered, processes that use the lists may cause the `KFPA11998-E` error message, `list operation not completed`, to be output.

(7) Command for checking list information (pdlistls command)

List information can be checked by executing the `pdlistls` command. The following information is displayed:

- List name
- List owner
- Name of the list's base table
- Owner of the list's base table
- `pd_max_list_users` value (maximum number of list owners)
- `pd_max_list_count` value (maximum number of lists that can be created by one user)

(8) Notes on using the inner replica facility

Exercise caution if the RDAREA being accessed is to be changed when a replica RDAREA has been defined for an RDAREA that contains a base table of a list. If the RDAREA being accessed is changed with the current database switch command (`pddbch` command) or with the `PDDBACCS` operand in the client environment definitions, the search results are invalid except in the following cases:

- The RDAREA being accessed when the list is searched matches the RDAREA that was accessed when the list was created.
- The RDAREA being accessed when the list is searched is duplicated from the data in the RDAREA that was accessed when the list was created.

Use one of the following methods to search such a list:

- Use the RDAREA that was accessed when the list was created.
- Use an RDAREA that is duplicated from the data in the RDAREA that was accessed when the list was created.
- Re-create the list in the current RDAREA being accessed.

For details about the inner replica facility, see the manual *HiRDB Staticizer Option Version 7 Description and User's Guide*.

(9) Changes when initializing (deleted) lists

As explained in (1) above, lists are deleted when HiRDB terminates. Deletion (and initialization) processing on these lists is performed during HiRDB startup, which means that the more lists that are created, the longer it takes for HiRDB to start. For this reason, we recommend that you consider changing the time at which lists are initialized in the following cases:

- When you wish to minimize HiRDB startup time
- When you wish to minimize system switchover time (using user server hot standby or the rapid system switchover facility)

The list initialization time can be changed as follows with the `pd_list_initialize_timing` operand:

- Initialize lists when the `ASSIGN LIST` statement is executed (`DEFER` specification).
- Initialize lists when the standby HiRDB is started (`STANDBY` specification).

(a) Initializing lists when the ASSIGN LIST statement is executed

With this specification, lists are initialized when the `ASSIGN LIST` statement is executed, rather than when HiRDB starts. This, of course, adds to the initialization overhead when an `ASSIGN LIST` statement is executed. To minimize the initialization processing overhead, set the list RDAREA size and maximum number of lists so that they are as small as possible. To accommodate creation of a large number of lists, create additional list RDAREAs. This enables you to distribute the initialization processing overhead when the `ASSIGN LIST` statement is executed.

(b) Initializing lists when the standby HiRDB is started (rapid system switchover facility only)

With this specification, lists are initialized when the standby HiRDB starts. Lists are not initialized when the system is switched over. Lists are not initialized when the `ASSIGN LIST` statement is executed after the system has been switched over either. Because of this, increase the list RDAREA size and maximum number of lists to reduce the number of list RDAREAs that are created. However, you must first perform the preparatory work explained below.

Preparatory work

You will need lists on both the running system and the standby system. Create them on each of the local disks. You can use either one of the following methods:

Copy the lists created on the running HiRDB to the standby HiRDB

1. Copy all HiRDB file system areas that contain list RDAREAs on the running HiRDB onto the local disk on the standby system.
2. Link so that all HiRDB file names defined in the list RDAREAs point to the HiRDB files system areas that were created in step 1.

Use the re-initialization function of the database structure modification utility (pdmod command)

1. Use the `pdstop` command to terminate the HiRDB on the running system normally.
2. Use the `pdfmkfs` command to create HiRDB file system areas for creating list RDAREAs. Create the same number of list RDAREAs with the same settings as those on the running system.
3. Link so that all HiRDB file names defined in the list RDAREAs point to the HiRDB files system areas created in step 2.
4. Use the `pdstart` command to start the HiRDB on the standby system normally. Because the HiRDB file system areas on the standby HiRDB contain no HiRDB files at this time, the list RDAREAs are opened and placed in error shutdown status.
5. Use the `pdmod` command to re-initialize all list RDAREAs. When doing so, specify only the names of the RDAREAs that are being initialized.
6. Use the `pdrels` command to release all list RDAREAs from shutdown status.
7. Use the `pdstop` command to terminate the standby HiRDB normally.

13.22 Standardizing spaces in table data

Executor: HiRDB administrator and table owner (or user with DBA privilege)

This section explains how to standardize spaces when both double-byte and single-byte spaces are present in table data. The *space conversion facility* is used to standardize spaces.

13.22.1 Overview of space conversion facility

When data is compared, one double-byte space and two single-byte spaces are not recognized as the same data. Therefore, when blanks are represented in table data by both one double-byte space and two single-byte spaces, the retrieval results may be inaccurate.

Example

The following are not recognized as being the same data items:

TV□□21-inch
TV□21-inch

where
□□ indicates two single-byte spaces
□ indicates one double-byte space

The double-byte space character being discussed here is coded as shown below. Two single-byte space characters are coded as `X'2020'`.

- Shift-JIS Kanji Code: `X'8140'`
- EUC Japanese Kanji Code or EUC Chinese Kanji Code: `X'A1A1'`
- Unicode (UTF-8):# `X'E38080'`

#: `NCHAR` and `NVARCHAR` cannot be used if the character codes are Unicode (UTF-8).

(1) Space conversion levels

As shown in Table 13-32, three levels of space character conversion are provided by the space conversion facility.

Table 13-32: Space conversion levels

Level	Explanation
Level 0	No space conversion.

Level	Explanation
Level 1	Converts as follows data spaces that occur in literals, embedded variables, and ? parameters in the data manipulation SQL and data spaces stored by utilities: • When a character string literal is being handled as a national character string literal, two single-byte spaces in succession are converted into one double-byte space; a single occurrence of a single-byte space is not converted. • When a character string literal is being handled as a mixed character string literal, one double-byte space is converted into two single-byte spaces. • When data is being stored in a column of the national character type and when data is being compared with a value expression of the national character type, two single-byte spaces in succession in an embedded variable or ? parameter are converted into one double-byte space; a single occurrence of a single-byte space is not converted. • When data is being stored in a column of the mixed character type and when data is being compared with a value expression of the mixed character type, one double-byte space in an embedded variable or ? parameter is converted into two single-byte spaces.
Level 3	Adds the following processing to the processing of space conversion level 1: • During retrieval of data of a value expression of the national character type, a single double-byte space is converted into two single-byte spaces.

Figure 13-85 illustrates Level 1 processing; Figure 13-86 illustrates Level 3 processing.

Figure 13-85: Level 1 processing

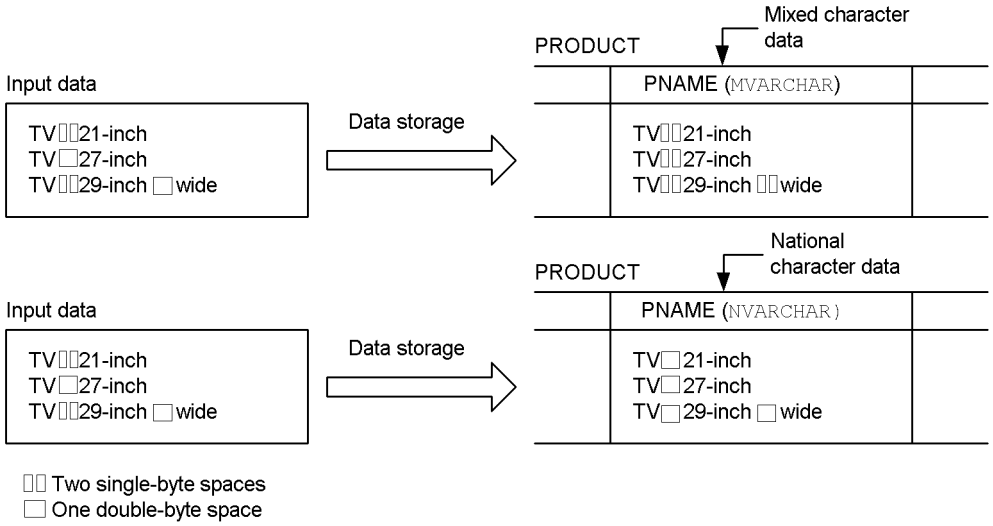
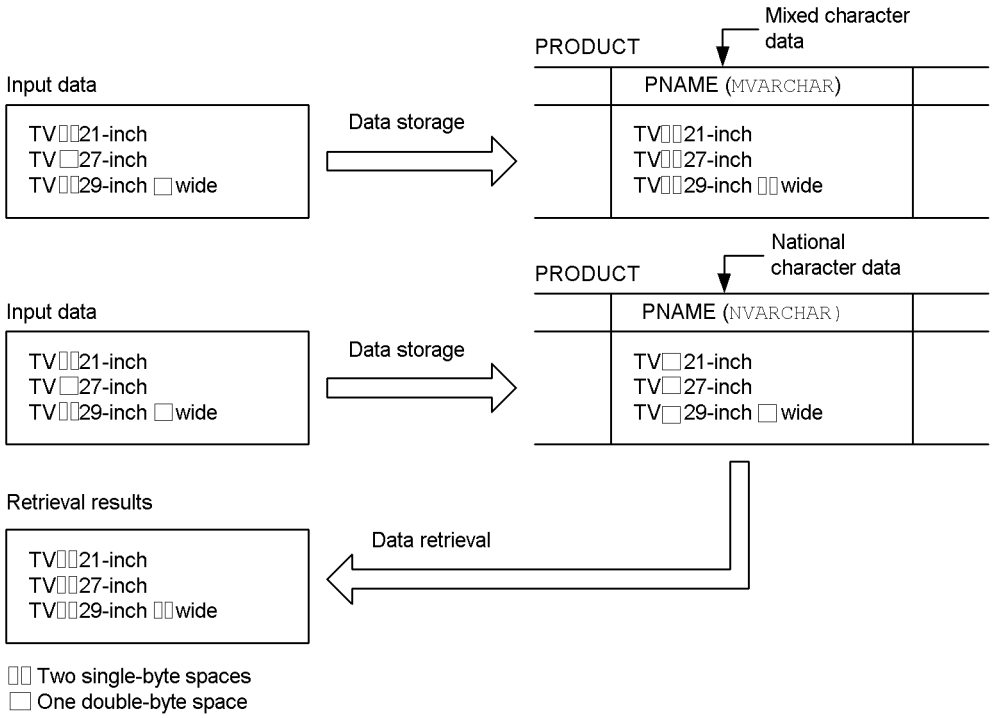


Figure 13-86: Level 3 processing



(2) Notes

1. When the space conversion level is changed, a UAP's retrieval results may not be the same after the change as they were before the change. For this reason, it is not advisable to change the space conversion level if a change in UAP retrieval results is not acceptable.
2. If sorting is performed when Level 3 is specified, HiRDB will perform space conversion on the sort results, and the expected results may not be obtained in all cases.
3. If national character data is retrieved when Level 3 is specified, performance may be poorer than with Level 0 or 1. For example, performance is degraded when large-sized data is retrieved, such as data of the type NVARCHAR (2000). This is because of the overhead involved in converting double-byte spaces into single-byte spaces.
4. When data is being stored in a column of a cluster key, space conversion may cause a uniqueness error. In such a case, you must either store the data without performing space conversion or standardize the space characters in the existing data (For details on standardizing the space characters in existing data, see

13.22.3 Standardizing space characters in a table).

5. Spaces in a national character string are converted in units of two characters, beginning at the beginning of the string.
6. If you are using HiRDB External Data Access, the character codes must be compatible with the character codes on the DBMS of the foreign server. If they are not compatible, unexpected results may be returned.
7. The following must be noted when Level 1 or 3 is specified as the space conversion level:
   - When determining the storage target RDAREA using a UAP that uses a hash function for table partitioning on a hash partitioned table, specify a space conversion level in the argument of the hash function for table partitioning. If no space conversion level is specified, the result of the hash function for table partitioning may be corrupt.
   - When a UAP is used to apply key range partitioning to a table (and the partitioning key is national character data or mixed character data), convert the partitioning key values with the space conversion function. Otherwise, the key range partitioning results may be invalid.

   For details on the hash function for table partitioning and the space conversion function, see the manual *HiRDB Version 8 UAP Development Guide*.

13.22.2 Setting the space conversion level

The desired space conversion level can be specified in any of the following operands:

- `pd_space_level` operand in the system common definition
- `PDSPACELVL` operand in the client environment definition
- `spacelvl` operand in the `option` statement of the database load utility
- `spacelvl` operand in the `option` statement of the database reorganization utility

13.22.3 Standardizing space characters in a table

The procedure for standardizing space characters is explained below.

Procedure

To standardize space characters:

1. Standardize the codes used to represent space characters in the existing data, as explained in *(1)* below.
2. Specify `pd_space_level=1` in the system common definition.

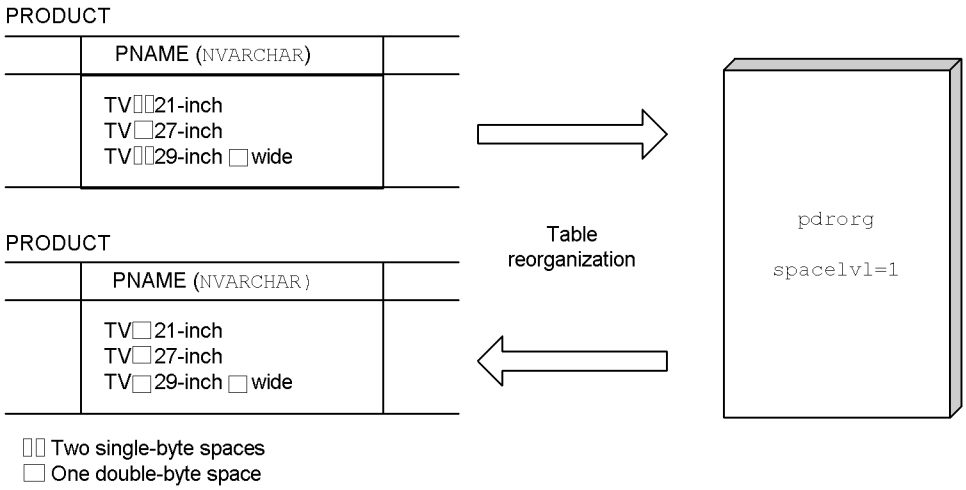
   This specification standardizes the space characters in the data to be stored from now on (by means of data loading or `INSERT`).

3. Retrieve data by specifying `PDSPACELVL=3` in the client environment definition as needed.

(1) Standardizing space characters in existing data

Figure 13-87 illustrates the procedure for standardizing the space characters in existing data.

Figure 13-87: Procedure for standardizing space characters in existing data



Explanation

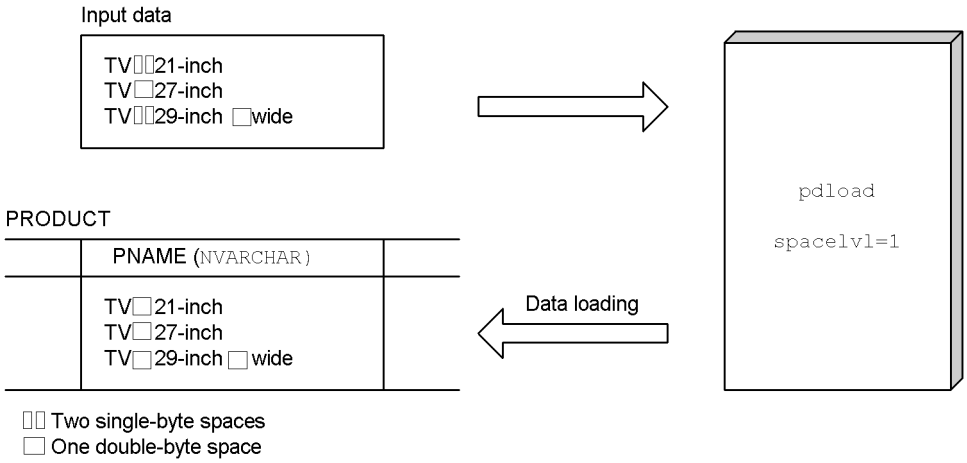
To standardize the space characters in existing data, use the database reorganization utility (`pdrorg` command) to reorganize the table. Specify `spacelvl=1` in the `option` statement of the `pdrorg` command.

For details on reorganizing a table, see *13.2 Reorganizing a table* and *13.3 Reorganizing a table (examples)*.

(2) Standardizing space characters in new data

Figure 13-88 illustrates the procedure for standardizing the space characters in new data.

Figure 13-88: Procedure for standardizing space characters in new data



Explanation

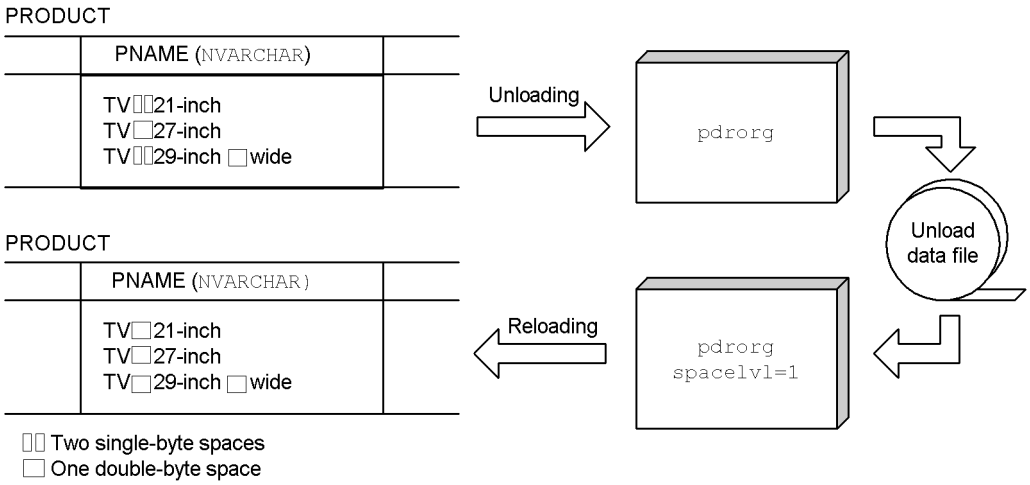
To standardize the space characters in new data, specify `spacelvl=1` in the `option` statement when the data is loaded with the database load utility (`pdload` command). If the `spacelvl` operand in the `option` statement is omitted, the value specified in the `pd_space_level` operand in the system common definition will be assumed.

For details on loading data, see the manual *HiRDB Version 8 Command Reference*.

(3) Migrating table data to another system

Figure 13-89 illustrates the procedure for standardizing the space characters when table data is migrated to another system.

Figure 13-89: Procedure for standardizing space characters when table data is migrated to another system



Explanation

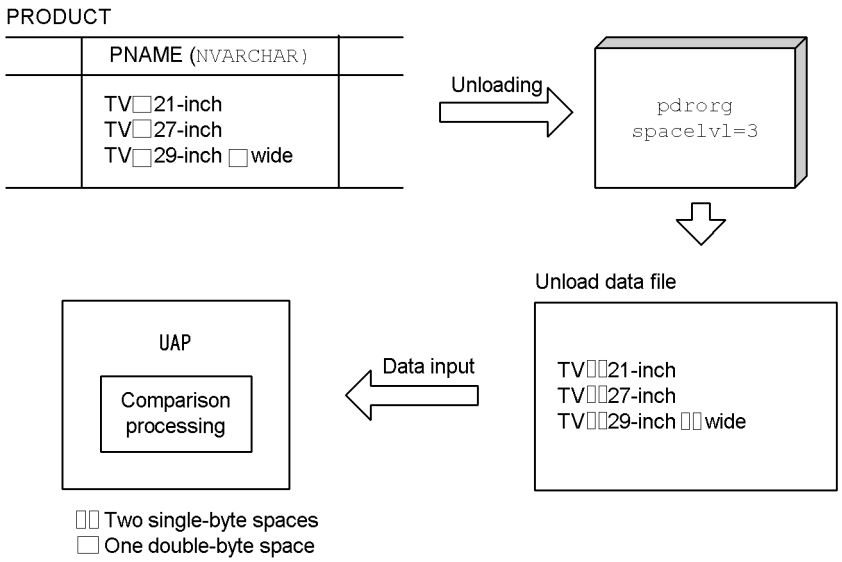
In the case of table data that is migrated to another system, specify `spacelvl=1` in the `option` statement of the database reorganization utility (`pdrorg` command) when the data is loaded at the migration destination system.

For details on migrating table data to another system, see *12.1 Migrating a table to another HiRDB system*.

(4) Using data unloaded with the -W option specified in a UAP

Figure 13-90 illustrates the procedure for converting space characters when data unloaded with the `-W` option specified is used by a UAP (i.e., when this UAP performs comparison processing). In this case, the space characters in the table must already be standardized.

Figure 13-90: Using data unloaded with the -W option specified in a UAP



Explanation

- When the data is unloaded, specify `spacelvl=3` in the `option` statement of the database reorganization utility (`pdrorg` command). Each double-byte space in the national character data is converted into two single-byte spaces.
- The comparison processing of national character data by the UAP is carried out in units of single-byte spaces.

13.22.4 Distributed database environment

(1) Data substitution or comparison

During data substitution or comparison, the space conversion level of the distributed server is applied; the space conversion level of the distributed client is ignored.

If the distributed server is XDM/RD or XDM/RD E2, the specification in the `KEIS CODE SPACE LEVEL` operand of the RD environment definition is applied.

(2) Data retrieval

(a) When space conversion level of distributed client is 0 or 1

When data is retrieved with `pd_space_level=0` or `1` (or `PDSPACELVL=0` or `1`) specified in the distributed client, the space conversion level of the distributed server is applied; the space conversion level of the distributed client is ignored.

For example, when Level 3 is specified in the distributed server, each double-byte

space in national character data is converted into two single-byte spaces.

If the distributed server is XDM/RD or XDM/RD E2, the specification in the `KEIS CODE SPACE LEVEL` operand of the RD environment definition is applied.

(b) When space conversion level of distributed client is 3

When data is retrieved with `pd_space_level=3` (or `PDSPACELVL=3`) specified in the distributed client, the space conversion level of the distributed server is not applied. In this case, HiRDB at the distributed client converts each double-byte space in the retrieval results into two single-byte spaces. One double-byte space is also converted into two single-byte spaces when the data type of the column to be accessed is national character data, mixed character data, or character data. Consequently, the spaces in the retrieval results data returned to the UAP are always single-byte spaces.

13.23 Converting the sign portion of the decimal type

Executor: HiRDB administrator and table owner (or user with DBA privilege)

The three data formats decimal, day interval, and time interval are signed, packed formats consisting of an integer portion and a sign portion. Normally, HiRDB stores as is in a database the sign that is input from a UAP or utility,# using X'C' (positive), X'D' (negative), or X'F' (positive) as valid values for the sign portion of signed packed data. HiRDB also treats +0 (sign portion X'C' or X'F') and -0 (sign portion X'D') as different values.

The facility for conversion to a decimal signed normalized number is used to convert the sign portion of decimal, day interval, and time interval signed packed format data. This section explains how to use the facility for conversion to a decimal signed normalized number to convert the sign portion of signed packed format data.

#: Format conversion during SQL execution or computation may also result in sign conversion. The sign may also be converted when a multicolumn index is used.

13.23.1 Overview of the facility for conversion to a decimal signed normalized number

(1) Specification of the sign portion of the signed packed format

Table 13-33 shows the specification of the sign portion of signed packed format data in HiRDB.

Table 13-33: Specification of the sign portion of the signed packed format data

Sign portion	Meaning
X'C'	Indicates a positive value.
X'D'	Indicates a negative value.
X'F'	Indicates a positive value.

(2) Rules for converting the sign portion of signed packed format data

When the facility for conversion to a decimal signed normalized number is used, the sign portion of signed packed format data is converted at the time of data input according to the rules described in Tables 13-34 and 13-35. This conversion of the sign portion is called *normalization of the sign portion*. Once the sign portion is normalized, +0 and -0 can be handled as the same value.

Table 13-34: Rules for converting the sign portion of the signed packed format data (for non-0 data)

Sign portion of embedded variable data	Without normalization	With normalization
X'A'	Error	Converted to X'C'
X'B'	Error	Converted to X'D'
X'C'	No conversion	No conversion
X'D'	No conversion	No conversion
X'E'	Error	Converted to X'C'
X'F'	No conversion	Converted to X'C'
X'0' to X'9'	Error	Error

Table 13-35: Rules for converting the sign portion of the signed packed format data (for 0 data)

Sign portion of 0 data	Without normalization	With normalization
X'A'	Error	Converted to X'C'
X'B'	Error	
X'C'	No conversion	
X'D'	No conversion	
X'E'	Error	
X'F'	No conversion	

(3) Application criteria

To use in HiRDB a UAP that specifies the sign portion in a different way, it may be desirable to use the facility for conversion to a decimal signed normalized number. This facility should be used only after checking carefully the sign conversion rules.

For example, when an XDM/RD or XDM/RD E2 UAP is moved to HiRDB, it may be appropriate to use the facility for conversion to a decimal signed normalized number, because XDM/RD and HiRDB specify the decimal signed portion differently.

(4) Environment setting

To use the facility for conversion to a decimal signed normalized number, specify `Y` in the `pd_dec_sign_normalize` operand in the system common definition.

If possible, this operand should be specified when HiRDB is first installed. To

normalize the signed portion while HiRDB is already being used, it is necessary to reload the data of tables for which the decimal type is defined.

(5) Notes

1. When an SQL statement is executed while there is both normalized data and data that has not been normalized, corruption of the database or of the execution results may result. Thus, if you change the specification value of the `pd_dec_sign_normalize` operand, data must be reloaded and the normalization status must be standardized by executing the database reorganization utility (`pdrorg` command) and database load utility (`pdload` command); for details, see *13.23.2 Normalizing existing data*.
2. The following points must be noted when the facility for conversion to a decimal signed normalized number is used:
   - When you determine a storage destination RDAREA while using a UAP that uses a hash function for table partitioning on a hash partitioned table, specify use of the facility for conversion to a decimal signed normalized number in the arguments of the hash function for table partitioning. Otherwise, the results of the hash function for table partitioning may be invalid.
   - When a UAP is used to apply key range partitioning to a table with key range partitioning (the partitioning key is decimal data), you should use the facility for conversion to a decimal signed normalized number to convert the partitioning key values. If the partitioning key values are not converted, the key range partitioning results may be invalid.

   For details on the hash function for table partitioning and of the facility for conversion to a decimal signed normalized number, see the manual *HiRDB Version 8 UAP Development Guide*.

13.23.2 Normalizing existing data

A table (`TABLE01`) contains a decimal-type column. This decimal sign portion is normalized.

(1) Specify pd_dec_sign_normalize=Y

Use the `pdstop` command to terminate HiRDB normally. Once stopped, specify `Y` in the `pd_dec_sign_normalize` operand, and then use the `pdstart` command to start HiRDB normally.

The system reconfiguration command (`pdchgconf` command) can be used to change HiRDB system definitions while HiRDB is running. Note that HiRDB Advanced High Availability must be installed in order to use this command. For details about changing HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

(2) Use the pdrorg command to unload data from TABLE01

```
pdrorg -k unld -W bin -j -t TABLE01 -g /pdrorg/unld01
```

Explanation

`-k`: Specifies `unld` for unloading.

`-W bin`: Specifies that the unload data file is to be used as the input file (binary format) for the `pdload` command.

`-j`: Specifies that a LOB column is defined in the table that is to be unloaded.

`-t`: Specifies the name of the table that is to be unloaded.

`-g`: Specifies that `TABLE01` is row-partitioned between servers in a HiRDB/Parallel Server. Specifying the `-g` option consolidates the unload data files (into a single file).

`/pdrorg/unld01`: Specifies the name of the control statements file for the `pdrorg` command.

(3) Use the pdload command to load data into TABLE01

```
pdload -d -b -W TABLE01 /pdload/load01
```

Explanation

`-d`: Specifies the creation mode.

`-b`: Specifies that the data to be loaded is in binary format.

`-W`: Specifies that a binary-format input data file, created with the `pdrorg` command, is to be used.

`TABLE01`: Specifies the name of the table into which the data is to be loaded.

`/pdload/load01`: Specifies the name of the control statements file for the `pdload` command.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

Supplemental

A binary input file (`-W bin`) is used in the procedure described above. However, if a DAT input file (`-W dat`) is used, the sign is first converted into characters during unloading. Then, the conversion performed during data loading

normalizes the signed portion automatically. Therefore, the signed portion is normalized regardless of the value specified in the `pd_dec_sign_normalize` operand.

Chapter
14. Handling Indexes

This chapter explains the procedures for handling indexes.

It contains the following sections:

14.1 Improving index storage efficiency (index reorganization)

This section explains how to improve index storage efficiency (index reorganization).

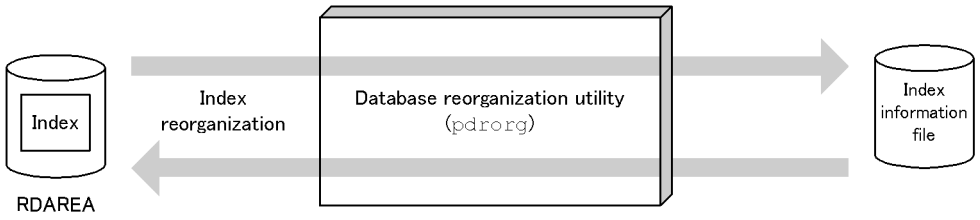
14.1.1 Overview of index reorganization

When data is deleted (DELETE) or updated (UPDATE) repeatedly, index storage efficiency deteriorates, reducing the efficiency of index-based retrievals. Any of the following measures can be taken with the database reorganization utility to prevent this:

- Index reorganization
- Index re-creation
- Table reorganization

This section explains index reorganization, which is the best measure in terms of performance. Figure 14-1 provides an overview of index reorganization processing.

Figure 14-1: Overview of index reorganization processing



Explanation

An index information file is created by retrieving index key information, and the index is rearranged based on this information. This is called *index reorganization*. Index reorganization can be executed by index or by index RDAREA.

(1) Application range

Reorganization of a plug-in index cannot be executed.

(2) Application criteria

Index reorganization is used for freeing the unusable areas in index storage pages that have resulted from addition, deletion, or updating of a large volume of data.

(3) When to use index reorganization and when to use table reorganization

- Use index reorganization when there has been frequent data updating (UPDATE).
- Use table reorganization when there has been frequent data deletion (DELETE) or

addition (INSERT).

- When reorganizing a table would take too much time, reorganizing its indexes only can reduce retrieval time.

(4) Difference from index re-creation

In index re-creation, the table's data is retrieved. In contrast, the table's data is not retrieved in index reorganization. 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 when the following condition is satisfied:

number-of-used-pages-in-RDAREAs-storing-table > *number-of-used-pages-in-RDAREAs-storing-index*

(5) Notes on index reorganization

When multiple indexes in the same RDAREA are reorganized simultaneously, the pdhold command must be used to shut down the RDAREA before executing the reorganization. Then, after index reorganization is completed, the pdrels command must be used to release the RDAREA's shutdown status.

(6) Reducing the reorganization execution time

If no database update log is being collected during index reorganization (no-log mode or pre-update log acquisition mode is in effect), there is a commensurate reduction in processing time. Collection of a database update log is specified with the -l option of the pdrorg command. The default is the pre-update log acquisition mode.

(7) Reorganizing an index in an RDAREA that has insufficient free space

When an index is reorganized, a percentage of unused area per page can be specified in the PCTFREE operand of CREATE TABLE or CREATE INDEX. 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.

Note that this is a temporary measure that is used when the RDAREA cannot be immediately expanded before index reorganization. In preparation for data updating, you should use the database structure modification utility (pdmod command) to expand the RDAREA so that reorganization can be performed that is within the value specified in the PCTFREE operand of CREATE TABLE.

14.1.2 Example 1: Reorganizing an index

In this example, an index (INDEX01) defined for a table (TABLE01) is reorganized. The following conditions apply at the time of reorganization:

- The RDAREA storing the index is RDAREA1
- RDAREA1 contains INDEX01 only
- Reorganization is performed in the pre-update log acquisition mode

(1) Use the pdhold command to shut down the RDAREA to be reorganized

```
pdhold -r RDAREA1
```

(2) Use the pdrorg to reorganize the index

```
pdrorg -k ixor -t TABLE01 /pdrorg/rorg01
```

Explanation

-k: Specifies ixor, which is used to reorganize the index.

-t: Specifies the name of the table for which the index is defined.

/pdrorg/rorg01: Specifies the name of the control statements file for the pdrorg command. The following shows an example of a control statements file specification:

```
idxname name=INDEX01
idxwork bes1 /pdrorg
```

INDEX01: Index identifier

bes1: Name of the server on which the index information file is created (specified when a HiRDB/Parallel Server is being used)

/pdrorg: Name of the directory in which the index information file is created

(3) Back up the RDAREA that was reorganized

Because you reorganized the index in the pre-update acquisition mode (default), back up the RDAREA containing the index that was reorganized (RDAREA1). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(4) Use the pdrels command to release the RDAREA from shutdown status

```
pdrels -r RDAREA1
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

14.1.3 Actions when an error occurs during index reorganization

This section explains the actions that can be taken when an error occurs during index reorganization and the `pdrorg` command terminates abnormally. The operational procedure after the `pdrorg` command has terminated abnormally is explained below:

Procedure

To perform corrective actions:

1. Check if the `KFPL00715-I` message has been output to the message log file or the syslogfile. This message indicates that index loading has started; if it has been output, an index information file has been created.
2. If the `KFPL00715-I` message has not been output, execute the `pdrorg` command (index reorganization) again. However, in the case of abnormal termination due to entry of the `pdcancel` command, etc., the index key information may have been lost, rendering re-execution of index reorganization impossible. In such a case, execute index re-creation (`-k ixrc` specification).
3. If the `KFPL00715-I` message has been output, perform the following operation using the created index information file as the input file:

   - **When index reorganization is being executed in a mode other than the no-log mode:**

     Use the `pdrorg` command to create the index in the batch mode (`-k ixmk` specification).

   - **When index reorganization is being executed in the no-log mode:**

     The RDAREAs that store the index are in error shutdown status, so their error shutdown status must be released by using the `pdmod` command to reinitialize the RDAREAs. Then use the `pdrorg` command to re-create the index (`-k ixrc` specification).

Notes

One of the common errors that may occur during index reorganization is a space shortage in an RDAREA storing the index. The actions to be taken in such a case

are explained in *14.1.4 Example 2: When an RDAREA shortage occurs during index reorganization (execution in a mode other than no-log mode)* and *14.1.5 Example 3: When an RDAREA shortage occurs during index reorganization (execution in no-log mode)*.

However, if a value other than 0 was specified in the `PCTFREE` operand during index definition, specifying the `idxfree` operand in the `option` statement of the `pdrorg` command may be sufficient without having to expand the index-storage RDAREA.

14.1.4 Example 2: When an RDAREA shortage occurs during index reorganization (execution in a mode other than no-log mode)

A space shortage in an index-storage RDAREA during index reorganization has caused the `pdrorg` command to terminate abnormally.

(1) Increase the size of the RDAREA storing the index

For details about increasing the size of RDAREAs, see *15.3 Increasing the size of an RDAREA (RDAREA expansion)*.

(2) Use the pdhold command to shut down the RDAREA storing the index

```
pdhold -r RDAREA1
```

(3) Use the pdrorg command to create the index in the batch mode

Using as input information the index information file that was output during index reorganization, perform batch index creation (`-k ixmk` specification).

```
pdrorg -k ixmk -t TABLE01 /pdrorg/rorg01
```

Explanation

`-k`: Specifies `ixmk`, which is used to perform batch creation of the index.

`-t`: Specifies the name of the table for which the index is defined.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command. The following shows an example of a control statements file specification:

```
index INDEX01 RDAREA1 /pdrorg/index_inf01
```

`INDEX01`: Index identifier

RDAREA1: Name of the RDAREA storing the index

/pdrorg/index_inf01: Name of the index information file

(4) Back up the RDAREA storing the index

Because you created the index in the batch mode in the pre-update acquisition mode (default), back up the RDAREA storing the index (RDAREA1). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release the RDAREA from shutdown status

```
pdrels -r RDAREA1
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

14.1.5 Example 3: When an RDAREA shortage occurs during index reorganization (execution in no-log mode)

A space shortage in an index-storage RDAREA during index reorganization has caused the pdrorg command to terminate abnormally.

(1) Re-initialize the RDAREA storing the index

Because the RDAREA storing the index is in error shutdown status, use the pdmod command to re-initialize the RDAREA storing the index and release it from error shutdown status. While doing so, increase the size of the RDAREA storing the index. For details about re-initializing RDAREAs, see *15.4 Increasing the size of an RDAREA or modifying its attributes (RDAREA reinitialization)*.

(2) Use the pdhold command to release the RDAREAs storing the index and the table

```
pdhold -r RDAREA1,RDAREA2
```

(3) Use the pdrorg command to re-create the index

All indexes stored in the re-initialized RDAREA must be re-created.

```
pdrorg -k ixrc -t TABLE01 /pdrorg/rorg01
```

Explanation

-k: Specifies ixrc, which is used to re-create the index.

`-t`: Specifies the name of the table for which the index is defined.

`/pdrorg/rorg01`: Specifies the name of the control statements file for the `pdrorg` command. The following shows an example of a control statements file specification:

```
idxname name=INDEX01
idxwork bes1 /pdrorg
sort bes1 /tmp/sortwork/,8192
```

`INDEX01`: Index identifier

`bes1`: Name of the server on which the index information file is created (specified when a HiRDB/Parallel Server is being used)

`/pdrorg`: Name of the directory in which the index information file is created

`bes1`: Name of the server on which the work file for sorting is created (specified when a HiRDB/Parallel Server is being used)

`/tmp/sortwork/`: Name of the work directory for sorting

(4) Back up the RDAREAs storing the index and the table

Back up the RDAREAs storing the index and the table (`RDAREA1` and `RDAREA2`). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release the RDAREAs storing the index and the table from shutdown status

```
pdrels -r RDAREA1,RDAREA2
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

14.2 Defining an index for a table that contains data

Executor: Table owner

An index can be added to a table that already contains data in order to modify the table processing procedure or to improve table retrieval performance.

Note:

An index (B-tree index) cannot be defined for a column for which an abstract data type is defined (however, this does not apply in the case of an abstract data type provided by a plug-in).

(1) Index addition procedure

The following is the procedure for defining an index for a table that contains data:

Procedure

To define an index:

1. Use `CREATE INDEX` to define the index. If a UAP is accessing the table while the index is being defined, the UAP is placed in lock-release wait status.
2. If necessary, the optimizing information collection utility (`pdgetcst` command) should be executed after the index has been defined. Plug-in indexes are not processed by the optimizing information collection utility. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.
3. Defining an index invalidates the index information of any stored routines that use the table for which that index is defined. If this happens, use the `ALTER PROCEDURE` or `ALTER ROUTINE` statement to re-create each stored routine.

   In addition, defining an index for a table specified in a trigger SQL statement invalidates the index information of that trigger. If this happens, use the `ALTER TRIGGER` or `ALTER ROUTINE` statement to re-create the trigger.

(2) Reducing index creation time (EMPTY option)

When an index is to be defined for a table that has a large amount of data, it will take a considerable amount of time to create the index entity (to execute `CREATE INDEX`), and it will not be possible to execute any other definition SQL during this period.

However, if `CREATE INDEX` is executed with the `EMPTY` option specified, an index definition is created without creating the actual index entity. This is called an

unfinished index. Because the index entity is not created, execution of `CREATE INDEX` is completed immediately, so that execution of other definition SQLs is not delayed.

Note that the `EMPTY` option can also be specified for a plug-in index.

Reference note:

1. Because no index entity has been created for an unfinished index, the unfinished index cannot be used for retrieval processing, nor can a column be updated in a table for which an unfinished index is defined (if updating is attempted, an SQL error results).
2. The database condition analysis utility (`pddbst` command) can be used to determine whether or not an index is unfinished. In the case of condition analysis by index or by RDAREA (logical analysis), an unfinished index is indicated under the *status* heading; in the case of cluster key and clustering data page storage condition analysis, an unfinished index is reported in a warning message.
3. The index re-creation facility (`-k ixrc`) of the database reorganization utility (`pdrorg` command) is used to create an index entity. When an unfinished index's index entity is created, the unfinished status is released. If the table is deleted in its entirety by the `PURGE TABLE` statement, all indexes for the table are released from unfinished status.
4. An index for a partitioned table is also partitioned and stored in multiple RDAREAs, and the unfinished status is managed for each partitioned index. The database reorganization utility (`pdrorg` command) can create the index entity for each RDAREA that stores a part of the index. When only a part of the partitioned index is created, SQLs may or may not execute successfully, depending on the specified conditions.

14.3 Deleting an index

Executor: HiRDB administrator and table owner (or user with DBA privilege)

When changes are made to the processing procedure for a table that contains data, it may develop that some indexes are no longer needed because they are associated with table retrieval conditions that are no longer applicable. If such unneeded indexes are retained, extra processing time is required for them when rows or columns are added or updated. It is more efficient if unneeded indexes are deleted immediately.

The following is the procedure for deleting an index:

Procedure

To delete an index:

1. Use `DROP INDEX` to delete the index. If a UAP is accessing the table while the index is being deleted, the UAP is placed in lock-release wait status.
2. If necessary, the optimizing information collection utility (`pdgetcst` command) should be executed after the index has been deleted. Plug-in indexes are not processed by the optimizing information collection utility. For details about whether or not execution of the optimizing information collection utility is required, see the manual *HiRDB Version 8 Command Reference*.
3. When an index is deleted, any stored routines that use the deleted index become invalid, and the index information of any stored routines that use the table for which the deleted index is defined become invalid. If this happens, use the `ALTER PROCEDURE` or `ALTER ROUTINE` statement to re-create each stored routine.

   In addition, if an index that is used by a trigger is deleted, the trigger becomes invalid. Deleting an index for a table specified in a trigger SQL statement invalidates the index information of that trigger. If this happens, use the `ALTER TRIGGER` or `ALTER ROUTINE` statement to re-create the trigger.
4. If any index-only global buffer is assigned to the deleted index, delete that global buffer. The procedure explained below is used to delete a global buffer.

Deleting a global buffer

Use one of the following methods to delete a global buffer:

1. Terminate HiRDB normally, and then change the `pdbuffer` operand specification (delete the `pdbuffer` operand associated with the index that was deleted).

2. Use the system reconfiguration command (`pdchgconf` command) to change the `pdbuffer` operand specification (delete the `pdbuffer` operand associated with the index that was deleted). This method eliminates the need to terminate HiRDB normally. Note that HiRDB Advanced High Availability must be installed in order to use this command. For details about changing HiRDB system definitions using the system reconfiguration command, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

3. Use the `pdbufmod` command to delete the global buffer. For details about deleting global buffers with the `pdbufmod` command, see *9.3 Adding, modifying, and deleting global buffers while HiRDB is running (dynamic updating of global buffers)*. To use this method to delete global buffers, both of the following conditions must be satisfied

   - HiRDB Advanced High Availability is installed.
   - The value `Y` is specified in the `pd_dbbuff_modify` operand.

   When you use this method, the information associated with the deleted global buffer becomes invalid if HiRDB is terminated normally or through a planned termination. Therefore, change the `pdbuffer` operand specification while HiRDB is stopped (delete the `pdbuffer` operand associated with the index that was deleted). If you do not change the `pdbuffer` operand specification, an error will occur when HiRDB starts, because the global buffer is not assigned to any index.

14.4 Creating a definition SQL from an existing index

Executor: HiRDB administrator or a user with DBA privilege

The `pddefrev` command can be used to create a definition SQL from an existing index.

This command is useful for creating a new index by using an existing index's definition. A definition SQL created with the `pddefrev` command is used as the input information to the database definition utility (`pddef` command).

Reference note:

Plug-in indexes are not processed by the `pddefrev` command (a definition SQL cannot be created).

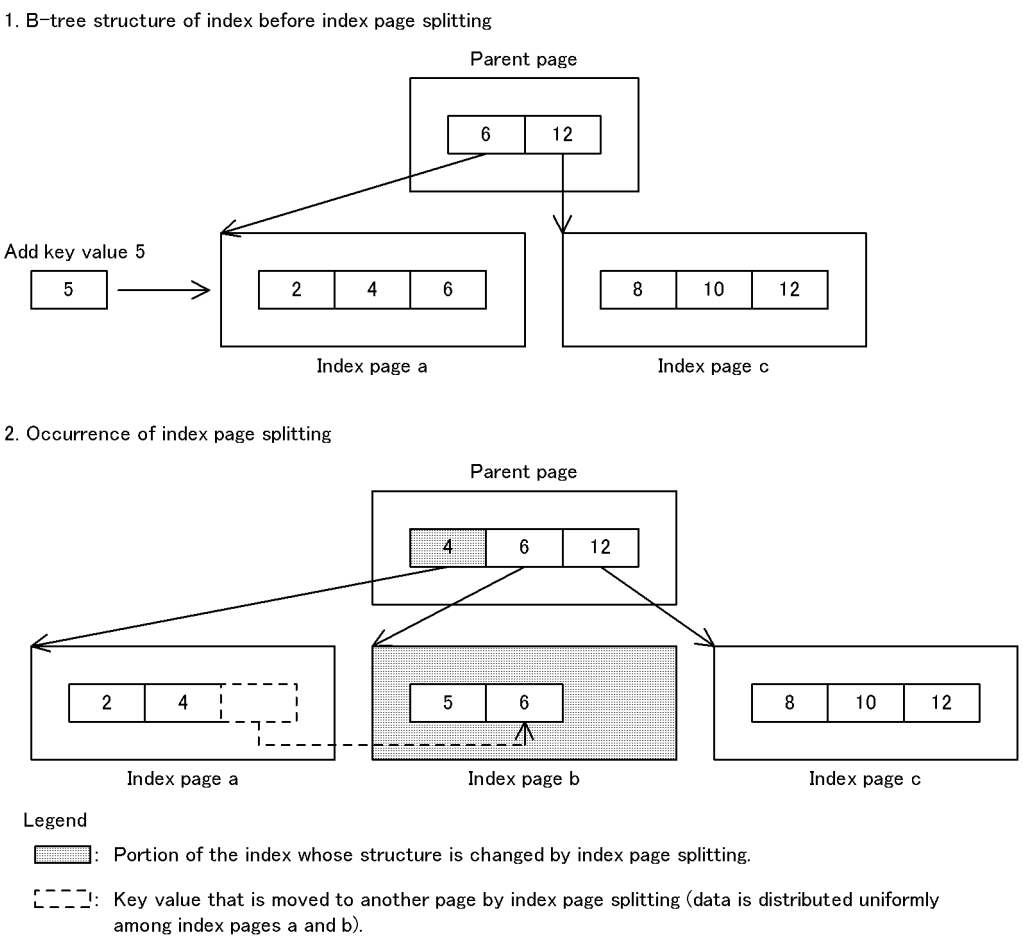
14.5 Reducing the number of index page splits (unbalanced index split)

Executor: Table owner

(1) Index page splitting

HiRDB indexes are based on a B-tree structure. When there is not enough space to add a key value to an index page, index page splitting occurs. Index page splitting is the method used by HiRDB to obtain additional free space when there is not enough space to add a key value to an index page. This method involves dividing the index information in the index page into two equal-sized halves and moving the latter half to a new page. Figure 14-2 shows an example of a typical index page split

Figure 14-2: Example of index page splitting



(a) Increase in the amount of log information due to index page splitting

As data is added to a table that has an index, key values are also added to the index pages. If the percentage of unused space in the index pages is small, index page splitting may occur frequently. Frequent index page splitting results in an increase in the amount of log information due to changes made to the index structure.

(b) Reducing the frequency of index page splits

The frequency of index page splits can be reduced by changing the percentage of unused space in the index pages; the percentage value is obtained from the index page split information acquired by executing the statistics analysis utility (`pdstedit`

command). The percentage of unused space in the index pages is changed by modifying the value of the `PCTFREE` operand in the `CREATE INDEX` definition SQL. When this is done, the index must then be redefined.

For details on using the index page split information acquired by the statistics analysis utility, see *21.7 Tuning indexes*.

(2) Unbalanced index split

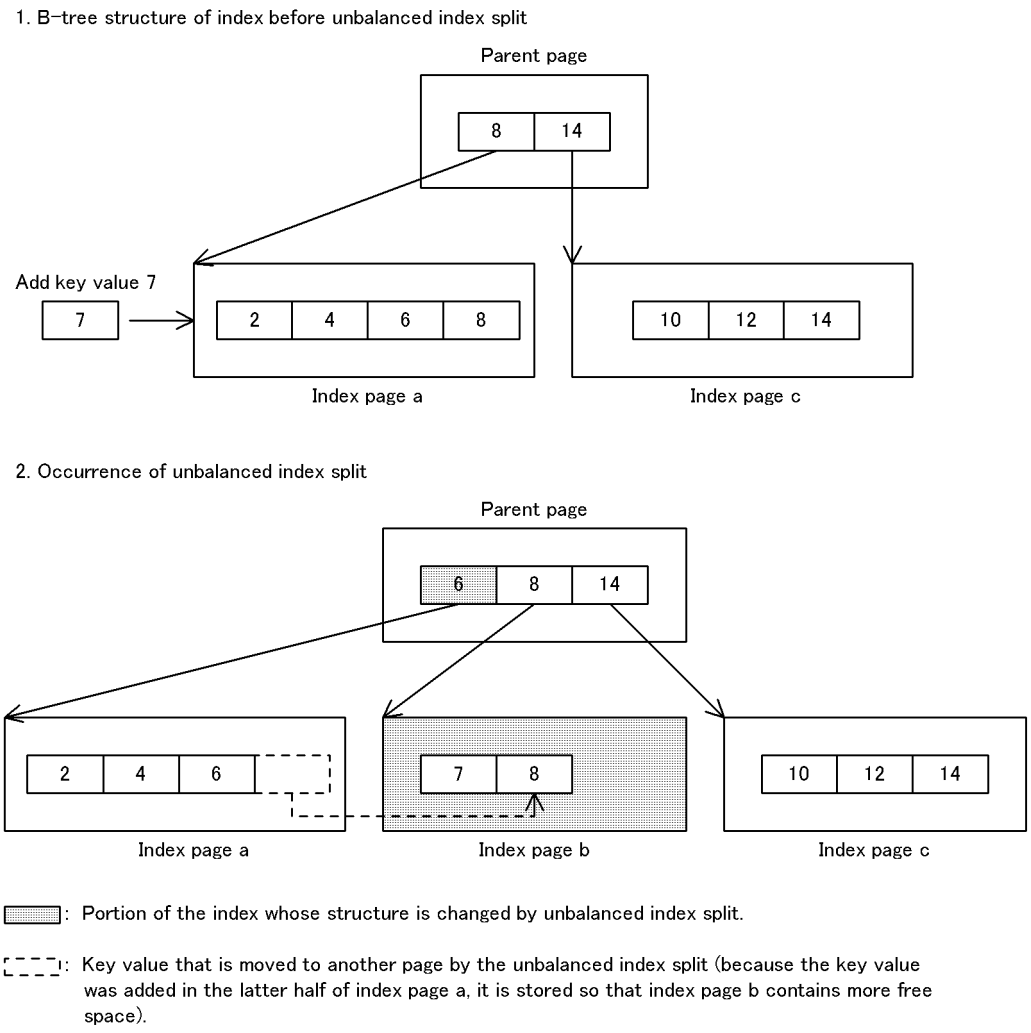
In the case of normal index page splitting, adding consecutive key values reduces the index page data storage efficiency. For this situation, an unbalanced index split can be used. *Unbalanced index split* is a method for dividing unevenly the index information in an index page, rather than dividing it uniformly into two equal-size halves. This method improves the data storage efficiency when consecutive key values are added.

When unbalanced index split is specified, the ratio used for dividing the index information in an index page is determined by the location in the page at which a new key value is to be inserted.

If the insertion location is in the first half of the index page, the key value is expected to be added in the first part of the page. Therefore, the first portion is stored in the page on the left-hand side using as the split location the next key value that is greater than the key being added.

If the location of the key value to be added is in the second half of the index page, the key is expected to be added in the latter part of the page. Therefore, the latter portion is stored in the page on the right-hand side using as the split location the key value that is being added. Figure 14-3 shows an example of an unbalanced index split when a key value is to be added in the latter half of an index page.

Figure 14-3: Example of an unbalanced index split



(a) Scope of an unbalanced index split

Unbalanced index split is applicable only to leaf pages other than the last page. During rollback, regular index page splitting is used instead of an unbalanced index split. In this case, a page is split at a 50:50 ratio.

On the last leaf page, data is divided on the basis of the value specified in the `PCTFREE` operand in the `CREATE INDEX` definition SQL statement in order to handle addition of key values in ascending order for data loading, etc.

(b) Criteria for an unbalanced index split

Unbalanced index split is effective when it is applied to an index that satisfies the following conditions:

- There is balance in terms of duplication of key values.
- Lengths of key values are close to uniform.
- Consecutive intermediate key values are added frequently.

(c) Specification

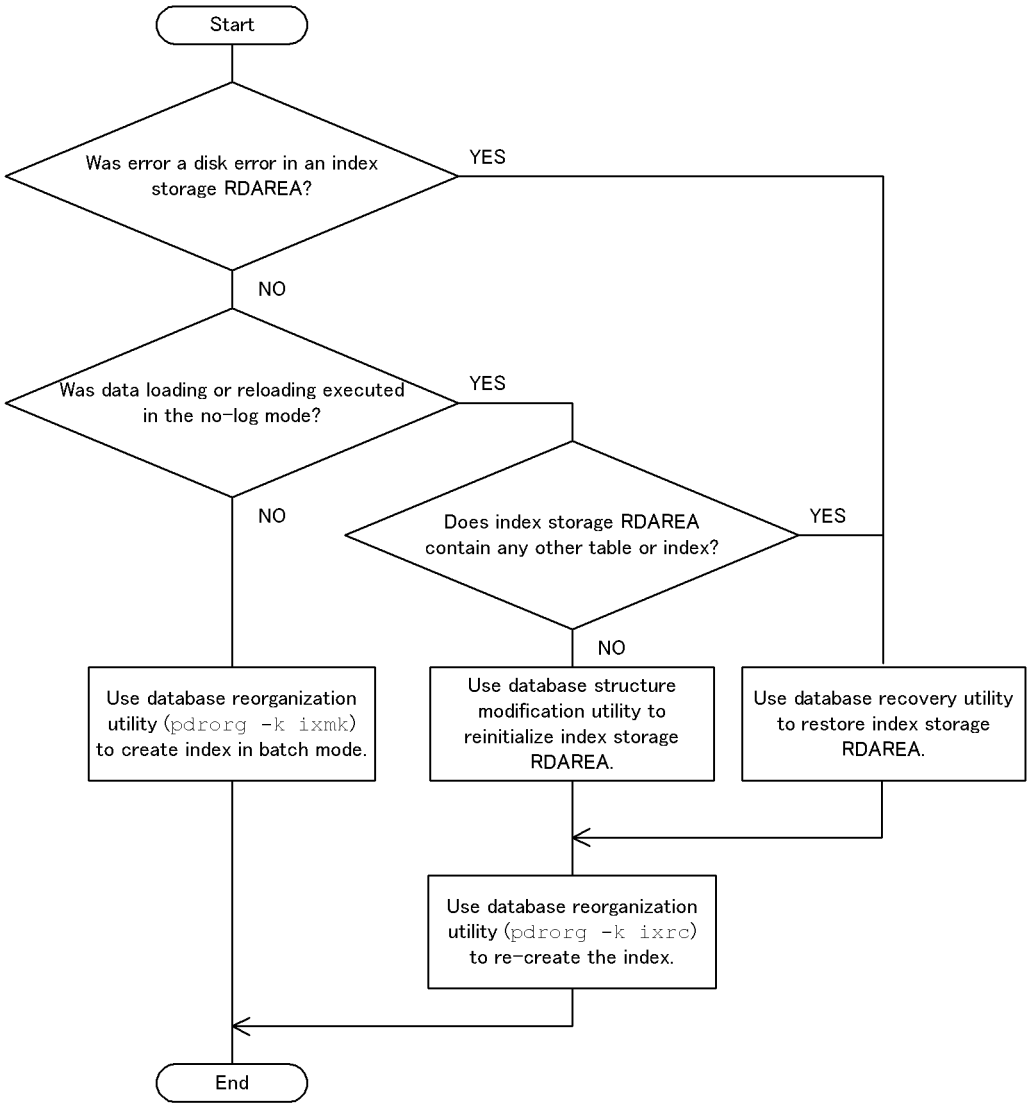
To use unbalanced index split, `UNBALANCED SPLIT` is specified as the index option in the `CREATE INDEX` or `CREATE TABLE` definition SQL statement.

14.6 Error handling during batch index creation

Executor: HiRDB administrator

When batch index creation is specified during data loading or reloading (during execution of the database load utility or database reorganization utility), batch index creation processing is executed once data loading or reload processing is completed. When batch index creation is specified during data loading or reloading, batch index creation processing is executed once data load or reload processing is completed. If an error occurs during the batch index creation processing (`KFPL00716-I` message is output), the table data is in completed status but the index data is in incomplete status. Figure 14-4 shows the procedure for recovering the index data in this situation.

Figure 14-4: Procedure for recovering index data



14.6.1 Example of recovery when reloading (data loading) was performed in the log acquisition mode or the pre-update log acquisition mode

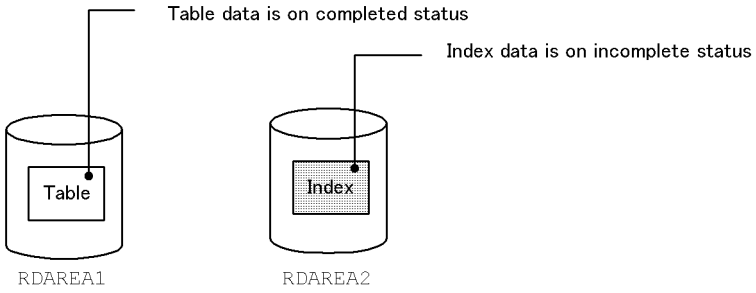
Example

- A table for which an index is defined was reloaded. However, because the size of the RDAREA storing the index was too small at this time (or a sort

processing error occurred), an error occurs during batch index creation. When a sort processing error occurs, the KFPL15062-E message is issued.

- Data was loaded in the addition mode into a table for which a plug-in index is defined. However, because the size of the RDAREA storing the index was too small at this time, an error occurs during batch plug-in index creation.

In such a case, the data is in the following status:



(1) Use the pdmod command to increase the size of the RDAREA storing the index

If an error occurs because the RDAREA is too small, use the pdmod command to increase the size of the RDAREA storing the index (RDAREA2). For details about increasing the size of RDAREAs, see *15.3 Increasing the size of an RDAREA (RDAREA expansion)*.

Do not perform this operation when a sort processing error occurs. Take remedial action based on the error message that was issued.

(2) Use the pdhold command to shut down the RDAREA storing the index

```
pdhold -r RDAREA2
```

This operation is not required if the RDAREA is already shut down.

(3) Use the pdrorg command to create the index in the batch mode

```
pdrorg -k ixmk -t TABLE1 /pdrorg/rorg01
```

Using as input information the index information file that was created when reloading (or data loading) was performed, create the index in the batch mode.

(4) Back up the RDAREA storing the index

Because you created the index in the batch mode in the pre-update acquisition mode (default), back up the RDAREA storing the index (`RDAREA2`). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(5) Use the pdrels command to release the RDAREA from shutdown status

```
pdrels -r RDAREA2
```

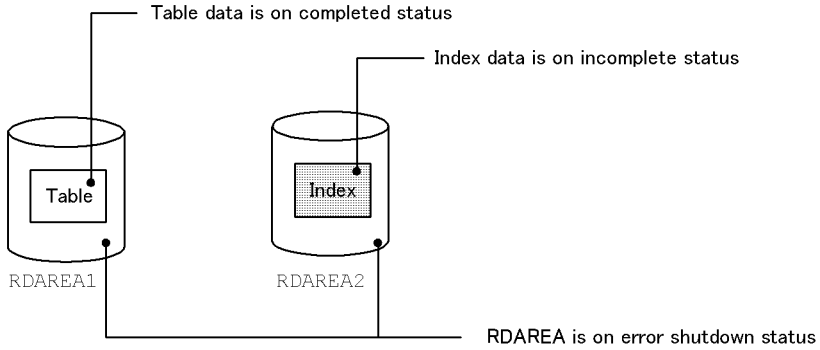
It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

14.6.2 Example of recovery when reloading (data loading) was performed in the no-log mode (when the RDAREA storing the index contains no other tables or indexes)

Example

- A table for which an index is defined was reloaded. However, because the size of the RDAREA storing the index was too small at this time (or a sort processing error occurred), an error occurs during batch index creation. When a sort processing error occurs, the `KFPL15062-E` message is issued.
- Data was loaded in the addition mode into a table for which a plug-in index is defined. However, because the size of the RDAREA storing the index was too small at this time, an error occurs during batch plug-in index creation.

In such a case, the data is in the following status:



(1) Check the message

If you are loading data, check if the `KFPL00703-I` message was issued. If you are

reloading, check if the KFPL00714-I or KFPL00734-I message was issued. If one of these messages has been issued, the table data is in completed status. In this case, use the pdrels command to release the RDAREA storing the table data (RDAREA1) from shutdown status.

```
pdrels -r RDAREA1
```

(2) Use the pdmod command to re-initialize the RDAREA storing the index

Use the pdmod command to re-initialize the RDAREA storing the index (RDAREA2). For details about re-initializing RDAREAs, see *15.4 Increasing the size of an RDAREA or modifying its attributes (RDAREA reinitialization)*.

(3) Use the pdmod command to increase the size of the RDAREA storing the index

If an error occurs because the RDAREA is too small, use the pdmod command to increase the size of the RDAREA storing the index (RDAREA2). For details about increasing the size of RDAREAs, see *15.3 Increasing the size of an RDAREA (RDAREA expansion)*.

Do not perform this operation when a sort processing error occurs. Take remedial action based on the error message that was issued.

(4) Use the pdhold command to shut down the RDAREA storing the index

```
pdhold -r RDAREA2
```

This operation is not required if the RDAREA is already shut down.

(5) Use the pdrorg command to re-create the index

```
pdrorg -k ixrc -t TABLE1 /pdrorg/rorg02
```

(6) Back up the RDAREA storing the index

Because you re-created the index in the pre-update acquisition mode (default), back up the RDAREA storing the index (RDAREA2). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(7) Use the pdrels command to release the RDAREA from shutdown status

```
pdrels -r RDAREA2
```

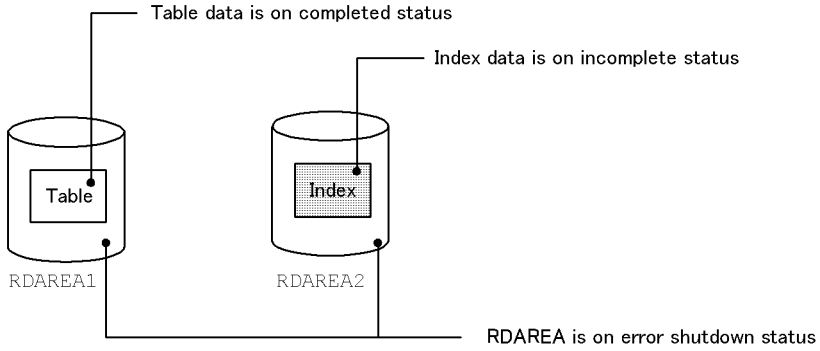
It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

14.6.3 Example of recovery when reloading (data loading) was executed in the no-log mode (when the RDAREA storing the index contains other tables or indexes)

Example

- A table for which an index is defined was reloaded. However, because the size of the RDAREA storing the index was too small at this time (or a sort processing error occurred), an error occurs during batch index creation. When a sort processing error occurs, the `KFPL15062-E` message is issued.
- Data was loaded in the addition mode into a table in which a plug-in index is defined. However, because the size of the RDAREA storing the index was too small at this time, an error occurs during batch plug-in index creation.

In such a case, the data is in the following status:



(1) Check the message

If you are loading data, check if the `KFPL00703-I` message was issued. If you are reloading, check if the `KFPL00714-I` or `KFPL00734-I` message was issued. If one of these messages has been issued, the table data is in completed status. In this case, use the `pdrels` command to release the RDAREA storing the table data (`RDAREA1`) from shutdown status.

```
pdrels -r RDAREA1
```

(2) Use the pdrstr command to recover the RDAREA storing the index

Recover the RDAREA storing the index (`RDAREA2`). For details about recovering

RDAREAs, see *19. Database Recovery Procedures*.

(3) Use the pdmod command to increase the size of the RDAREA storing the index

If an error occurs because the RDAREA is too small, use the `pdmod` command to increase the size of the RDAREA storing the index (`RDAREA2`). For details about increasing the size of RDAREAs, see *15.3 Increasing the size of an RDAREA (RDAREA expansion)*.

Do not perform this operation when a sort processing error occurs. Take remedial action based on the error message that was issued.

(4) Use the pdhold command to shut down the RDAREA storing the index

```
pdhold -r RDAREA2
```

This operation is not required if the RDAREA is already shut down.

(5) Use the pdrorg command to re-create the index

```
pdrorg -k ixrc -t TABLE1 /pdrorg/rorg03
```

(6) Back up the RDAREA storing the index

Because you re-created the index in the pre-update acquisition mode (default), back up the RDAREA storing the index (`RDAREA2`). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(7) Use the pdrels command to release the RDAREA from shutdown status

```
pdrels -r RDAREA2
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

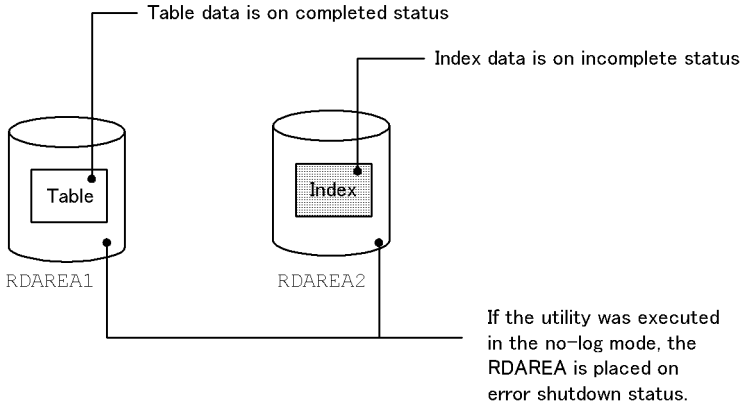
14.6.4 Example of recovery in the event of an error on the disk that contains the index storage RDAREA

Example

- Reload processing was executed on a table for which an index is defined. An error occurred during batch index creation because of an error on the disk that contains the index storage RDAREA.

- Data load processing was executed in the addition mode on a table for which a plug-in index is defined. An error occurred during batch creation of plug-in index because of an error on the disk that contains the index storage RDAREA.

In such a case, the data is in the following status:



(1) Check the message

If you are loading data, check if the `KFPL00703-I` message was issued. If you are reloading, check if the `KFPL00714-I` or `KFPL00734-I` message was issued. If one of these messages has been issued, the table data is in completed status. In this case, use the `pdrels` command to release the RDAREA storing the table data (`RDAREA1`) from shutdown status. This operation is not required if reloading (or data loading) was performed in the no-log mode.

```
pdrels -r RDAREA1
```

(2) Take action to remedy the disk error

Perform troubleshooting to locate and remedy the disk error.

(3) Use the pdhold command to shut down the RDAREA storing the index

```
pdhold -r RDAREA2
```

This operation is required only if reloading (or data loading) was performed in the log acquisition mode or the pre-update log acquisition mode.

(4) Use the pdrstr command to recover the RDAREA storing the index

Recover the RDAREA storing the index (RDAREA2). For details about recovering RDAREAs, see *19. Database Recovery Procedures*.

Do not release the RDAREA from shutdown even after it has been recovered.

(5) Use the pdrorg command to re-create the index

```
pdrorg -k ixrc -t TABLE1 /pdrorg/rorg05
```

(6) Back up the RDAREA storing the index

Because you re-created the index in the pre-update acquisition mode (default), back up the RDAREA storing the index (RDAREA2). For details about backing up RDAREAs, see *6.4.6 Example 6 (Backing up RDAREAs)*.

(7) Use the pdrels command to release the RDAREA from shutdown status

```
pdrels -r RDAREA2
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

14.7 Delayed batch creation of a plug-in index

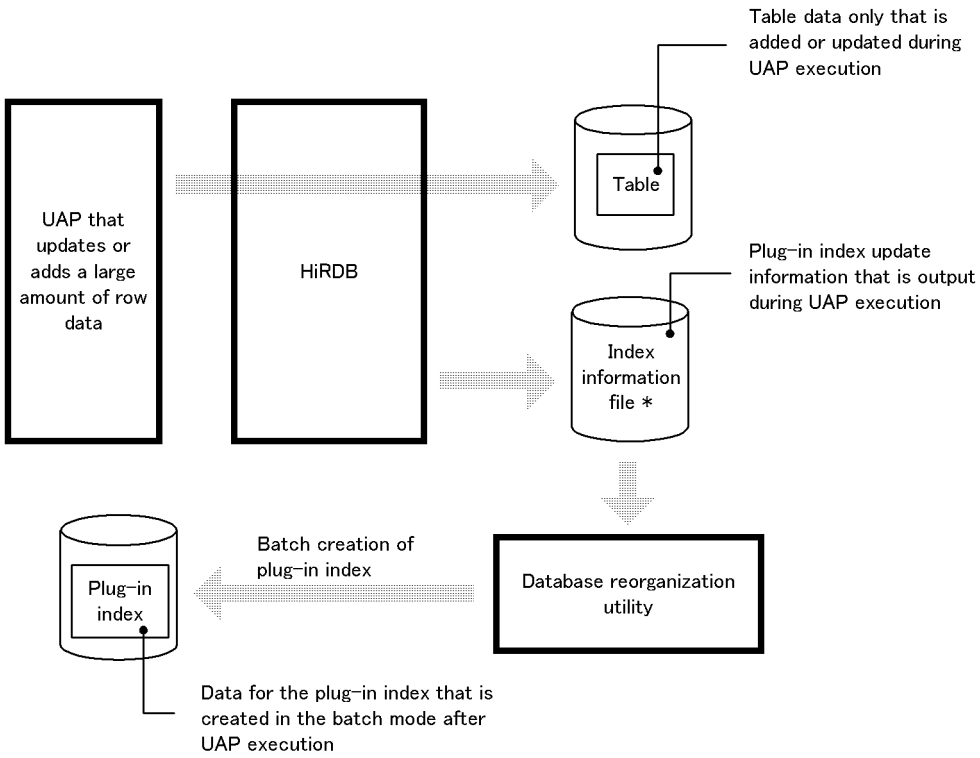
Executor: HiRDB administrator

14.7.1 Delayed batch creation of a plug-in index

When a large amount of row data is added to (or updated in) a table for which a plug-in index is defined, key values are added to the plug-in index. This may have an adverse effect on the performance of plug-in index key value addition processing.

Instead, when row data is added, plug-in index data addition processing can be skipped, and the database reorganization utility can be used later to execute plug-in index data addition processing in the batch mode. This is called *delayed batch creation of a plug-in index*. Figure 14-5 provides an overview of delayed batch creation of a plug-in index.

Figure 14-5: Delayed batch creation of a plug-in index



* When the system switchover facility is being used, create the index information file in a HiRDB file system area (character special file) on the shared disk.

(1) Requirements

The plug-in that is being used must support delayed batch creation of plug-in index. This facility cannot be used if it is not supported by the plug-in.

At the time this manual was issued, only the following plug-in supports this facility:

- HiRDB Text Search Plug-in

(2) Advantages

Because data addition processing for the plug-in index can be skipped, the UAP execution time for adding or updating a large amount of row data (table data creation time) is reduced.

(3) Scope of application

This facility can be applied only to row addition or update processing. In other words, it is applicable to the following SQL statements:

- `INSERT` statement
- `UPDATE` statement

This facility is not applicable to row deletion processing (`DELETE` statement).

(4) Criteria

- This facility is applied to batch processing that involves addition or updating of a large amount of data in a table for which an eligible plug-in index is defined.
- When the HiRDB Text Search Plug-in is used, this facility is effective for batch processing that involves addition or updating of a large amount of data in a table for which the plug-in index defined. However, if the plug-in index is to be updated and then the contents are to be referenced during online processing, the differential index facility that is provided by HiRDB Text Search Plug-in is effective.

(5) Limitations

The following limitations apply to delayed batch creation of plug-in index:

(a) The plug-in index is not usable for retrieving an added or updated row until batch creation of the plug-in index is completed

Once the UAP has executed, conformity between the table's data and the plug-in index's data is lost until batch creation of plug-in index has been completed.*

If the plug-in index is used to retrieve an added or updated row during this period of nonconformity, an invalid result will probably be returned. For this reason, it is important that the plug-in index not be used to retrieve added or updated rows until batch creation of plug-in index is completed.

Valid processing results can be obtained as long as the plug-in index is not used. Also, the plug-in index can still be used to retrieve a row that was not added or updated by the UAP.

* When data is added, the plug-in index's data remains in the same status it was in before the UAP executed. When data is updated, the plug-in index's data is placed in deleted status until it is updated.

Thus, for an added or updated row, there is a loss of conformity between its table data and its plug-in index data.

(b) An ordinary UAP cannot update the plug-in index until batch creation of the plug-in index is completed

The plug-in index cannot be updated by an ordinary UAP (UAP for which delayed batch creation of plug-in index is not specified) until batch creation of plug-in index is completed. However, the `PURGE TABLE` statement can be executed.

(c) UAPs cannot be executed concurrently on the same table

The index information files are placed in the lock mode (`EX`). Therefore, UAPs that use this facility cannot be executed concurrently on the same table.

(d) No more data loading can be executed until batch creation of the plug-in index is completed

Data loading cannot be performed until batch creation of the plug-in index is completed. However, if the table is row-partitioned, you can use the delayed batch creation facility to perform UAP-based processing and to load additional data into RDAREAs in which no data is stored.

14.7.2 Environment setup

The environment setup procedure depends on whether the index information file is to be created in a regular file or in a HiRDB file system area. It is usually created in a regular file, but it must be created in a HiRDB file system area when the system switchover facility is being used.

(1) Creating the index information file under a regular file (usual procedure)

(a) Create a directory for the index information files

A directory for the index information files must be created. The index information files are created in this directory.

Hint:

- Index information files become very large. Create this directory on a disk that has sufficient free space. For details about the size of index information files, see *14.7.4(1) Index information files*.
- To avoid duplication of file names, no other files (other than index information files) should be placed in this directory.
- Because HiRDB creates and deletes files in this directory, the HiRDB administrator must have write privilege for this directory.

(b) Specify the pd_plugin_ixmk_dir operand

Correct the HiRDB system definition. First, terminate HiRDB normally.

HiRDB/Single Server

The name of the directory created in (a) is specified in the `pd_plugin_ixmk_dir` operand of the single server definition.

HiRDB/Parallel Server

The name of the directory created in (a) is specified in the `pd_plugin_ixmk_dir` operand in the back-end server definition for the back-end server that contains the table that is to be processed. If the table is row-partitioned and stored in multiple servers, the `pd_plugin_ixmk_dir` operand must be specified in the back-end server definition of each such back-end server.

(c) Specify PDPLGIXMK=YES in the client environment definition

`PDPLGIXMK=YES` must be specified in the client environment definition (for details of the `PDPLGIXMK` operand, see the manual *HiRDB Version 8 UAP Development Guide*).

(2) Creating the index information file in a HiRDB file system area (when the system switchover facility is being used)

(a) Create a HiRDB file system area for the index information file

```
pdfmkfs -n 50 -l 256 -k UTL -e 60000 /hd001/ixdir
```

Explanation

- `-l` option specifies the number of RDAREAs (number of RDAREAs in the server) that will store the plug-in indexes to be updated by this facility.
- `-k` option specifies `UTL`.

- `-e` option specifies `60000`.
- The HiRDB file system area created here must be used only for index information files to be used in delayed batch creation of plug-in indexes.
- Create the HiRDB file system area in a character special file on the shared disk.
- Index information files become very large. Create this directory on a disk that has sufficient free space. For details about the size of index information files, see *14.7.4(1) Index information files*.
- To avoid duplication of file names, do not store any files other than index information files in this HiRDB file system area.
- Creation and deletion of files in the HiRDB file system area created here are performed by HiRDB. Therefore, the HiRDB administrator's write privilege is required for this HiRDB file system area.

(b) Specify the pd_plugin_ixmk_dir operand

Correct the HiRDB system definition. First, terminate HiRDB normally.

HiRDB/Single Server

Specify the name of the HiRDB file system area created in step (a) in the `pd_plugin_ixmk_dir` operand of the single server definition; specify this name as an absolute path name.

HiRDB/Parallel Server

Specify the name of the HiRDB file system area created in step (a) in the `pd_plugin_ixmk_dir` operand of the back-end server definition for the back-end server in which the target table is located; specify this name as an absolute path name. If the table is row-partitioned between servers, specify the `pd_plugin_ixmk_dir` operand in the back-end server definition of each such back-end server.

(c) Add operands to the client environment definition

Specify the following operands in the client environment definition (for details of these operands, see the manual *HiRDB Version 8 UAP Development Guide*):

Operand	Explanation
`PDPLGIXMK`	Specifies whether or not delayed batch creation of plug-in indexes can be performed; specify `YES`.
`PDPLGPFSZ`	Specifies the initial size of index information files for delayed batch creation.
`PDPLGPFSZEXP`	Specifies the size increase increment for index information files for delayed batch creation. When an index becomes full, its size is increased by the value specified here.

Guideline for estimating the number of updatable plug-in indexes

If the PDPLGPFSZ and PDPLGPFSZEXP operands are omitted, 8192 KB is used as the default for both operands. The number of updatable plug-in indexes in such a case will be as follows:

Plug-in used	Without incremental increases	With incremental increases
HiRDB Text Search Plug-in	215,000 indexes	5,160,000 indexes

Specification example of the PDPLGPFSZ and PDPLGPFSZEXP operands

Shown below is a specification example of the above operands when multiple plug-in indexes are updated.

Example

- File size of plug-in index 1: 10240 KB
- File size of plug-in index 2: 81920 KB

Example of specifying the PDPLGPFSZ and PDPLGPFSZEXP operands for this case:

```
PDPLGPFSZ=10240
PDPLGPFSZEXP=10240
```

With these specifications, the file size of plug-in index 2 is increased seven times, and file areas can be used without any waste of space. Note that each file can be increased in size up to 23 times.

14.7.3 Procedure during UAP execution

The following is the procedure from UAP execution to delayed batch creation of plug-in index:

1. Execute the UAP. When the UAP executes, information about the plug-in index is output to the index information file. The name of the created index information file is provided in the KFPH25100-I message.
2. Use the pdhold command to shut down the plug-in index storage RDAREA:

   ```
   pdhold -r RDAREA1
   ```

3. Use the pdrorg command to specify creation of the plug-in index in the batch mode (the index information file created in step 1 is used as the input):

   ```
   pdrorg -k ixmk -t TABLE1 /pdrorg/rorg06
   ```

4. Use the pdrels command to release the plug-in index storage RDAREA from

shutdown status:

```
pdrels -r RDAREA1
```

14.7.4 Notes

(1) Index information files

(a) Number of files created

An index information file is created for each RDAREA that stores the plug-in index. Therefore, the number of index information files created equals the number of RDAREAs that store the plug-in index updated by the UAP.

(b) Names of created files

The names to be assigned to index information files are determined by the index identifiers and RDAREA names (according to the rules shown below). These index information file names are specified in the control statements file of the database reorganization utility (`pdrorg` command).

Condition	Index information file name
When all the following conditions are satisfied: • *index-identifier-length* + *RDAREA-name-length* < 30 bytes • The index identifier and RDAREA name are character strings, as explained in the note below.	*index-identifier* . *RDAREA-name*
When all the following conditions are satisfied: • *index-identifier-length* + *RDAREA-name-length* ≧ 30 bytes • The index identifier and RDAREA name are character strings, as explained in the note below.	*index-ID* . *RDAREA-ID* . *leading-portion-of-index-identifier**
The index identifier and RDAREA name consist of character strings other than as explained in the note below.	*index-ID* . *RDAREA-ID*

Note

A character string consists of the characters A to Z, a to z, 0 to 9, period (.), underline (_), and @. It is recommended that index identifiers and RDAREA names be assigned according to this rule.

* First 12 bytes of the index identifier

When the inner replica facility is being used

When a replica RDAREA contains a plug-in index, the index information file name is determined according to the following conditions:

Condition	Index information file name
The name of the directory containing the index information file is specified in the `pd_plugin_ixmk_dir` operand.	*index-identifier*.*original-RDAREA-name*.*RDAREA-generation-number*
Both of the following conditions are satisfied: • *index-identifier-length* = *RDAREA-name-length* < 27 bytes • The index identifier and the RDAREA name are character strings, as explained in the note below.	
The following condition is satisfied: • *index-identifier-length* + *RDAREA-name-length* ≥ 27 bytes • The index identifier and the RDAREA name are character strings, as explained in the note below.	*index-ID*.*RDAREA-ID*.*leading-portion-of-index-identifier**.*RDAREA-generation-number*
The index identifier or the RDAREA name consists of a character string other than that explained in the note below.	*index-ID*.*RDAREA-ID*.*RDAREA-generation-number*

Note

A character string consists of the characters A to Z, a to z, 0 to 9, period (.), underline (_), and @. It is recommended that index identifiers and RDAREA names be assigned according to this rule.

* First 9 bytes of the index identifier

(c) Size of a created file

The size of an index information file will be large. The size of one file can be determined from the following formula:

$(12 + E) \times (A + B + C \times D) + 1024$ (bytes)

A: Number of row data items added (number of data items updated by `INSERT` statement)

B: Number of row data items updated (number of data items updated by `UPDATE` statement)

C: Number of UAPs specifying delayed batch creation of plug-in index

D: Number of `COMMIT` statements issued in the UAP

E: Depends on the type of plug-in; for the HiRDB Text Search Plug-in, the value is 27.

(d) File deletion

Index information files are deleted automatically by HiRDB when they are no longer needed. The files are deleted at the following times:

- When batch creation of plug-in index is completed successfully
- When re-creation of plug-in index has been executed

The files are deleted automatically only when this processing is executed; otherwise, such files must be deleted manually by the user.

(2) Backup

Once the UAP executes, conformity is lost between the table's data and the plug-in index's data until batch creation of plug-in index is completed. Thus, a backup of the data should not be made during this time.

(3) Deadlock on index information files

If more than one UAP specifying delayed batch creation of plug-in index is executed on the same row-partitioned table, deadlock may occur on the index information files.

(4) Stored procedures and stored functions

Delayed batch creation of plug-in index is not supported for stored procedures and stored functions created under HiRDB Version 5.0 05-02 or earlier. If an attempt is made in such a case to execute delayed batch creation of plug-in index by specifying `PDPLGIXMK=YES`, the `KFPA11537-E` message will be output and an error will result. When this happens, one of the following actions should be taken:

- **To execute delayed batch creation of a plug-in index**

  Execute `ALTER ROUTINE` or `ALTER PROCEDURE` on the stored procedures and stored functions that update the plug-in index.

- **To not execute delayed batch creation of a plug-in index**

  Execute the UAP with `PDPLGIXMK=NO` specified.

14.7.5 Error handling procedures

(1) Error in an index information file

If an error occurs in an index information file after the UAP has executed but before batch creation of plug-in indexes is completed, delayed batch creation of plug-in indexes cannot be executed. In such a case, the plug-in index must be created by one of the following methods:

- Use the database reorganization utility to re-create the plug-in index
- Execute `DROP INDEX`, then execute `CREATE INDEX` to create the plug-in index

If there is a large number of data items, either method will require a considerable amount of time because the amount of plug-in index data is equivalent to the number of data items added by the UAP plus the number of data items contained initially in the table.

Therefore, a directory with sufficient disk space must be provided in order to avoid a space shortage for the index information files.

(2) Error during UAP execution

Rollback executed

Appropriate action to correct the error must be taken, then the UAP can be re-executed.

Rollback not executed

If the UAP was executed in the no-log mode, rollback is not executed. In such a case, the database must be restored with the database recovery utility, then the UAP can be re-executed.

(3) Error during batch creation of a plug-in index

Appropriate action to correct the error must be taken, then the database reorganization utility can be executed to create the plug-in index in the batch mode.

Chapter

15. Handling RDAREAs

This chapter explains the procedures for handling RDAREAs.

It contains the following sections:

15.1 RDAREA space shortage

Executor: HiRDB administrator

If a table is subjected to many data additions and deletions, a shortage of space may occur in the table's storage RDAREA. HiRDB issues the following messages for a table whose data retrieval efficiency or storage efficiency has been reduced:

- KFPA12300-I
- KFPH00211-I
- KFPH00212-I

(1) When to increase the size of an RDAREA

The size of an existing RDAREA should be increased or a new RDAREA should be created in the following cases:

- When any of these messages is output frequently for a table stored in a particular RDAREA
- When any of these messages is output during or immediately after table reorganization
- When any of these messages is output during or immediately after index reorganization

In such a case, the database structure modification utility (pdmod command) should be used to increase the size of the RDAREA or to create a new RDAREA.

(2) Using the pddbls command to obtain the number of unused segments in the RDAREA

The pddbls command can be used determine how much unused space remains in RDAREA (number of unused segments).

Example

Use the pddbls command to check the number of unused segments in the RDAREA named RDAREA1:

```
pddbls -r RDAREA1 -a

STATE OF RDAREA
    RDAREA        ID       STATUS                TYPE
                           OPNMODE
    RDAREA1       4        OPEN                  USER
                           INITIAL
    SEGMENT   75 / 700
```

Explanation

SEGMENT displays the number of unused segments, followed by the total number of segments. The number of unused segments is the remaining available space in the RDAREA.

In this example, the number of unused segments is 75, and the total number of segments is 700. When the number of unused segments is low, the RDAREA should be expanded.

(3) Space shortage in a list RDAREA

If execution of an ASSIGN LIST statement results in the number of created lists exceeding the number of lists that can be created in the RDAREA (KFPA11812-E message is output), you must use the database structure modification utility to add a list RDAREA.

If execution of an ASSIGN LIST statement results in a shortage of RDAREA pages (KFPA11756-E message is output), you must use one of the following methods to increase the RDAREA size:

- RDAREA addition
- RDAREA expansion
- RDAREA reinitialization

When the size of a list RDAREA is expanded by means of RDAREA reinitialization, the lists stored in that list RDAREA are deleted and must be re-created with the ASSIGN LIST statement.

(4) Using the facility for RDAREA automatic extension

When an RDAREA space shortage occurs, the RDAREA can be extended automatically if space is available in the HiRDB file system area; this is called *RDAREA automatic extension* (for details on RDAREA automatic extension, see *15.7 RDAREA automatic extension*).

15.2 Creating an RDAREA (RDAREA addition)

Executor: HiRDB administrator

A new RDAREA can be added by using the `create rdarea` statement of the database structure modification utility (`pdmod` command).

15.2.1 Before adding an RDAREA

(1) RDAREAs that can be added

The following RDAREAs can be added:

- Data dictionary LOB RDAREA
- Data dictionary RDAREAs*
- User RDAREAs
- User LOB RDAREAs
- List RDAREAs

* These are the data dictionary RDAREAs for storing dictionary tables used for routine management.

(2) When to add an RDAREA

A new RDAREA should be added in the following cases:

1. When a new table or index is to be created in a new RDAREA, rather than in an existing RDAREA
2. When free space is to be created in an existing RDAREA by moving part of a table or index to a new RDAREA
3. When a narrowed search is to be performed or when the number of created lists exceeds the maximum (add a list RDAREA)

(3) Notes

When adding an RDAREA, keep in mind the values of the following operands. If either of these operand values is exceeded, RDAREAs cannot be added.

- Maximum number of RDAREAs, as specified in the `pd_max_rdarea` operand
- Maximum number of HiRDB files comprising RDAREAs, as specified in the `pd_max_file_no` operand

(4) Allocating a global buffer

If you plan to use the RDAREA that was added right away, you must allocate a global

buffer by means of one of the methods described below.

(a) Allocating a global buffer when the pdmod command is executed

With this method, you allocate global buffers using the `globalbuffer` operand of the `create rdarea` statement in the `pdmod` command. However, this method can be used only to allocate existing global buffers (global buffers that have been specified with the `-r` or `-o` option of the `pdbuffer` operand). This method cannot be used in any of the following cases (in these cases, you must use one of the methods described in (b) or (c) below):

- When the page length of the RDAREA being added is equal to or greater than the buffer size of the global buffer
- When an index or LOB global buffer is being allocated
- When a newly added global buffer is being allocated

Global buffers that are allocated using this method become invalid when HiRDB terminates normally or by means of a planned termination, so you will have to change the `pdbuffer` operand specification while HiRDB is stopped. Note also that you may be able to use the system reconfiguration command (`pdchgconf` command) to change the value specified in the `pdbuffer` operand while HiRDB is running.

Reference note:

- If you do not change the value specified in the `pdbuffer` operand, the RDAREA that was added will be allocated a global buffer specified with the `-o` option when HiRDB is started subsequently.
- If you restart HiRDB, a global buffer specified with the `globalbuffer` operand remains allocated.

(b) Allocating a global buffer with the pdbufmod command

With this method, you allocate global buffers with the `pdbufmod` command. This method can also be used to allocate newly added global buffers. However, in order to add global buffers using this method, both of the following conditions must be satisfied:

- HiRDB Advanced High Availability is installed.
- The value `Y` is specified in the `pd_dbbuff_modify` operand.

For details about allocating global buffers with the `pdbufmod` command, see *9.3 Adding, modifying, and deleting global buffers while HiRDB is running (dynamic updating of global buffers)*.

(c) Allocating a global buffer with the system reconfiguration command (pdchgconf command)

With this method, you allocate global buffers by using the system reconfiguration command (`pdchgconf` command) to change the value specified in the `pdbuffer` operand. You can also use this method to allocate newly added global buffers, in addition to existing global buffers. Note that HiRDB Advanced High Availability must be installed in order to use this command.

For details about changing HiRDB system definitions using the system reconfiguration command, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

(5) Rules for using stored procedures and stored functions

1. When a data dictionary LOB RDAREA is added, a data dictionary table for routine management is also created. If there is not enough space in the data dictionary RDAREA, it must first be expanded.
2. When a data dictionary LOB RDAREA is added, data dictionary RDAREAs for storing the data dictionary table for routine management can also be added at the same time.
3. When a data dictionary LOB RDAREA is added, two RDAREAs must be added at the same time.

15.2.2 Example

This example adds a user RDAREA named `RDAREA1`.

Procedure

1. Use the `pdfmkfs` command to create a HiRDB file system area for the RDAREA. This operation is not required if you add the RDAREA to an existing HiRDB file system area.
2. Create a control statements file for the `pdmod` command.
3. Use the `pdmod` command to add the RDAREA.
4. Use the `pdcopy` command to back up data.
5. Use the `pdbufmod` command to allocate a global buffer.
6. Update the `pdbuffer` operand.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfmkfs command to create a HiRDB file system area for the RDAREA

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area01
```

Explanation

A HiRDB file system area (`/rdarea/area01`) is created for the RDAREA.

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

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

`-k`: Specifies `DB` to create a HiRDB file system area for RDAREAs.

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

`/rdarea/area01`: Specifies the name of the HiRDB file system area to be created.

(2) Create the control statements file for the pdmod command

A control statements file (`/pdmod/create01`) that contains the `pdmod` command's `create rdarea` statement is created. The following are the contents of the control statements file:

```
create rdarea RDAREA1                              1
   globalbuffer gbuf01                             2
   for user used by PUBLIC                         3
   server name bes1                                4
   page 4096 characters                            5
   storage control segment 10 pages                6
   file name "/rdarea/area01/file01"               7
      initial 1000 segments;                       8
```

Explanation

1. Specifies a name for the RDAREA to be created (`RDAREA1`).
2. Specifies the global buffer (`gbuf01`) to be allocated for `RDAREA1`. Because a global buffer specified with this operand will not be allocated when HiRDB is started subsequently, you will have to change the value specified in the `pdbuffer` operand. Note that if you allocate a global buffer in step (5) below, you need not specify this operand.
3. Specifies that `RDAREA1` is to be a public RDAREA.
4. Specifies the name of the server to which the RDAREA is to be added; this option is specified only in the case of a HiRDB/Parallel Server.

5. Specifies the page length.
6. Specifies the segment size.
7. Specifies the HiRDB file that is to constitute the RDAREA; `/rdarea/rdarea01` is the HiRDB file system area created in step (1).
8. Specifies the number of HiRDB file segments.

(3) Use the pdmod command to add the RDAREA

```
pdmod -a /pdmod/create01
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command created in step (2).

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, the contents of the master directory RDAREA and the data dictionary RDAREA are updated. Therefore, all RDAREAs are backed up here.

`-b`: Specifies the name of the backup file.

`-p`: Specifies the output destination of the `pdcopy` command's processing results listing.

(5) Use the pdbufmod command to allocate a global buffer

This step adds a new global buffer (`gbuf01`) and allocates it to `RDAREA1`.

```
pdbufmod -k add -a gbuf01 -r RDAREA1 -n 1000
```

Explanation

`-k add`: Specifies that a global buffer is to be added.

`-a`: Specifies the name of the global buffer being added.

-r: Specifies the RDAREA for which the global buffer is being allocated.

-n: Specifies the sector count of the global buffer.

Note, however, to use the pdbufmod command, both of the following conditions must be satisfied:

- HiRDB Advanced High Availability is installed.
- The value Y is specified in the pd_dbbuff_modify operand.

(6) Update the pdbuffer operand

The global buffer allocated in this procedure becomes invalid if HiRDB is terminated normally or through a planned termination. Therefore, change the pdbuffer operand specification while HiRDB is stopped. An example of the pdbuffer operand specification follows:

```
                              :
pdbuffer -a gbuf01 -r RDAREA1,RDAREA2,RDAREA3 -n 1000
                              :
```

Explanation

The added RDAREA (RDAREA1) is assigned to a global buffer (gbuf01).

Note that you can use the system reconfiguration command (pdchgconf command) to change the pdbuffer operand specification while HiRDB is running. However, to use the system reconfiguration command, you must have HiRDB Advanced High Availability. For details about changing HiRDB system definitions using the system reconfiguration command, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

15.3 Increasing the size of an RDAREA (RDAREA expansion)

Executor: HiRDB administrator

The `expand rdarea` statement of the database structure modification utility (`pdmod` command) can be used to increase (expand) the size of an RDAREA. RDAREA expansion means increasing the size of an RDAREA while retaining the data stored in it. An RDAREA is expanded by adding HiRDB files to the existing RDAREA.

15.3.1 Before expanding an RDAREA

(1) RDAREAs that can be expanded

The following RDAREAs can be expanded:

- User RDAREAs
- User LOB RDAREAs
- Master directory RDAREA
- Data directory RDAREA
- Data dictionary RDAREAs
- Data dictionary LOB RDAREAs
- Registry RDAREA
- Registry LOB RDAREA
- List RDAREAs

(2) When to expand an RDAREA

RDAREA expansion should be performed under the following circumstances:

1. The number of rows in a table has increased to the point that no more data can be stored.
2. RDAREA reinitialization cannot be used to increase the size of the RDAREA (for details on RDAREA reinitialization, see *15.4 Increasing the size of an RDAREA or modifying its attributes (RDAREA reinitialization)*).
3. The RDAREA can contain multiple HiRDB files.

(3) Notes

1. If you increase the number of HiRDB files, keep in mind the value in the `pd_max_file_no` operand. If the number of all HiRDB files in the RDAREAs exceeds the value in this operand, RDAREAs cannot be added.
2. You must handle the RDAREA's status as expanded below:

- If the RDAREA is in error shutdown status, eliminate the cause of the error shutdown, and then use the pdrels command to release the error shutdown status.
- If the RDAREA is in closed status, use the pdopen command to place it in open status.

15.3.2 Example

This example expands a user RDAREA named RDAREA1.

Procedure

1. Use the pdfmkfs command to create a HiRDB file system area for the RDAREA. Perform this option only if the HiRDB file system area of the RDAREA you plan to expand does not have enough free space.
2. Create a control statements file for the pdmod command.
3. Use the pdmod command to expand the RDAREA.
4. Use the pdcopy command to back up data.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfmkfs command to create a HiRDB file system area for the RDAREA

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area11
```

Explanation

A HiRDB file system area (/rdarea/area11) is created for the RDAREA.

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

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

-k: Specifies DB to create a HiRDB file system area for RDAREAs.

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

/rdarea/area11: Specifies a name for the HiRDB file system area to be created.

(2) Create the control statements file for the pdmod command

A control statements file (/pdmod/expand01) that contains the pdmod command's expand rdarea statement is created. The following are the contents of the control statements file:

```
expand rdarea RDAREA1                                  1
   file name "/rdarea/area11/file01"                   2
      initial 1000 segments;                           3
```

Explanation

1. Specifies the RDAREA (RDAREA1) to be expanded.
2. Specifies the HiRDB file to be added; /rdarea/rdarea11 is the HiRDB file system area created in step (1).
3. Specifies the number of segments for the HiRDB file.

(3) Use pdmod command to expand the RDAREA

```
pdmod -a /pdmod/expand01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (2).

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02
-p /pdcopy/list02
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, the contents of the master directory RDAREA and the data dictionary RDAREA are updated. Therefore, all RDAREAs are backed up here.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

15.4 Increasing the size of an RDAREA or modifying its attributes (RDAREA reinitialization)

Executor: HiRDB administrator

The `initialize rdarea` statement of the database structure modification utility (`pdmod` command) can be used to increase (expand) the size of an RDAREA or to modify its attributes (reinitialize it). RDAREA reinitialization means erasing all data in an RDAREA in order to increase its size or to modify its attributes (e.g., to increase the size of an HiRDB file constituting the RDAREA or to modify the segment size).

By reinitializing an RDAREA, you can change the HiRDB files that constitute the RDAREA and change the disk layout for RDAREAs. For the procedure, see *15.4.10 Example 9 (changing the disk layout for RDAREAs)*.

Note that the master directory RDAREA, data dictionary RDAREAs, and data directory RDAREA cannot be reinitialized. To change the disk layout for these RDAREAs, you must use the database initialization utility to initialize the database again. For details about how to migrate a database, see *12. Migrating Resources Between Systems*.

15.4.1 Before reinitializing an RDAREA

(1) RDAREAs that can be reinitialized

The following RDAREAs can be reinitialized:

- User RDAREAs
- User LOB RDAREAs
- Data dictionary LOB RDAREA (for storing objects only)
- Registry RDAREA
- Registry LOB RDAREA
- List RDAREAs

(2) When to reinitialize an RDAREA

RDAREA reinitialization should be performed under the following circumstances:

1. The number of rows in a table has increased to the point that no more data can be stored.
2. The HiRDB file structure needs to be modified (number of files, size of a HiRDB file).
3. The HiRDB file organization (HiRDB file name) is to be changed.

(3) Notes

1. Back up the RDAREA being reinitialized before and after reinitialization. If an error occurs in the RDAREA after it has been initialized, it cannot be recovered from a backup made before it was reinitialized.
2. Because all the data is erased from an RDAREA when it is reinitialized, it is important to unload the data in advance with the database reorganization utility (`pdrorg` command).
3. An RDAREA to be reinitialized should be placed in shutdown and closed status with the `pdhold` command.
4. If you increase the number of HiRDB files, pay attention to the value of the `pd_max_file_no` operand. If the number of all HiRDB files in RDAREAs exceeds the value of this operand, RDAREAs cannot be added.
5. When a user LOB RDAREA is reinitialized, data for the LOB column structure base table remains unchanged. A LOB column is treated as data with a length of 0.
6. If a data dictionary LOB RDAREA for storing stored objects is reinitialized, all SQL objects must be re-created with `ALTER PROCEDURE ALL`.
7. When a list RDAREA is reinitialized, all lists in that list RDAREA are deleted. Therefore, to execute a narrowed search, re-create the lists with the `ASSIGN LIST` statement.
8. The database reorganization utility (`pdrorg` command) cannot be executed for a list RDAREA.
9. An RDAREA containing a falsification prevented table cannot be reinitialized.
10. When a data dictionary LOB RDAREA (for storing objects) is reinitialized, the `SQL_DIV_COLUMN` data dictionary table should be referenced to check the name of the data dictionary LOB RDAREA. The following is an example of this referencing:

    Example

```
SELECT RDAREA_NAME FROM MASTER.SQL_DIV_COLUMN
   WHERE TABLE_SCHEMA='HiRDB'
     AND TABLE_NAME='SQL_ROUTINES'
     AND COLUMN_NAME='ROUTINE_OBJECT'
```

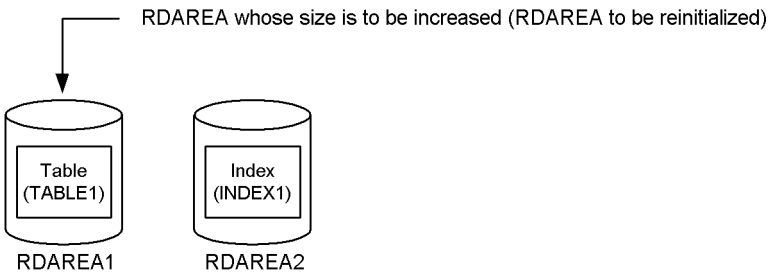
(4) Initializing a registry RDAREA or registry LOB RDAREA

- When a registry RDAREA or registry LOB RDAREA is reinitialized, the registry information used by plug-ins must be reregistered.
- When a registry LOB RDAREA is reinitialized, the registry RDAREA must also be reinitialized.

15.4.2 Example 1 (index is defined)

In this example, a user RDAREA (RDAREA1) is reinitialized and, at the same time, the capacity of the RDAREA is increased.

- One table (TABLE1) is stored in RDAREA1.
- An index (INDEX1) is defined for TABLE1; this index is stored in a different user RDAREA (RDAREA2).



Note: When RDAREA1 is reinitialized, the data for INDEX1 is deleted from RDAREA2.

Procedure

1. Use the pdfstatfs command to determine if the HiRDB file system area has free space.
2. Prepare a HiRDB file system area.
3. Use the pdhold command to shut down RDAREA1.
4. Use the pdcopy command to back up data.
5. Create a control statements file for the pdrorg command.
6. Use the pdrorg command to unload the data from TABLE1.
7. Use the pdclose command to close RDAREA1.
8. Create a control statements file for the pdmod command.
9. Use the pdmod command to reinitialize (increase the size of) RDAREA1.
10. Use the pdopen command to open RDAREA1.
11. Create a control statements file for the pdrorg command.
12. Use the pdrorg command to reload data for TABLE1.
13. Use the pdcopy command to back up data.
14. Use the pdrels command to release RDAREA1 from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the

explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system area has free space

```
pdfstatfs /rdarea/area01
```

(2) Prepare a HiRDB file system area

Assume you determined in step (1) that the HiRDB files system area has no free space. In order to increase the size of the RDAREA when you reinitialize it, you must use one of the following methods to prepare a HiRDB file system area:

1. Allocate a new HiRDB file system area that is larger than the existing HiRDB file system area.
2. Allocate a new HiRDB file system area in addition to the existing HiRDB file system area.
3. Expand the existing HiRDB file system area.

This example uses method 1 to prepare a HiRDB file system area.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area02
```

(3) Use the pdhold command to shut down RDAREA1

```
pdhold -r RDAREA1
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown explained in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdrorg command

A control statements file containing the pdrorg command's unload statement (/pdrorg/unld01) is created. The following are the contents of the control statements file:

```
unload /unld/unldfile
```

Explanation

The name of the unload file is specified.

(6) Use the pdrorg command to unload the data from TABLE1

```
pdrorg -k unld -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies unld in order to unload.

-t: Specifies the name of the table to be unloaded.

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command created in step (5).

(7) Use the pdclose command to close RDAREA1

```
pdclose -r RDAREA1
```

(8) Create the control statements file for the pdmod command

A control statements file that contains the pdmod command's initialize rdarea statement (/pdmod/init01) is created. The following are the contents of the control statements file:

```
initialize rdarea RDAREA1                              1
   with reconstruction                                 2
   file name "/rdarea/area02/file01"                   3
      initial 3000 segments;                           4
```

Explanation

The newly created HiRDB file system area is allocated for RDAREA1.

1. Specifies the RDAREA (RDAREA1) to be reinitialized.
2. Because the file structure is being changed from what it was before reinitialization, specify with reconstruction.
3. Specifies the HiRDB file that is to constitute the RDAREA.
4. Specifies the number of segments for the HiRDB file.

(9) Use the pdmod command to reinitialize (increase the size of) RDAREA1

```
pdmod -a /pdmod/init01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (8).

(10) Use the pdopen command to open RDAREA1

```
pdopen -r RDAREA1
```

(11) Create the control statements file for the pdrorg command

A control statements file containing the pdrorg command's unload, index, and sort statements (/pdrorg/reld01) is created. The following are the contents of the control statements file:

```
unload /unld/unldfile                          1
index INDEX1 /unld/index_inf                   2
sort /tmp/sortwork/,8192                       3
```

Explanation

1. Specifies the name of the unload file.
2. Specifies the index identifier (INDEX1) and the name of the index information file (/unld/index_inf).
3. Specifies the name of the sort work directory.

(12) Use the pdrorg command to reload data for TABLE1

```
pdrorg -k reld -t TABLE1 /pdrorg/reld01
```

Explanation

To create the index (`INDEX1`) at the same time, the `-i` option is omitted in order to execute in the batch index creation mode.

`-k`: Specifies `reld` in order to reload.

`-t`: Specifies the name of the table to be reloaded.

`/pdrorg/reld01`: Specifies the name of the control statements file for the `pdrorg` command created in step (11).

(13) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

(14) Use the pdrels command to release RDAREA1 from shutdown status

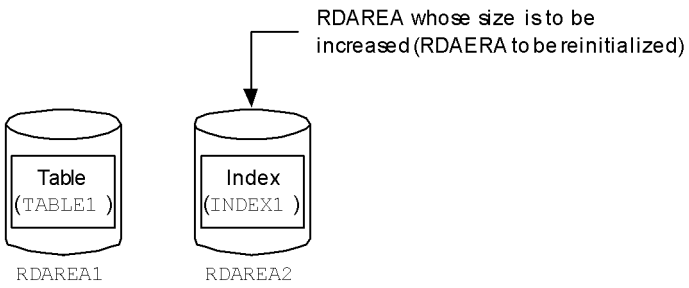
```
pdrels -r RDAREA1
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

15.4.3 Example 2 (index is defined)

In this example, a user RDAREA (`RDAREA2`) is reinitialized and, at the same time, the capacity of the RDAREA is increased.

- One table (`TABLE1`) is stored in `RDAREA1`.
- An index (`INDEX1`) is defined for `TABLE1`; this index is stored in a different user RDAREA (`RDAREA2`).



Note: Even when RDAREA2 is re-initialized, the data for TABLE1 stored in RDAREA1 is not deleted.

Procedure

1. Use the pdfstatfs command to determine if the HiRDB file system area has free space.
2. Prepare a HiRDB file system area.
3. Use the pdhold command to place RDAREA2 in shutdown and closed status.
4. Use the pdcopy command to back up data.
5. Create a control statements file for the pdmod command.
6. Use the pdmod command to reinitialize (increase the size of) RDAREA1.
7. Use the pdopen command to open RDAREA2.
8. Create a control statements file for the pdload command.
9. Use the pdload command to create an index in the batch mode (execute batch index creation without loading any data items).
10. Use the pdcopy command to back up data.
11. Use the pdrels command to release RDAREA2 from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system area has free space

```
pdfstatfs /rdarea/area01
```

(2) Prepare a HiRDB file system area

Assume you determined in step (1) that the HiRDB files system area has no free space. In order to increase the size of the RDAREA when you reinitialize it, you must use one of the following methods to prepare a HiRDB file system area:

1. Allocate a new HiRDB file system area that is larger than the existing HiRDB file system area.
2. Allocate a new HiRDB file system area in addition to the existing HiRDB file system area.
3. Expand the existing HiRDB file system area.

This example uses method 1 to prepare a HiRDB file system area.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area02
```

(3) Use the pdhold command to place RDAREA2 in shutdown and closed status

```
pdhold -r RDAREA2 -c
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdmod command

A control statements file that contains the pdmod command's initialize rdarea statement (/pdmod/init01) is created. The following are the contents of the control

statements file:

```
initialize rdarea RDAREA2                                    1
   with reconstruction                                       2
   file name "/rdarea/area02/file01"                         3
      initial 3000 segments;                                 4
```

Explanation

The newly created HiRDB file system area is allocated for RDAREA2.

1. Specifies the RDAREA to be reinitialized (RDAREA2).
2. Because the file structure is being changed from what it was before reinitialization, specify with reconstruction.
3. Specifies the HiRDB file that is to constitute the RDAREA.
4. Specifies the number of segments for the HiRDB file.

(6) Use the pdmod command to reinitialize (increase the size of) RDAREA2

```
pdmod -a /pdmod/init01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (5).

(7) Use the pdopen command to open RDAREA2

```
pdopen -r RDAREA2
```

(8) Create the control statements file for the pdload command

A control statements file containing the pdload command's source, index, and sort statements (/pdload/load01) is created. The following are the contents of the control statements file:

```
source /load/loadfile error=/tmp/err          1
index INDEX1 /load/index_inf                  2
sort /tmp/sortwork/,8192                      3
```

Explanation

1. Specifies the input data file and error information file.
2. Specifies the index identifier (INDEX1) and the name of the index information file (/load/index_inf).
3. Specifies the name of the sort work directory.

(9) Use the pdload command to create an index in batch mode (execute batch index creation with loading of no data items)

```
pdload TABLE1 /pdload/load01
```

Explanation

The index (INDEX1) is to be created in the batch mode with loading of no data items. The -i option is omitted in order to execute in the batch index creation mode.

TABLE1: Specifies the name of the table for which data loading is to be executed.

/pdrorg/load01: Specifies the name of the control statements file for the pdload command created in step (8).

(10) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

(11) Use the pdrels command to release RDAREA2 from shutdown status

```
pdrels -r RDAREA2
```

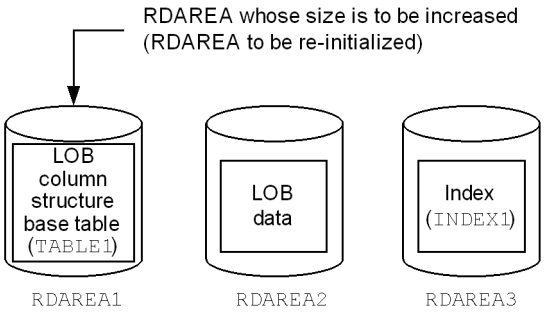
It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

An RDAREA storing a plug-in index can also be reinitialized in the same manner in order to increase its size.

15.4.4 Example 3 (LOB column is defined)

In this example, a user RDAREA (RDAREA1) is reinitialized and, at the same time, the capacity of the RDAREA is increased.

- The LOB column structure base table of a table (TABLE1) is stored in RDAREA1.
- The LOB data is stored in a user LOB RDAREA (RDAREA2).
- An index (INDEX1) is defined for TABLE1; this index is stored in a different user RDAREA (RDAREA3).



Note: When RDAREA1 is re-initialized, LOB data stored in RDAREA2 and index data stored in RDAREA3 are deleted.

Procedure

1. Use the pdfstatfs command to determine if the HiRDB file system area has free space.
2. Prepare a HiRDB file system area.
3. Use the pdhold command to shut down RDAREA1 and RDAREA2.
4. Use the pdcopy command to back up data.
5. Create a control statements file for the pdrorg command.
6. Use the pdrorg command to unload the data from TABLE1.
7. Use the pdclose command to close RDAREA1.
8. Create a control statements file for the pdmod command.
9. Use the pdmod command to reinitialize (increase the size of) RDAREA1.
10. Use the pdopen command to open RDAREA1.
11. Create a control statements file for the pdrorg command.
12. Use the pdrorg command to reload data for TABLE1.
13. Use the pdcopy command to back up data.
14. Use the pdrels command to release RDAREA1 and RDAREA2 from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system area has free space

```
pdfstatfs /rdarea/area01
```

(2) Prepare a HiRDB file system area

Assume you determined in step (1) that the HiRDB files system area has no free space. In order to increase the size of the RDAREA when you reinitialize it, you must use one of the following methods to prepare a HiRDB file system area:

1. Allocate a new HiRDB file system area that is larger than the existing HiRDB file system area.
2. Allocate a new HiRDB file system area in addition to the existing HiRDB file system area.
3. Expand the existing HiRDB file system area.

This example uses method 1 to prepare a HiRDB file system area.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area02
```

(3) Use the pdhold command to shut down RDAREA1 and RDAREA2

```
pdhold -r RDAREA1,RDAREA2
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as

shown in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdrorg command

A control statements file containing the pdrorg command's unload statement (/pdrorg/unld01) is created. The following are the contents of the control statements file:

```
unload /unld/unldfile1
```

Explanation

Specifies the name of the unload file.

(6) Use the pdrorg command to unload the data from TABLE1

```
pdrorg -k unld -j -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies unld in order to unload.

-j: Specifies that the table to be unloaded contains a LOB column.

-t: Specifies the name of the table to be unloaded.

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command created in step (5).

(7) Use the pdclose command to close RDAREA1

```
pdclose -r RDAREA1
```

(8) Create the control statements file for the pdmod command

A control statements file that contains the pdmod command's initialize rdarea statement (/pdmod/init01) is created. The following are the contents of the control statements file:

```
initialize rdarea RDAREA1                              1
   with reconstruction                                 2
   file name "/rdarea/area02/file01"                   3
      initial 3000 segments;                           4
```

Explanation

The newly created HiRDB file system area is allocated for RDAREA1.

1. Specifies the RDAREA (RDAREA1) to be reinitialized.
2. Because the file structure is being changed from what it was before reinitialization, specify with reconstruction.
3. Specifies the HiRDB file that is to constitute the RDAREA.
4. Specifies the number of segments for the HiRDB file.

(9) Use the pdmod command to reinitialize (increase the size of) RDAREA1

```
pdmod -a /pdmod/init01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (8).

(10) Use the pdopen command to open RDAREA1

```
pdopen -r RDAREA1
```

(11) Create the control statements file for the pdrorg command

A control statements file containing the pdrorg command's unload, index and sort statements (/pdrorg/reld01) is created. The following are the contents of the control statements file:

```
unload /unld/unldfile                      1
index INDEX1 /unld/index_inf               2
sort /tmp/sortwork/,8192                   3
```

Explanation

1. Specifies the name of the unload file.

2. Specifies the index identifier (INDEX1) and the name of the index information file (/unld/index_inf).
3. Specifies the name of the sort work directory.

(12) Use the pdrorg command to reload data for TABLE1

```
pdrorg -k reld -j -t TABLE1 /pdrorg/reld01
```

Explanation

To create the index (INDEX1) at the same time, the -i option is omitted in order to execute in the batch index creation mode.

-k: Specifies reld in order to reload.

-j: Specifies that the table to be reloaded contains a LOB column.

-t: Specifies the name of the table to be reloaded.

/pdrorg/reld01: Specifies the name of the control statements file for the pdrorg command created in step (11).

(13) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

(14) Use the pdrels command to release RDAREA1 and RDAREA2 from shutdown status

```
pdrels -r RDAREA1,RDAREA2
```

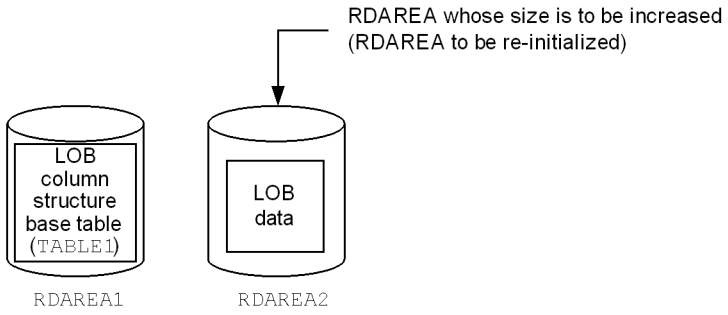
It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

15.4.5 Example 4 (LOB column is defined)

In this example, a user LOB RDAREA (RDAREA2) is reinitialized and, at the same time, the capacity of the RDAREA is increased.

- The LOB column structure base table of a table (TABLE1) is stored in RDAREA1.

- The LOB data is stored in a user LOB RDAREA (RDAREA2).



Note: Even when RDAREA2 is re-initialized, LOB column structure base table data stored in RDAREA1 is not deleted.

Procedure

1. Use the pdfstatfs command to determine if the HiRDB file system area has free space.
2. Prepare a HiRDB file system area.
3. Use the pdhold command to shut down RDAREA1 and RDAREA2.
4. Use the pdcopy command to back up data.
5. Create a control statements file for the pdrorg command.
6. Use the pdrorg command to unload the data from TABLE1.
7. Use the pdclose command to close RDAREA2.
8. Create a control statements file for the pdmod command.
9. Use the pdmod command to reinitialize (increase the size of) RDAREA2.
10. Use the pdopen command to open RDAREA2.
11. Use the pdrorg command to reload data for TABLE1.
12. Use the pdcopy command to back up data.
13. Use the pdrels command to release RDAREA1 and RDAREA2 from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system area has free space

```
pdfstatfs /rdarea/area01
```

(2) Prepare a HiRDB file system area

Assume you determined in step (1) that the HiRDB files system area has no free space. In order to increase the size of the RDAREA when you reinitialize it, you must use one of the following methods to prepare a HiRDB file system area:

1. Allocate a new HiRDB file system area that is larger than the existing HiRDB file system area.
2. Allocate a new HiRDB file system area in addition to the existing HiRDB file system area.
3. Expand the existing HiRDB file system area.

This example uses method 1 to prepare a HiRDB file system area.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area02
```

(3) Use the pdhold command to shut down RDAREA1 and RDAREA2

```
pdhold -r RDAREA1,RDAREA2
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdrorg command

A control statements file containing the pdrorg command's unload statement (/pdrorg/unld01) is created. The following are the contents of the control statements file:

```
unload /unld/unldfile
```

Explanation

Specifies the name of the unload file.

(6) Use the pdrorg command to unload LOB data

```
pdrorg -k unld -j -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies unld in order to unload.

-j: Specifies that the table to be unloaded contains a LOB column.

-t: Specifies the name of the table to be unloaded.

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command created in step (5).

(7) Use the pdclose command to close RDAREA2

```
pdclose -r RDAREA2
```

(8) Create the control statements file for the pdmod command

A control statements file that contains the pdmod command's initialize rdarea statement (/pdmod/init01) is created. The following are the contents of the control statements file:

```
initialize rdarea RDAREA2                              1
   with reconstruction                                 2
   file name "/rdarea/area02/file01"                   3
      initial 3000 segments;                           4
```

Explanation

The newly created HiRDB file system area is allocated for RDAREA2.

1. Specifies the RDAREA to be reinitialized (RDAREA2).
2. Because the file structure is being changed from what it was before reinitialization, specify with reconstruction.
3. Specifies the HiRDB file that is to constitute the RDAREA.
4. Specifies the number of segments for the HiRDB file.

(9) Use the pdmod command to reinitialize (increase the size of) RDAREA2

```
pdmod -a /pdmod/init01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (8).

(10) Use the pdopen command to open RDAREA2

```
pdopen -r RDAREA2
```

(11) Use the pdrorg command to reload data of TABLE1

```
pdrorg -k reld -j -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies reld in order to reload.

-j: Specifies that the table to be reloaded contains a LOB column.

-t: Specifies the name of the table to be reloaded.

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command created in step (5).

(12) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in

6.3 RDAREAs to be backed up together. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

(13) Use the pdrels command to release RDAREA1 and RDAREA2 from shutdown status

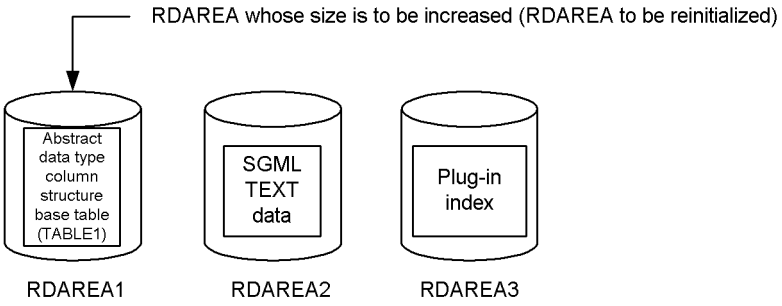
```
pdrels -r RDAREA1,RDAREA2
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

15.4.6 Example 5 (abstract data type is defined)

In this example, a user RDAREA (`RDAREA1`) is reinitialized and, at the same time, the capacity of the RDAREA is increased.

- `RDAREA1` contains the abstract data type column structure base table of a table (`TABLE1`). The abstract data type provided by the plug-in (`SGMLTEXT` type) is defined in `TABLE1`.
- The SGML TEXT data is stored in a user LOB RDAREA (`RDAREA2`).
- The plug-in index is stored in a user LOB RDAREA (`RDAREA3`).



Note: When RDAREA1 is reinitialized, the SGML TEXT data stored in RDAREA2 and the data for the plug-in index stored in RDAREA3 are deleted.

Procedure

1. Use the `pdfstatfs` command to determine if the HiRDB file system area has free space.
2. Prepare a HiRDB file system area.
3. Use the `pdhold` command to shut down `RDAREA1` and `RDAREA2`.

4. Use the pdcopy command to back up data.
5. Create a control statements file for the pdrorg command.
6. Use the pdrorg command to unload the data from TABLE1.
7. Use the pdclose command to close RDAREA1.
8. Create a control statements file for the pdmod command.
9. Use the pdmod command to reinitialize (increase the size of) RDAREA1.
10. Use the pdopen command to open RDAREA1.
11. Create a control statements file for the pdrorg command.
12. Use the pdrorg command to reload data for TABLE1.
13. Use the pdcopy command to back up data.
14. Use the pdrels command to release RDAREA1 and RDAREA2 from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system area has free space

```
pdfstatfs /rdarea/area01
```

(2) Prepare a HiRDB file system area

Assume you determined in step (1) that the HiRDB files system area has no free space. In order to increase the size of the RDAREA when you reinitialize it, you must use one of the following methods to prepare a HiRDB file system area:

1. Allocate a new HiRDB file system area that is larger than the existing HiRDB file system area.
2. Allocate a new HiRDB file system area in addition to the existing HiRDB file system area.
3. Expand the existing HiRDB file system area.

This example uses method 1 to prepare a HiRDB file system area.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area02
```

(3) Use the pdhold command to shut down RDAREA1 and RDAREA2

```
pdhold -r RDAREA1,RDAREA2
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdrorg command

A control statements file (/pdrorg/unld01) is created for the unload statement of the pdrorg command. The following are the contents of the control statements file:

```
unload /unld/unldfile1
```

Explanation

Specifies the name of the unload file.

(6) Use the pdrorg command to unload data from TABLE1

```
pdrorg -k unld -j -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies unld for unloading.

-j: Specifies that one of the following types of tables is to be unloaded:

- Table containing a LOB column
- Table in which is defined an abstract data type with the LOB attribute

`-t`: Specifies the name of the table that is to be unloaded.

`/pdrorg/unld01`: Specifies the name of the control statements file for the `pdrorg` command created in step (5).

(7) Use the pdclose command to close RDAREA1

```
pdclose -r RDAREA1
```

(8) Create the control statements file for the pdmod command

A control statements file (`/pdmod/init01`) is created for the `initialize rdarea` statement of the `pdmod` command. The following are the contents of the control statements file:

```
initialize rdarea RDAREA1                                  1
   with reconstruction                                     2
   file name "/rdarea/area02/file01"                       3
      initial 3000 segments;                               4
```

Explanation

The newly created HiRDB file system area is allocated for `RDAREA1`.

1. Specifies the RDAREA (`RDAREA1`) that is to be reinitialized.
2. Because the file structure is being changed from what it was before reinitialization, specify `with reconstruction`.
3. Specifies the HiRDB file comprising the RDAREA.
4. Specifies the number of HiRDB file segments.

(9) Use the pdmod command to reinitialize (extend the size of) RDAREA1

```
pdmod -a /pdmod/init01
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command created in step (8).

(10) Use the pdopen command to open RDAREA1

```
pdopen -r RDAREA1
```

(11) Create the control statements file for the pdrorg command

A control statements file (`/pdrorg/reld01`) is created for the `unload`, `index`, and `sort` statements of the `pdrorg` command. The following are the contents of the control statements file:

```
unload /unld/unldfile                    1
index INDEX1 /unld/index_inf             2
sort /tmp/sortwork/,8192                 3
```

Explanation

1. Specifies the name of the unload log file.
2. Specifies an index identifier (`INDEX1`) and the name of the index information file (`/unld/index_inf`).
3. Specifies the name of the work directory for sorting.

(12) Use the pdrorg command to reload data into TABLE1

```
pdrorg -k reld -j -t TABLE1 /pdrorg/reld01
```

Explanation

To also create the index (`INDEX1`) at the same time, the `-i` option is omitted and the index is created in the batch mode.

`-k`: Specifies `reld` for reloading.

`-j`: Specifies that one of the following types of tables is to be reloaded:

- Table containing a LOB column
- Table in which is defined an abstract data type with the LOB attribute

`-t`: Specifies the name of the table that is to be reloaded.

`/pdrorg/reld01`: Specifies the name of the control statements file for the `pdrorg` command created in step (11).

(13) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

(14) Use the pdrels command to release shutdown status of RDAREA1 and RDAREA2

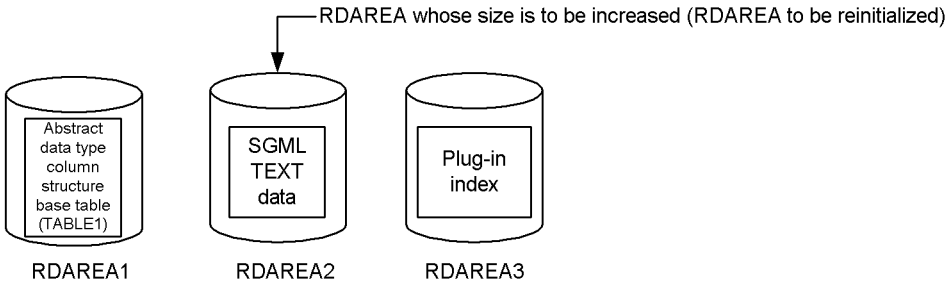
```
pdrels -r RDAREA1,RDAREA2
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

15.4.7 Example 6 (abstract data type is defined)

In this example, a user LOB RDAREA (`RDAREA2`) is reinitialized and, at the same time, the capacity of the RDAREA is increased.

- `RDAREA1` contains the abstract data type column structure base table of a table (`TABLE1`). The abstract data type provided by the plug-in (`SGMLTEXT` type) is defined in `TABLE1`.
- The SGML TEXT data is stored in a user LOB RDAREA (`RDAREA2`).
- The plug-in index is stored in a user LOB RDAREA (`RDAREA3`).



Note: Reinitializing RDAREA2 will not delete data stored in RDAREA1 and RDAREA3.

Procedure

1. Use the `pdfstatfs` command to determine if the HiRDB file system area

has free space.

2. Prepare a HiRDB file system area.
3. Use the pdhold command to shut down RDAREA1 and RDAREA2.
4. Use the pdcopy command to back up data.
5. Create a control statements file for the pdrorg command.
6. Use the pdrorg command to unload the data from TABLE1.
7. Use the pdclose command to close RDAREA2.
8. Create a control statements file for the pdmod command.
9. Use the pdmod command to reinitialize (increase the size of) RDAREA2.
10. Use the pdopen command to open RDAREA2.
11. Create a control statements file for the pdrorg command.
12. Use the pdrorg command to reload data for TABLE1.
13. Use the pdcopy command to back up data.
14. Use the pdrels command to release RDAREA1 and RDAREA2 from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system area has free space

```
pdfstatfs /rdarea/area01
```

(2) Prepare a HiRDB file system area

Assume you determined in step (1) that the HiRDB files system area has no free space. In order to increase the size of the RDAREA when you reinitialize it, you must use one of the following methods to prepare a HiRDB file system area:

1. Allocate a new HiRDB file system area that is larger than the existing HiRDB file system area.
2. Allocate a new HiRDB file system area in addition to the existing HiRDB file system area.
3. Expand the existing HiRDB file system area.

This example uses method 1 to prepare a HiRDB file system area.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area02
```

(3) Use the pdhold command to shut down RDAREA1 and RDAREA2

```
pdhold -r RDAREA1,RDAREA2
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdrorg command

A control statements file (/pdrorg/unld01) is created for the unload statement of the pdrorg command. The following are the contents of the control statements file:

```
unload /unld/unldfile
```

Explanation

Specifies the name of the unload file.

(6) Use the pdrorg command to unload data from TABLE1

```
pdrorg -k unld -j -t TABLE1 /pdrorg/unld01
```

Explanation

`-k`: Specifies `unld` for unloading.

`-j`: Specifies that one of the following types of tables is to be unloaded:

- Table containing a LOB column
- Table in which an abstract data type with the LOB attribute is defined

`-t`: Specifies the name of the table that is to be unloaded.

`/pdrorg/unld01`: Specifies the name of the control statements file for the `pdrorg` command created in step (5).

(7) Use the pdclose command to close RDAREA2

```
pdclose -r RDAREA2
```

(8) Create the control statements file for the pdmod command

A control statements file (`/pdmod/init01`) is created for the `initialize rdarea` statement of the `pdmod` command. The following are the contents of the control statements file:

```
initialize rdarea RDAREA2                              1
   with reconstruction                                 2
   file name "/rdarea/area02/file01"                   3
      initial 3000 segments;                           4
```

Explanation

The newly created HiRDB file system area is allocated for `RDAREA2`.

1. Specifies the RDAREA (`RDAREA2`) that is to be reinitialized.
2. Because the file structure is being changed from what it was before reinitialization, specify `with reconstruction`.
3. Specifies the HiRDB file comprising the RDAREA.
4. Specifies the number of HiRDB file segments.

(9) Use the pdmod command to reinitialize (extend the size of) RDAREA2

```
pdmod -a /pdmod/init01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (8).

(10) Use the pdopen command to open RDAREA2

```
pdopen -r RDAREA2
```

(11) Create the control statements file for the pdrorg command

A control statements file (/pdrorg/reld01) is created for the unload, index, and sort statements of the pdrorg command. The following are the contents of the control statements file:

```
unload /unld/unldfile                        1
index INDEX1 /unld/index_inf                 2
sort /tmp/sortwork/,8192                     3
```

Explanation

1. Specifies the name of the unload log file.
2. Specifies an index identifier (INDEX1) and the name of the index information file (/unld/index_inf).
3. Specifies the name of the work directory for sorting.

(12) Use the pdrorg command to reload data into TABLE1

```
pdrorg -k reld -j -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies reld for reloading.

-j: Specifies that one of the following types of tables is to be reloaded:

- Table containing a LOB column
- Table in which an abstract data type with the LOB attribute is defined

-t: Specifies the name of the table that is to be reloaded.

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command created in step (11).

(13) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

(14) Use the pdrels command to release shutdown status of RDAREA1 and RDAREA2

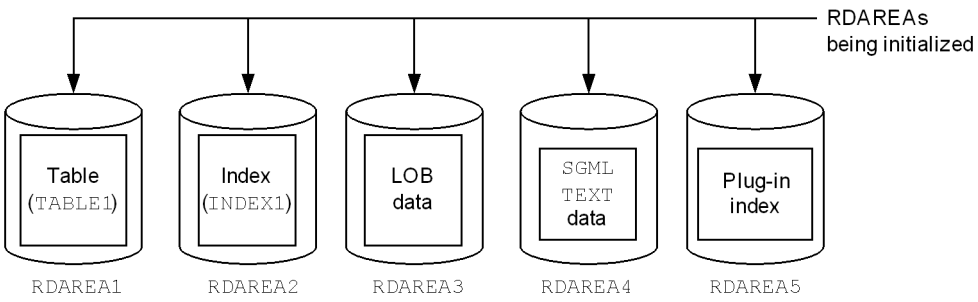
```
pdrels -r RDAREA1,RDAREA2
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

15.4.8 Example 7 (using a UAP, all RDAREAs associated with a table are reinitialized, and data is recovered)

In this example, two user RDAREAs (`RDAREA1` and `RDAREA2`) and three user LOB RDAREAs (`RDAREA3`, `RDAREA4`, and `RDAREA5`) are reinitialized and, at the same time, the capacities of the RDAREAs are increased.

- A table (`TABLE1`) with a LOB column structure base table and an abstract data type column structure base table is stored in `RDAREA1`. A plug-in provided abstract data type (`SGMLTEXT` type) is defined for `TABLE1`.
- An index (`INDEX1`) is defined for `TABLE1`. This index is stored in a user RDAREA (`RDAREA2`).
- LOB data is stored in a user LOB RDAREA (`RDAREA3`).
- `SGMLTEXT` data is stored in a user LOB RDAREA (`RDAREA4`).
- The plug-in index is stored in a user LOB area (`RDAREA5`).



Procedure

1. Use the `pdfstatfs` command to determine if the HiRDB file system areas have free space.
2. Prepare the HiRDB file system areas.
3. Use the `pdhold` command to shut down `RDAREA1` to `RDAREA5`.
4. Use the `pdcopy` command to back up data.
5. Create a control statements file for the `pdrorg` command.
6. Use the `pdrorg` command to unload the data from `TABLE1`.
7. Use the `pdclose` command to close `RDAREA1` to `RDAREA5`.
8. Create a control statements file for the `pdmod` command.
9. Use the `pdmod` command to reinitialize `RDAREA1` to `RDAREA5`.
10. Use the `pdopen` command to open `RDAREA1` to `RDAREA5`.
11. Create a control statements file for the `pdrorg` command.
12. Use the `pdrorg` command to reload data for `TABLE1`.
13. Use the `pdcopy` command to back up data.
14. Use the `pdrels` command to release `RDAREA1` to `RDAREA5` from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system areas have free space

```
pdfstatfs /rdarea/area01
pdfstatfs /rdarea/area02
pdfstatfs /rdarea/area03
pdfstatfs /rdarea/area04
pdfstatfs /rdarea/area05
```

Check all the HiRDB files system areas in the RDAREAs to be initialized.

(2) Prepare the HiRDB file system areas

Assume you determined in step (1) that the HiRDB files system areas have no free space. In order to increase the sizes of the RDAREAs when you reinitialize them, you must use one of the following methods to prepare the HiRDB file system areas:

1. Allocate new HiRDB file system areas that are larger than the existing HiRDB file system areas.
2. Allocate new HiRDB file system areas in addition to the existing HiRDB file system areas.
3. Expand the existing HiRDB file system areas.

This example uses method 1 to prepare the HiRDB file system areas.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area11
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area12
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area13
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area14
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area15
```

(3) Use the pdhold command to shut down RDAREA1 to RDAREA5

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdrorg command

A control statements file (/pdrorg/unld01) is created for the unload statement of the pdrorg command. The following are the contents of the control statements file:

```
unload /unld/unldfile
```

Explanation

Specifies the name of the unload file.

(6) Use the pdrorg command to unload data from TABLE1

```
pdrorg -k unld -j -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies unld for unloading.

-j: Specifies that one of the following types of tables is to be unloaded:

- Table containing a LOB column
- Table in which an abstract data type with the LOB attribute is defined

-t: Specifies the name of the table that is to be unloaded.

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command created in step (5).

(7) Use the pdclose command to close RDAERA1 to RDAREA5

```
pdclose -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

(8) Create the control statements file for the pdmod command

A control statements file (`/pdmod/init01`) is created for the `initialize rdarea` statement of the `pdmod` command. The following are the contents of the control statements file:

```
initialize rdarea RDAREA1                                    1
   with reconstruction                                       2
   file name "/rdarea/area11/file01"                         3
      initial 3000 segments;                                 4
initialize rdarea RDAREA2
   with reconstruction
   file name "/rdarea/area12/file01"
      initial 3000 segments;
initialize rdarea RDAREA3
   with reconstruction
   file name "/rdarea/area13/file01"
      initial 3000 segments;
initialize rdarea RDAREA4
   with reconstruction
   file name "/rdarea/area14/file01"
       initial 3000 segments;
initialize rdarea RDAREA5
   with reconstruction
   file name "/rdarea/area15/file01"
       initial 3000 segments;
```

Explanation

The newly created HiRDB file system areas are allocated for the RDAREAs being reinitialized.

1. Specifies the RDAREAs being initialized.
2. Because the file structure is being changed from what it was before reinitialization, specify `with reconstruction`.
3. Specifies the HiRDB file comprising the RDAREA.
4. Specifies the number of HiRDB file segments.

(9) Use the pdmod command to reinitialize RDAREA1 to RDAREA5

```
pdmod -a /pdmod/init01
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command created in step (8).

Reference note:

When RDAREA1 is reinitialized, the KFPX14255-W, KFPX24231-W, and KFPX24242-W messages are issued, warning of an invalid RDAREA. However, because you are reinitializing RDAREA2 through RDAREA5 at the same time, this does not pose a problem. In addition, when RDAREA2, RDAREA4, and RDAREA5 are being reinitialized, the abstract data type with the LOB attribute is denied access to the index, because the index is incomplete. However, reloading the data in step (12) releases these RDAREAs, so this does not pose a problem either.

(10) Use the pdopen command to open RDAREA1 to RDAREA5

```
pdopen -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

(11) Create the control statements file for the pdrorg command

A control statements file (/pdrorg/reld01) is created for the unload, index, and sort statements of the pdrorg command. The following are the contents of the control statements file:

```
unload /unld/unldfile                    1
index INDEX1 /unld/index_inf             2
sort /tmp/sotwork/,8192                  3
```

Explanation

1. Specifies the name of the unload log file.
2. Specifies an index identifier (INDEX1) and the name of the index information file (/unld/index_inf).
3. Specifies the name of the work directory for sorting.

(12) Use the pdrorg command to reload data into TABLE1

```
pdrorg -k reld -j -t TABLE1 /pdrorg/reld01
```

Explanation

-k: Specifies reld for reloading.

-j: Specifies that one of the following types of tables is to be reloaded:

- Table containing a LOB column

- Table in which an abstract data type with the LOB attribute is defined

`-t`: Specifies the name of the table that is to be reloaded.

`/pdrorg/unld01`: Specifies the name of the control statements file for the `pdrorg` command created in step (11).

(13) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

(14) Use the pdrels command to release RDAREA1 to RDAREA5 from shutdown status

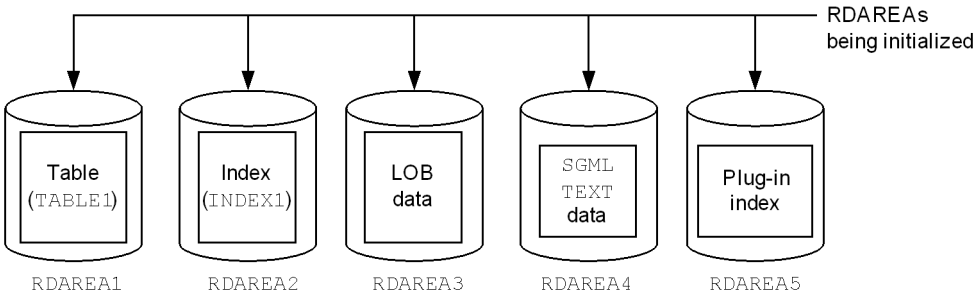
```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

15.4.9 Example 8 (using a UAP, all RDAREAs associated with a table are reinitialized, and data is recovered)

In this example, two user RDAREAs (`RDAREA1` and `RDAREA2`) and three user LOB RDAREAs (`RDAREA3`, `RDAREA4`, and `RDAREA5`) are reinitialized and, at the same time, the capacities of the RDAREAs are increased.

- A table (`TABLE1`) with a LOB column structure base table and an abstract data type column structure base table is stored in `RDAREA1`. A plug-in provided abstract data type (`SGMLTEXT` type) is defined for `TABLE1`.
- An index (`INDEX1`) is defined for `TABLE1`. This index is stored in a user RDAREA (`RDAREA2`).
- LOB data is stored in a user LOB RDAREA (`RDAREA3`).
- `SGMLTEXT` data is stored in a user LOB RDAREA (`RDAREA4`).
- The plug-in index is stored in a user LOB area (`RDAREA5`).



Procedure

1. Use the `pdfstatfs` command to determine if the HiRDB file system areas have free space.
2. Prepare the HiRDB file system areas.
3. Use the `pdhold` command to shut down and close `RDAREA1` to `RDAREA5`.
4. Use the `pdcopy` command to back up data.
5. Create a control statements file for the `pdmod` command.
6. Use the `pdmod` command to reinitialize `RDAREA2` to `RDAREA5`.
7. Use the `pdrels` command to release `RDAREA2` to `RDAREA5` from shutdown status, and to place them in open status.
8. Create a control statements file for the `pdmod` command.
9. Use the `pdmod` command to reinitialize `RDAREA1`.
10. Use the `pdrels` command to release `RDAREA1` from shutdown status, and to place it in open status.
11. Execute the UAP.
12. Use the `pdcopy` command to back up data.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfstatfs command to determine if the HiRDB file system areas have free space

```
pdfstatfs /rdarea/area01
pdfstatfs /rdarea/area02
pdfstatfs /rdarea/area03
pdfstatfs /rdarea/area04
pdfstatfs /rdarea/area05
```

Check all the HiRDB files system areas containing the RDAREAs to be initialized.

(2) Prepare the HiRDB file system areas

Assume you determined in step (1) that the HiRDB files system areas have no free space. In order to increase the sizes of the RDAREAs when you reinitialize them, you must use one of the following methods to prepare the HiRDB file system areas:

1. Allocate new HiRDB file system areas that are larger than the existing HiRDB file system areas.
2. Allocate new HiRDB file system areas in addition to the existing HiRDB file system areas.
3. Expand the existing HiRDB file system areas.

This example uses method 1 to prepare a HiRDB file system areas.

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area11
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area12
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area13
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area14
pdfmkfs -n 100 -l 10 -k DB -i /rdarea/area15
```

(3) Use the pdhold command to shut down and close RDAREA1 to RDAREA5

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5 -c
```

(4) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(5) Create the control statements file for the pdmod command

A control statements file (/pdmod/init01) is created for the initialize rdarea statement of the pdmod command. The following are the contents of the control statements file:

```
initialize rdarea RDAREA2                                    1
   with reconstruction                                       2
   file name "/rdarea/area12/files01"                        3
      initial 3000 segments;                                 4
initialize rdarea RDAREA3
   with reconstruction
   file name "/rdarea/area13/files01"
      initial 3000 segments;
initialize rdarea RDAREA4
   with reconstruction
   file name "/rdarea/area14/files01"
       initial 3000 segments;
initialize rdarea RDAREA5
   with reconstruction
   file name "/rdarea/area15/files01"
       initial 3000 segments;
```

Explanation

The newly created HiRDB file system area is allocated for the RDAREAs being reinitialized.

1. Specifies the RDAREAs being initialized.
2. Because the file structure is being changed from what it was before reinitialization, specify with reconstruction.
3. Specifies the HiRDB file comprising the RDAREA.
4. Specifies the number of HiRDB file segments.

(6) Use the pdmod command to reinitialize RDAREA2 to RDAREA5

```
pdmod -a /pdmod/init01
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command created in step (5).

Reference note:

When `RDAREA2`, `RDAREA4`, and `RDAREA5` are reinitialized, the abstract data type with the LOB attribute is denied access to the index, because the index is incomplete. However, reinitializing `RDAREA1` in step (9) releases these RDAREAs, so this does not pose a problem.

(7) Use the pdrels command to release RDAREA2 to RDAREA5 from shutdown status and place them in open status

```
pdrels -r RDAREA2,RDAREA3,RDAREA4,RDAREA5 -o
```

(8) Create the control statements file for the pdmod command

A control statements file (`/pdmod/init02`) is created for the `initialize rdarea` statement of the `pdmod` command. The following are the contents of the control statements file:

```
initialize rdarea RDAREA1                              1
   with reconstruction                                 2
   file name "/rdarea/area11/file01"                   3
      initial 3000 segments;                           4
```

Explanation

The newly created HiRDB file system area is allocated for `RDAREA1`.

1. Specifies the RDAREA (`RDAREA2`) that is to be reinitialized.
2. Because the file structure is being changed from what it was before reinitialization, specify `with reconstruction`.
3. Specifies the HiRDB file comprising the RDAREA.
4. Specifies the number of HiRDB file segments.

(9) Use the pdmod command to reinitialize RDAREA1

```
pdmod -a /pdmod/init02
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command created in step (8).

(10) Use the pdrels command to release RDAREA1 from shutdown status and place it in open status

```
pdrels -r RDAREA1 -o
```

(11) Execute the UAP

Execute the UAP that inserts data into `TABLE1` to recover the data.

(12) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs as shown in section 6.3. In this example, all RDAREAs are backed up.

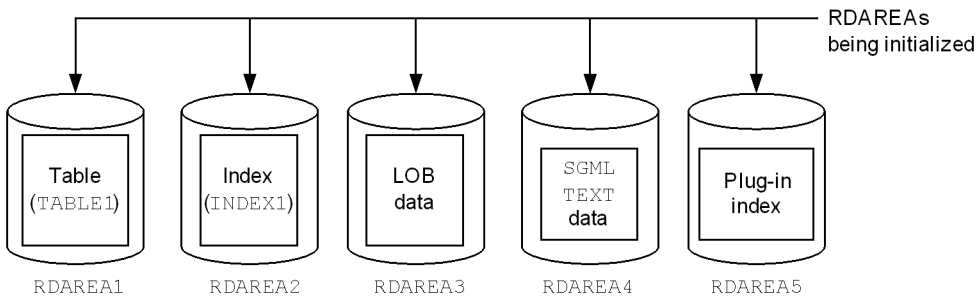
It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

15.4.10 Example 9 (changing the disk layout for RDAREAs)

In this example, two user RDAREAs (`RDAREA1` and `RDAREA2`) and three user LOB RDAREAs (`RDAREA3`, `RDAREA4`, and `RDAREA5`) are initialized, and the disk layout for the RDAREAs is changed from `/rdarea` to `/rdarea2`.

- `RDAREA1` stores the LOB column structure base table and abstract data type column structure base table of the table `TABLE1`. An abstract data type (`SGMLTEXT` type) provided by a plug-in is defined for `TABLE1`.
- An index (`INDEX1`) is defined for `TABLE1`. This index is stored in a user RDAREA (`RDAREA2`).
- LOB data is stored in a user LOB RDAREA (`RDAREA3`).
- SGMLTEXT data is stored in a user LOB RDAREA (`RDAREA4`).

- The plug-in index is stored in a user LOB RDAREA (RDAREA5).



Procedure

1. Prepare a HiRDB file system area.
2. Use the pdhold command to shut down RDAREA1 through RDAREA5.
3. Use the pdcopy command to back up data.
4. Create a control statements file for the pdrorg command.
5. Use the pdrorg command to unload the data from TABLE1.
6. Use the pdclose command to close RDAREA1 through RDAREA5.
7. Create a control statements file for the pdmod command.
8. Use the pdmod command to reinitialize RDAREA1 through RDAREA5.
9. Use the pdopen command to open RDAREA1 through RDAREA5.
10. Create a control statements file for the pdrorg command.
11. Use the pdrorg command to reload data for TABLE1.
12. Use the pdcopy command to back up data.
13. Use the pdrels command to release RDAREA1 through RDAREA5 from shutdown status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Prepare a HiRDB file system area

Prepare a HiRDB file system area on the disk where the new RDAREAs are to be placed (/rdarea2).

```
pdfmkfs -n 100 -l 10 -k DB -i /rdarea2/area11
pdfmkfs -n 100 -l 10 -k DB -i /rdarea2/area12
pdfmkfs -n 100 -l 10 -k DB -i /rdarea2/area13
pdfmkfs -n 100 -l 10 -k DB -i /rdarea2/area14
pdfmkfs -n 100 -l 10 -k DB -i /rdarea2/area15
```

For the `-n` and `-l` options, specify a value that is equal to or greater than the value for the HiRDB file system area that was created on the disk (`/rdarea`) before the change.

(2) Use the pdhold command to shut down RDAREA1 through RDAREA5

```
pdhold -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

(3) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-`m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up. When an RDAREA is reinitialized, other RDAREAs are also updated, as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs, as shown in Section 6.3. In this example, all RDAREAs are backed up.

-`b`: Specifies the name of the backup file.

-`p`: Specifies the output destination of the `pdcopy` command's processing results listing.

(4) Create a control statements file for the pdrorg command

A control statements file containing the `pdrorg` command's unload statement (`/pdrorg/unld01`) is created. The following are the contents of the control statements file:

```
unload /unld/unldfile
```

Explanation

The name of the unload file is specified.

(5) Use the pdrorg command to unload the data from TABLE1

```
pdrorg -k unld -j -t TABLE1 /pdrorg/unld01
```

Explanation

-k: Specifies unld in order to unload.

-j: Specifies that the following tables are to be unloaded:

- Table with a LOB column
- Table for which an abstract data type with the LOB attribute has been defined

-t: Specifies the name of the table to be unloaded.

/pdrorg/unld01: Specifies the name of the control statements file for the pdrorg command created in step (4).

(6) Use the pdclose command to close RDAREA1 through RDAREA5

```
pdclose -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

(7) Create a control statements file for the pdmod command

A control statements file that contains the pdmod command's initialize rdarea statement (/pdmod/init01) is created. The following are the contents of the control statements file:

```
initialize rdarea RDAREA1                                    1
   with reconstruction                                       2
   file name "/rdarea2/area11/file01"                        3
      initial 3000 segments;                                 4
initialize rdarea RDAREA2
   with reconstruction
   file name "/rdarea2/area12/file01"
      initial 3000 segments;
initialize rdarea RDAREA3
   with reconstruction
   file name "/rdarea2/area13/file01"
      initial 3000 segments;
initialize rdarea RDAREA4
   with reconstruction
   file name "/rdarea2/area14/file01"
       initial 3000 segments;
initialize rdarea RDAREA5
   with reconstruction
   file name "/rdarea2/area15/file01"
       initial 3000 segments;
```

Explanation

The newly created HiRDB file system area is allocated for the RDAREAs that are to be reinitialized.

1. Specifies an RDAREA to be reinitialized.
2. Because the file structure is being changed from what it was before reinitialization, specify `with reconstruction`.
3. Specifies the HiRDB file that is to constitute the RDAREA. This example specifies the new disk on which the RDAREA is to be placed (`/rdarea2`).
4. Specifies the number of segments for the HiRDB file. This value must be equal to or greater than the value for the HiRDB file that had been created on the disk before the change (`/rdarea`).

(8) Use the pdmod command to reinitialize RDAREA1 through RDAREA5

```
pdmod -a /pdmod/init01
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command created in step (7).

Reference note:

When `RDAREA1` is reinitialized, the `KFPX14255-W`, `KFPX24231-W`, and `KFPX24242-W` messages are issued, warning that there is an invalid RDAREA. However, because you are reinitializing `RDAREA2` through `RDAREA5` at the same time, this does not pose a problem. In addition, when `RDAREA2`, `RDAREA4`, and `RDAREA5` are being reinitialized, the abstract data type with the LOB attribute is denied access to the index, because the index is incomplete. However, reloading the data in step (11) releases these RDAREAs, so this does not pose a problem either.

(9) Use the pdopen command to open RDAREA1 through RDAREA5

```
pdopen -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

(10) Create a control statements file for the pdrorg command

A control statements file (`/pdrorg/reld01`) is created for the `unload`, `index`, and `sort` statements of the `pdrorg` command. The following are the contents of the control statements file:

```
unload /unld/unldfile                    1
index INDEX1 /unld/index_inf             2
sort /tmp/sotwork/,8192                  3
```

Explanation

1. Specifies the name of the unload file.
2. Specifies an index identifier (`INDEX1`) and the name of the index information file (`/unld/index_inf`).
3. Specifies the name of the work directory for sorting.

(11) Use the pdrorg command to reload data for TABLE1

```
pdrorg -k reld -j -t TABLE1 /pdrorg/reld01
```

Explanation

`-k`: Specifies `reld` in order to reload.

`-j`: Specifies that the following tables are to be reloaded:

- Table with a LOB column
- Table for which an abstract data type with the LOB attribute has been defined

`-t`: Specifies the name of the table to be reloaded.

`/pdrorg/reld01`: Specifies the name of the control statements file for the `pdrorg` command created in step (10).

(12) Use the pdcopy command to back up data

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

When an RDAREA is reinitialized, other RDAREAs are also updated, as explained in *6.3 RDAREAs to be backed up together*. Therefore, you must also back up other RDAREAs, as shown in Section 6.3. In this example, all RDAREAs are backed up.

(13) Use the pdrels command to release RDAREA1 through RDAREA5 from shutdown status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3,RDAREA4,RDAREA5
```

After the command has executed, you should check whether or not the execution results are correct. For details about how to check the command execution results, see the manual *HiRDB Version 8 Command Reference*.

15.5 Modifying an RDAREA opening trigger attribute (RDAREA modification)

Executor: HiRDB administrator

The RDAREA opening trigger attribute can be changed with the `alter rdarea` statement of the database structure modification utility (`pdmod` command).

15.5.1 Before changing the RDAREA opening trigger attribute

(1) RDAREA opening trigger attribute

The RDAREA opening trigger attribute refers to the time at which RDAREA open processing is executed by HiRDB.

Normally, RDAREA open processing is executed during HiRDB startup. However, as the number of RDAREAs increases, the amount of time required for HiRDB startup processing also increases. Instead, RDAREAs can be opened when they are accessed, rather than being opened during HiRDB startup processing. Thus, the RDAREA opening trigger attribute is changed. There are three triggers for opening an RDAREA, as shown in Table 15-1.

Table 15-1: RDAREA opening trigger attributes

Trigger	Initial status	Opening time	Closing time	Advantages	Disadvantages
`INITIAL`	Open	• During HiRDB startup • During `pdopen` command execution	During `pdclose` command execution	High-speed execution from the first SQL	More time is required to start the system.
`DEFER`	Closed	• First time RDAREA is accessed • During `pdopen` command execution	During `pdclose` command execution	• High-speed system startup • High-speed processing of SQLs after the first access	More time is required to access the RDAREA for the first time.
`SCHEDULE`	Closed	• First time RDAREA is accessed within a transaction • During `pdopen` command execution	• During transaction termination • During `pdclose` command execution	• High-speed system startup • Fewer open files (for DVD-RAM library devices)	More time is required to access the RDAREA for the first time by each transaction.

(2) Criteria

Initially, the RDAREA opening trigger is `INITIAL`. We recommend you consider changing the RDAREA opening trigger in the following cases:

- HiRDB startup processing takes a long time because there is a large number of RDAREAs.
- A DVD-RAM library device is being used.

Table 15-2 shows the operating procedure appropriate to each trigger.

Table 15-2: Operating procedure appropriate to each trigger

Trigger	Appropriate operating procedure
`INITIAL`	HiRDB file system area is opened and the RDAREA information is made resident in memory during system startup. When an RDAREA is accessed for the first time, it is also opened in that process. In this case, the RDAREA information is not re-created; therefore, high-speed operation can be achieved from the first SQL. The initial status of RDAREAs is to be opened during system startup. This status does not change unless an operation command is entered, except for a status change to error shutdown. *This trigger should be used when no special operating procedure is employed.* When this trigger is used, a closed RDAREA cannot be accessed.
`DEFER`	HiRDB file system area is opened and the RDAREA information is made resident in memory when an RDAREA is accessed for the first time, not during system start. High-speed operation can be achieved for the subsequent accesses, because no processing is required on the HiRDB file system area other than open processing. The RDAREAs' initial status during system startup is closed. Each RDAREA is opened when it is accessed for the first time. The status of RDAREAs does not change thereafter unless an operation command is entered, except for a status change to error shutdown. Specify this attribute when you wish to avoid situations in which many HiRDB file system areas are open concurrently or you wish to reduce the time required for HiRDB startup. When HiRDB is restarted, any RDAREAs to be recovered are opened during recovery processing. When this trigger is used, a closed RDAREA can be accessed.

Trigger	Appropriate operating procedure
SCHEDULE	HiRDB file system area is opened and the RDAREA information is made resident in memory when an RDAREA is accessed for the first time within a transaction, without opening it during system startup. The HiRDB file system area is closed when the transaction terminates. The transaction processing workload increases because the open processing and all subsequent processing is executed the first time an RDAREA is accessed by each transaction. The RDAREAs' initial status during system startup is closed. An RDAREA is open only while a transaction that accesses it is being processed. When the transaction terminates, all the RDAREAs opened by it are closed. When the pdopen command is entered, an RDAREA can be kept in open status until it is placed in shutdown and closed status. Other operation commands can also be used to change the RDAREA status. When an error is detected, the RDAREA is placed in error shutdown status. Specify this attribute when you wish to avoid situations in which many HiRDB file system areas are open concurrently or you wish to reduce the time required for HiRDB startup when a DVD-RAM library device is being used. When HiRDB is restarted, any RDAREAs to be recovered are opened during recovery processing, and they are closed when recovery processing finishes. When this trigger is used, a closed RDAREA can be accessed.

(3) RDAREAs whose opening trigger attribute can be changed

The opening trigger attribute can be changed for the following RDAREAs:

- User RDAREAs
- User LOB RDAREAs
- List RDAREAs

(4) Notes

1. Before an RDAREA's opening trigger attribute is changed, it must be placed in shutdown and closed status with the pdhold -c command.
2. Before an RDAREA whose opening trigger attribute has been changed can be used, it must be released from shutdown status and placed in open status by the pdrels -o command.
3. The same opening trigger attribute should be set for all RDAREAs in a HiRDB file system area.
4. RDAREAs are not open while the standby system unit to which the rapid system switchover facility switches is in standby status. In addition, because the purpose of this facility is to minimize system switchover time, only those RDAREAs that are needed for recovery when a system is switched over are opened, while the other RDAREAs remain closed. Therefore, the RDAREA opening trigger on a standby system is not the INITIAL attribute. RDAREAs set with the INITIAL attribute are changed to the DEFER attribute.

5. Because the purpose of the standbyless system switchover facility is to minimize system switchover time, this facility opens only those RDAREAs that are needed for recovery when a system is switched over, while the other RDAREAs remain closed. Therefore, the opening trigger of the RDAREAs on the alternate BES or the alternate portion is as follows:

   • When system switchover occurs, the opening trigger for RDAREAs in the alternate portion is the `SCHEDULE` attribute.

   • When returning the system to the normal BES after having recovered from the error, the opening trigger for RDAREAs set with the `INITIAL` or `DEFER` attribute is the `DEFER` attribute. The opening trigger for RDAREAs set with the `SCHEDULE` attribute remains unchanged.

6. Table 15-3 indicates the accessibility of UAP RDAREAs according to the open attribute.

Table 15-3: Accessibility of UAP RDAREAs according to the open attribute

Opening trigger for RDAREA	Status of RDAREA		RDAREA accessible?
`INITIAL`	Not shutdown	Open	Yes
		Closed	No
	Command shutdown	Open	No
		Closed	No
	Reference-possible hold	Open	Some[1]
		Closed	No
	Reference-possible backup hold	Open	Some[1]
	Updatable backup hold	Open	Yes
	Error shutdown	Open	No
		Closed	No
	No-log hold	Open	Some[2]
		Closed	No
	Initialization hold	Open	No[3]
		Closed	No[3]

Opening trigger for RDAREA	Status of RDAREA		RDAREA accessible?
	Online reorganization hold	Open	Some[4]
		Closed	Some[4]
DEFER or SCHEDULE	Not shutdown	Open	Yes
		Closed	Yes
	Command shutdown	Open	No
		Closed	No
	Reference-possible hold	Open	Some[1]
		Closed	Some[1]
	Reference-possible backup hold	Open	Some[1]
		Closed	No
	Updatable backup hold	Open	No
		Closed	Yes
	Error shutdown	Open	No
		Closed	No
	No-log hold	Open	Some[2]
		Closed	Some[2]
	Initialization hold	Open	No[3]
		Closed	No[3]
	Online reorganization hold	Open	Some[4]
		Closed	Some[4]

Legend:

Yes: Accessible

Some: Some SQL code accessible

No: Not accessible

[1] Only reference SQL can be accessed.

[2] Only PURGE TABLE can be executed.

[3] Placed in lock-release wait status

[4] Other than the following, only SQL statements that access a current RDAREA while in the log acquisition mode can be executed:

- CREATE TABLE
- CREATE INDEX
- DROP TABLE
- DROP INDEX
- DROP SCHEMA
- ALTER TABLE
- PURGE TABLE
- LOCK

15.5.2 Example

This example changes the opening trigger attribute for user RDAREAs (RDAREA1 to RDAREA3) from INITIAL to SCHEDULE; it also changes the opening trigger attribute for all user RDAREAs other than RDAREA1 to RDAREA3 from INITIAL to DEFER. RDAREA1 to RDAREA3 are located in the same HiRDB file system area.

Procedure

1. Terminate HiRDB normally.
2. Modify the system common definitions.
3. Start HiRDB normally.
4. Use the pdhold command to place the RDAREAs in shutdown and closed status to change their opening trigger attributes.
5. Create a control statements file for the pdmod command.
6. Use the pdmod command to change the RDAREAs' opening trigger.
7. Use the pdrels command to release the shutdown status of RDAREAs whose opening trigger attributes were changed, and then place them in open status.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Terminate HiRDB normally

```
pdstop
```

If you were to use the system reconfiguration command (pdchgconf command), you would not need to restart HiRDB normally in this step, because this command enables you to modify HiRDB system definitions while HiRDB is running. Note that HiRDB Advanced High Availability must be installed in order to use this command. For details about modifying HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

(2) Modify the system common definition

The following specifications are made in the system common definition:

```
                :
set pd_rdarea_open_attribute_use = Y                    1
set pd_rdarea_open_attribute = DEFER                    2
                :
```

Explanation

1. Specifies that DEFER or SCHEDULE is to be used as the opening trigger attribute for RDAREAs.
2. Specifies DEFER as the default opening trigger attribute for all RDAREAs.

(3) Start HiRDB normally

```
pdstart
```

(4) Use the pdhold command to place RDAREAs in shutdown and closed status to change their opening trigger attributes

```
pdhold -r RDAREA1,RDAREA2,RDAREA3 -c
```

(5) Create the control statements file for the pdmod command

A control statements file that contains the pdmod command's alter rdarea statement (/pdmod/alter01) is created. The following are the contents of the control statements file:

```
alter rdarea RDAREA01 open attribute SCHEDULE;
alter rdarea RDAREA02 open attribute SCHEDULE;
alter rdarea RDAREA03 open attribute SCHEDULE;
```

Explanation

The opening trigger attributes for RDAREA1 to RDAREA3 are changed from DEFER to SCHEDULE.

(6) Use the pdmod command to change the RDAREAs' opening trigger

```
pdmod -a /pdmod/alter01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (5).

Reference note:

The opening trigger specification does not take effect immediately when RDAREA is added by the database structure modification utility; immediately after adding the RDAREA, the INITIAL attribute takes effect. To enable the opening trigger specification to take effect, you must first terminate HiRDB, and then execute a restart. The opening trigger will then take effect regardless of the startup mode.

(7) Use the pdrels command to release shutdown status of RDAREAs whose opening trigger attributes were changed, then place them in open status

```
pdrels -r RDAREA1,RDAREA2,RDAREA3 -o
```

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

15.6 Deleting an RDAREA

Executor: HiRDB administrator

An RDAREA can be deleted by the `remove rdarea` statement of the database structure modification utility (`pdmod` command).

15.6.1 Before deleting an RDAREA

(1) RDAREAs that can be deleted

The following RDAREAs can be deleted:

- User RDAREAs
- User LOB RDAREAs
- List RDAREAs

(2) When to delete RDAREAs

An RDAREA should be deleted under the following circumstances:

- It is an existing RDAREA that is no longer needed.
- It is an RDAREA that needs to be re-created.

(3) Notes

- If there are any tables or indexes in an RDAREA that is to be deleted, those tables or indexes must be deleted before the RDAREA can be deleted.
- Before an RDAREA is deleted, it must be placed in shutdown or error shutdown status by the `pdhold` command and then in closed status by the `pdclose` command.

(4) Changing global buffer definitions

Always change the value specified to the `pdbuffer` operand after you delete an RDAREA. That is, delete the RDAREA from the global buffer definition. If you do not change the value specified in the `pdbuffer` operand, memory utilization efficiency may deteriorate. Consider the following situations as examples:

- If a deleted RDAREA is assigned to a global buffer specified with the `-o` option of the `pdbuffer` operand, another RDAREA may be allocated a global buffer that is larger than required.
- If only the deleted RDAREA is assigned to the global buffer specified with the `-o` option of the `pdbuffer` operand, the unused global buffer remains resident in memory.

15.6.2 Example

This example deletes a user RDAREA named RDAREA1.

Procedure

1. Delete any table or index from the RDAREA to be deleted.
2. Use the pdhold command to place the RDAREA to be deleted in shutdown and closed status.
3. Create a control statements file for the pdmod command.
4. Use the pdmod command to delete the RDAREA.
5. Use the pdbufmod command to delete the global buffer.
6. Update the pdbuffer operand.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Delete any table or index from RDAREA to be deleted

```
DROP TABLE TABLE1;
DROP INDEX INDEX1;
```

(2) Use the pdhold command to place the RDAREA to be deleted in shutdown and closed status

```
pdhold -r RDAREA1 -c
```

(3) Create the control statements file for the pdmod command

A control statements file that contains the pdmod command's remove rdarea statement (/pdmod/remove01) is created. The following are the contents of the control statements file:

```
remove rdarea RDAREA1;
```

Explanation

The RDAREA to be deleted (RDAREA1) is specified.

(4) Use the pdmod command to delete the RDAREA

```
pdmod -a /pdmod/remove01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (3).

(5) Use the pdbufmod command to delete the global buffer

If you no longer need the global buffer that was allocated for the deleted RDAREA (it is allocated for no other RDAREAs), delete that global buffer.

```
pdbufmod -k del -a gbuf01
```

Explanation

-k del: Specifies that a global buffer is to be deleted.

-a: Specifies the name of the global buffer being deleted.

All the following conditions must be satisfied in order to execute the pdbufmod command:

- HiRDB Advanced High Availability is installed.
- The value Y is specified in the pd_dbbuff_modify operand.

(6) Update the pdbuffer operand

The global buffer deletion information becomes invalid when HiRDB is terminated normally or through planned termination. Therefore, delete the name of the deleted RDAREA from the pdbuffer operand while HiRDB is stopped.

You can use the system reconfiguration command (pdchgconf command to change the pdbuffer operand specification while HiRDB is running. Note that HiRDB Advanced High Availability must be installed in order to execute the system reconfiguration command. For details about modifying HiRDB system definitions while HiRDB is running, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference*.

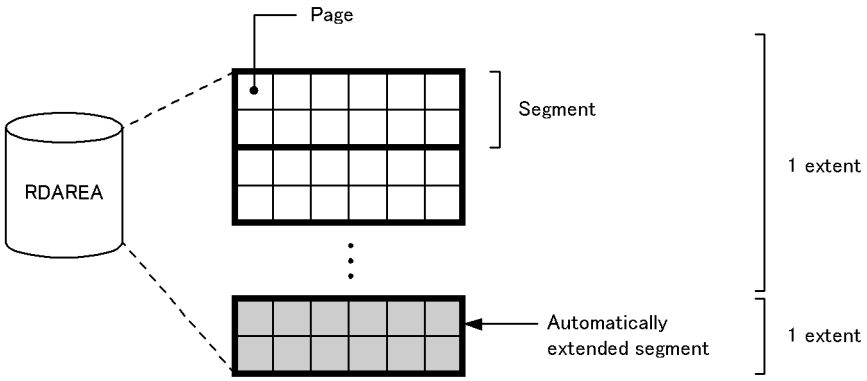
15.7 RDAREA automatic extension

Executor: HiRDB administrator

15.7.1 Automatic extension of an RDAREA

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 15-1 illustrates RDAREA automatic extension.

Figure 15-1: RDAREA automatic extension



A contiguous area in a HiRDB file system area is called an *extent*. The maximum number of extents in a HiRDB file is 24. During automatic extension, if contiguous free space can be allocated following the last allocated extent of the target HiRDB file, the number of extents does not increase. However, if non-contiguous free space is allocated, the number of extents increases. Extent information can be checked with the `pdfls` command.

When RDAREAs are deleted, reinitialized (allocated size is reduced or `with reconstruction` is specified), or combined, allocated extents are deleted or their sizes are reduced, as a result creating fragmented free spaces within the HiRDB file system. Note that if RDAREAs are added, expanded, or reinitialized in this state, multiple extents may be allocated to a single HiRDB file even if automatic extension is not executed.

(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

The following is the procedure for setting automatic extension:

Procedure

To set the 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).
2. 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, database structure modification utility, or registry facility initialization utility.

(3) Notes

1. Automatic extension cannot be applied to an RDAREA that stores an abstract data type provided by a plug-in. However, in the case of the HiRDB Text Search Plug-in with the `SGMLTEXT` type, a portion of the index management area can be extended automatically.
2. If automatic extension is not possible because of a space shortage in the HiRDB file system area, you must either extend the RDAREA or reorganize the tables or indexes in the RDAREA.
3. If the number of extents exceeds the maximum, which is 24, you must either consolidate the extents in the HiRDB file system area into a single extent or extend the RDAREA. To consolidate extents in a HiRDB file system area, first make a backup of the HiRDB file system area with the `pdfbkup` command, then initialize the HiRDB file system area with the `pdfmkfs` command. You can then use the `pdfrstr` command to restore the HiRDB file system area.
4. While an RDAREA is in updatable backup hold status or updatable backup hold (`WAIT` mode) status, RDAREA automatic extension is suppressed. For this reason, you should refrain, while an RDAREA is in either of theses statuses, from executing a job that adds or updates a large volume of data, thus requiring allocation of new pages. To release suppression of RDAREA automatic extension, use the `pdrels` command to release the RDAREA from the hold.
5. When the hybrid method is used in Real Time SAN Replication, HiRDB waits for

the database to be synchronized with the remote site when RDAREA automatic extension occurs. For this reason, an extension operation my result in a processing overhead of 2 seconds or longer.

15.7.2 Example

This example adds a user RDAREA (`RDAREA1`) to which the automatic extension facility is applied.

Procedure

1. Use the `pdfmkfs` command to create a HiRDB file system area for the RDAREA.
2. Create a control statements file for the `pdmod` command.
3. Use the `pdmod` command to add the RDAREA.
4. Use the `pdcopy` command to back up data.
5. Use the `pdbufmod` command to allocate a global buffer.
6. Update the `pdbuffer` operand.

The procedure step numbers correspond to the paragraph numbers in the explanation that follows. For example, step 3 above is explained in paragraph (3) below.

(1) Use the pdfmkfs command to create a HiRDB file system area for the RDAREA

```
pdfmkfs -n 100 -l 10 -e 230 -k DB -i /rdarea/area01
```

Explanation

Creates a HiRDB file system area (`/rdarea/area01`) for RDAREAs.

`-n`: Specifies the size (in megabytes) of the HiRDB file system area.

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

`-e`: Specifies the maximum number of extensions.

`-k`: Specifies `DB` because the HiRDB file system area is to be used for RDAREAs.

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

`/rdarea/area01`: Specifies a name for the HiRDB file system area that is to be created.

(2) Create the control statements file for the pdmod command

A control statements file (`/pdmod/create01`) is created for the `create rdarea`

statement of the pdmod command. The following are the contents of the control statements file:

```
create rdarea RDAREA1                              1
   globalbuffer gbuf01                             2
   for user used by PUBLIC                         3
   server name bes1                                4
   page 4096 characters                            5
   storage control segment 10 pages                6
   extension use 500 segments                      7
   file name "/rdarea/area01/file01"               8
      initial 1000 segments;                       9
```

Explanation

1. Specifies that an RDAREA named RDAREA1 is to be created.
2. Specifies the global buffer (gbuf01) to be allocated for RDAREA1. Because a global buffer specified with this operand is not allocated when HiRDB is started subsequently, you must change the value specified in the pdbuffer operand. Note that if you allocate a global buffer in step (5) below, you need not specify this operand.
3. Specifies that RDAREA1 is to be a public user RDAREA.
4. In the case of a HiRDB/Parallel Server (only), specifies the name of the server where the RDAREA is to be added.
5. Specifies the page size.
6. Specifies the segment size.
7. Specifies use of the automatic extension facility and the number of segments to be added in each iteration of automatic extension.
8. Specifies the HiRDB file comprising the RDAREA.

   /rdarea/area01 is the HiRDB file system area created in step (1).
9. Specifies the number of HiRDB file segments.

(3) Use the pdmod command to add the RDAREA

```
pdmod -a /pdmod/create01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command created in step (2).

(4) Use the pdcopy command to make a backup

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

`-m`: Specifies the name of the first HiRDB file in the master directory RDAREA.

`-M`: Specifies the backup acquisition mode.

`-a`: Specifies that all RDAREAs are to be backed up. When an RDAREA is added, the contents of the master directory and data dictionary RDAREAs are updated, so all RDAREAs are backed up.

`-b`: Specifies a name for the backup file.

`-p`: Specifies the output destination for the `pdcopy` command's processing results listing.

(5) Use the pdbufmod command to allocate a global buffer

This step adds a new global buffer (`gbuf01`) and allocates it for `RDAREA1`.

```
pdbufmod -k add -a gbuf01 -r RDAREA1 -n 1000
```

Explanation

`-k add`: Specifies that a global buffer is to be added.

`-a`: Specifies the name of the global buffer being added.

`-r`: Specifies the RDAREA for which the global buffer is being allocated.

`-n`: Specifies the sector count of the global buffer.

Note, however, that both the following conditions must be satisfied in order to use the `pdbufmod` command:

- HiRDB Advanced High Availability is installed.
- The value `Y` is specified in the `pd_dbbuff_modify` operand.

(6) Update the pdbuffer operand

The global buffer allocated in this procedure becomes invalid if HiRDB is terminated normally or through a planned termination. Therefore, change the `pdbuffer` operand specification while HiRDB is stopped. An example of the `pdbuffer` operand specification follows:

```
                    :
pdbuffer -a gbuf01 -r RDAREA1,RDAREA2,RDAREA3 -n 1000
                    :
```

Explanation

The added RDAREA (RDAREA1) is assigned to a global buffer (gbuf01).

You can use the system reconfiguration command (pdchgconf command) to change the pdbuffer operand specification while HiRDB is running. Note that HiRDB Advanced High Availability must be installed in order to use the system reconfiguration command. For details about changing HiRDB system definitions using the system reconfiguration command, see *9.2 Modifying HiRDB system definitions while HiRDB is running (system reconfiguration command)*.

It is recommended that after the command has executed you check whether or not the execution results are correct. For details on how to check command execution results, see the manual *HiRDB Version 8 Command Reference.*

15.7.3 Handling space shortages in HiRDB file system areas

The following procedure shows how to handle space shortages in HiRDB file system areas that prevent automatic extension from being performed.

Procedure

1. Use the pdfmkfs command to create a HiRDB file system area.
2. Use the pdrorg command to unload table data. This is the recommended procedure, although this step is not always required. It does, however, improve performance. For details about unloading data, see the manual *HiRDB Version 8 Command Reference*.
3. Use the pdmod command (expand rdarea statement) to expand the RDAREA. Add and allocate the HiRDB file system area created in step 1. For details about expanding RDAREAs, see *15.3 Increasing the size of an RDAREA (RDAREA expansion)*.
4. Use the pdrorg command to reload the table data. Reload the data if you unloaded it in step 2. For details about reloading data, see the manual *HiRDB Version 8 Command Reference*.
5. Use the pdcopy command to back up data. For details about making backups, see *6. Backup Procedures*.

Notes on using the inner replica facility

If you are using the inner replica facility, the number of HiRDB files making up the expanded RDAREA will no longer match the number for the other RDAREAs in the replica group. To copy the contents of an RDAREA to another RDAREA,

you can use one of the methods below. If the RDAREA being expanded is an original RDAREA, take the action described in method 1. If the RDAREA being expanded is a replica RDAREA, take the action described in method 2.

1. Delete all replica RDAREA definitions, and then perform the procedure described above. After finishing, redefine the replica RDAREAs.
2. Perform the procedure described above. Before copying the data object for use in another RDAREA, use the `define copy rdarea` statement of the database structure modification utility to copy the RDAREA configuration information from the data object source. For details, see *Modifying and copying the configuration information of an RDAREA in an inner replica* in the *HiRDB Staticizer Option Version 7 Description and User's Guide*.

15.7.4 Actions to take when the number of extents reaches the maximum value

If the number of extents reaches the maximum value (24), automatic extension can no longer be performed. If this happens, take one of the following actions:

1. Consolidate HiRDB file system area extents.
2. Reinitialize RDAREAs.
3. Expand RDAREAs as described in section 15.7.3.

Use either method 1 or method 3 if multiple RDAREAs are assigned to a single HiRDB file system area.

(1) Consolidate HiRDB file system area extents

The following procedure shows how to consolidate extents (how to merge all extents into one):

Procedure

1. Use the `pdstop` command to terminate HiRDB normally.
2. Use the `pdfbkup` command to back up the HiRDB file system area.
3. Use the `pdfmkfs` command to reinitialize the HiRDB file system area.
4. Use the `pdfrstr` command to recover the HiRDB file system area. Recover using the backup made in step 1.
5. Use the `pdstart` command to start HiRDB normally.

(2) Reinitialize RDAREAs

Reinitialize the RDAREAs. For details about how to reinitialize RDAREAs, see *15.4 Increasing the size of an RDAREA or modifying its attributes (RDAREA reinitialization)*. If you specify the `initialize rdarea` statement of the `pdmod` command, however, take note of the following cautions:

- Specify the `with reconstruction` operand.
- Specify the number of segments required in the `initial` statement.

If you do not specify these operands, the RDAREAs will be re-created at the same sizes they were when defined originally.

15.8 Moving an RDAREA (RDAREA migration)

Executor: HiRDB administrator

By using the `move rdarea` statement of the database structure modification utility (`pdmod` command), you can migrate, or move, RDAREAs to other back-end servers. The functionality that enables RDAREAs to be moved is available only with a HiRDB/Parallel Server.

15.8.1 Before moving RDAREAs

(1) RDAREAs that can be moved

The following RDAREAs can be moved:

- User RDAREAs
- User LOB RDAREAs

(2) Procedure

The procedure for moving RDAREAs is shown below. Note that the migration procedure described below explains the basic operation; the procedure you use may differ slightly depending on the configuration of your system. For details, see the examples starting in section 15.8.2.

Procedure

1. Re-estimate the memory requirements for the server machine running the target back-end server.
2. Use the `pdcopy` command to make required backups. The RDAREAs to be moved, the master directory RDAREA, and the data dictionary RDAREAs must be backed up.
3. Use the `pdhold` command to place the RDAREAs to be moved in shutdown and closed status.
4. Use the `pdmod` command to move the RDAREAs.
5. Use the `pdstop` command to terminate HiRDB normally. After you have moved the RDAREAs, HiRDB must be stopped. Processing after the RDAREAs have been moved cannot be guaranteed if you do not stop HiRDB at this point.
6. Transfer the HiRDB files comprising the RDAREAs being moved.*
7. Use the `pdstart` command to start HiRDB normally.
8. Use the `pdcopy` command to make required backups. The RDAREAs that were moved, the master directory RDAREA, and the data dictionary

RDAREAs must be backed up.

Note

Do not execute any UAPs or other utilities from the time you begin moving RDAREAs until the time you start HiRDB (steps 4-7).

* This step is required when you are moving RDAREAs to another server machine. The HiRDB files of the RDAREAs that are being moved must be transferred to the destination server machine. In addition, the path names of these HiRDB files must be the same as their path name on the source server machine. If the same path names cannot be used, create a symbolic link aliasing the original path to the directory on the target server machine that contains the files.

You must also delete the HiRDB files on the source server machine after they have been transferred. For details on how to do this, see the examples starting in section 15.8.2.

(3) Notes

(a) Move all related RDAREAs

If you decide to move an RDAREA, you must move all related RDAREAs, as listed below. If you do not move all related RDAREAs, an error will occur when you execute the database structure modification utility (`pdmod` command).

- Move any RDAREA that contains an index defined for any table stored in an RDAREA that is being moved.
- Move any user LOB RDAREA that contains LOB data of any LOB column defined in any table stored in an RDAREA that is being moved.
- Move, by partitioning unit, any RDAREA containing any table, index, or LOB column that is in a row-partitioned table stored in an RDAREA that is being moved.
- Move all RDAREAs within any replica group RDAREA that is being moved. In this case, you must also pre-register the generation number of the target HiRDB file system area.

(b) If a non-partitioning key index is defined for the table

You cannot move an RDAREA containing a table for which a non-partitioning key index is defined. In such a case, use the procedure explained below to move the RDAREA.

Procedure

1. Use `DROP INDEX` to delete the non-partitioning key index.
2. Move the RDAREA.

3. Use CREATE INDEX to re-create the non-partitioning key index.

Note that you may not be able to re-create a non-partitioning key index for which UNIQUE is defined. For details about the situations in which you cannot re-create a non-partitioning key index, see the description of CREATE INDEX in the manual *HiRDB Version 8 SQL Reference*.

(c) If a primary key is defined for a key other than a table partitioning key

You cannot move an RDAREA that contains a table for which a primary key other than a table partitioning key is defined. In such a case, use the procedure explained below to move the RDAREA.

Procedure

1. Use the pdrorg command to unload the table data.
2. Use DROP TABLE to delete the table definition.
3. Move the RDAREA.
4. Use CREATE TABLE to define a table.
5. Use the pdrorg command to reload the table data.

(d) If any routines for converting a table to a base table exist

Because any routine that converts a table stored in an RDAREA being moved to a base table becomes invalid when the RDAREA is moved, use ALTER ROUTINE to re-create the routine.

15.8.2 Example 1 (Moving RDAREAs to the back-end server on a new server machine)

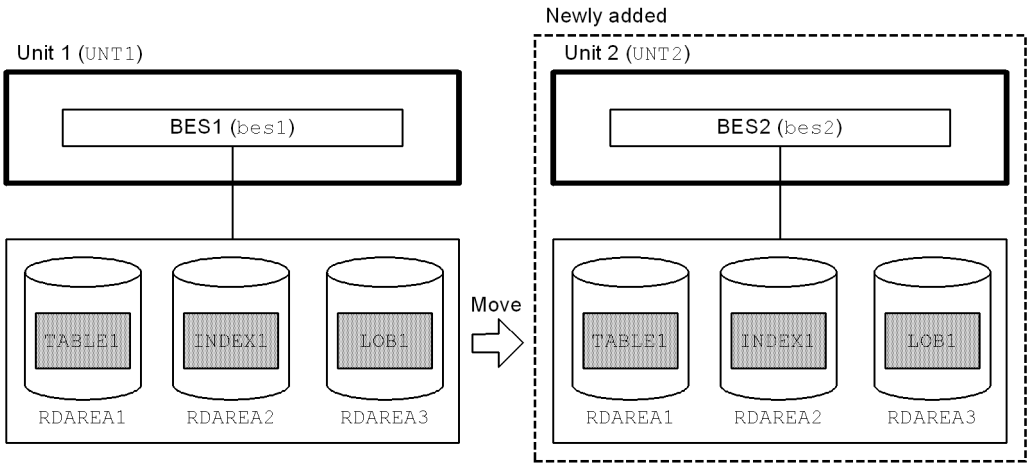
In this example, RDAREAs are moved to a back-end server created on a new server machine.

- RDAREA1 contains a non-row-partitioned table (TABLE1).
- An index (INDEX1) and a LOB column (LOB1) are defined for TABLE1. The index is stored in RDAREA2 and the LOB data is stored in RDAREA3.
- The HiRDB files configuring these RDAREAs are as follows:

  RDAREA1: /area1/rdarea1

  RDAREA2: /area2/rdarea2

  RDAREA3: /area3/rdarea3



Note: Parentheses enclose the unit and server names.

(1) Add unit 2

For details about adding a unit, see *11.1 Adding a unit*.

(2) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01-p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(3) Use the pdhold command to place the RDAREAs being moved in shutdown and closed status

```
pdhold -r RDAREA1,RDAREA2,RDAREA3 -c
```

(4) Create a control statements file for the pdmod command

Create a control statements file (/pdmod/move01) containing a move rdarea statement for the pdmod command. The following shows the contents of this control statements file:

```
move rdarea RDAREA1,RDAREA2,RDAREA3                 1
   to bes2;                                         2
```

Explanation

1. Specifies the names of the RDAREAs being moved.
2. Specifies the name of the target server.

(5) Use the pdmod command to move the RDAREAs

```
pdmod -a /pdmod/move01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command.

(6) Use the pdstop command to terminate HiRDB normally

```
pdstop
```

(7) Transfer the HiRDB files comprising the RDAREAs being moved

Use one of the following methods to transfer the HiRDB files of the RDAREAs being moved to the new server machine.

(a) Transfer by HiRDB file system area

This method requires the following to be satisfied:

- The HiRDB file system area is a regular file.
- The HiRDB file system area contains only the HiRDB file being moved.

Use the operating system's rcp or ftp command to transfer the HiRDB file system areas.

```
rcp /area1 host2:/area1
rcp /area2 host2:/area2
rcp /area3 host2:/area3
```

Explanation

Transfers the HiRDB file system areas for RDAREA1 through RDAREA3.

(b) Transfer by HiRDB file

If you cannot use the method described in (a), use the procedure described below to move the HiRDB files.

■ Operations on the source machine

1. Use the pdfbkup command to back up the HiRDB files to be moved.
2. Use the rcp or ftp command to move the backups you made of the HiRDB files.
3. Use the pdfrm command to delete the originals of the HiRDB files that you moved.

Example

```
pdfbkup /area1/rdarea1 /tmp/bk_rdarea1          1
pdfbkup /area2/rdarea2 /tmp/bk_rdarea2          1
pdfbkup /area3/rdarea3 /tmp/bk_rdarea3          1
rcp /tmp/"bk_*" host2:/tmp/                     2
pdfrm /area1/rdarea1                            3
pdfrm /area2/rdarea2                            3
pdfrm /area3/rdarea3                            3
```

Explanation

1. Backs up the HiRDB files of RDAREA1 through RDAREA3.
2. Moves the backups of the HiRDB files.
3. Deletes the HiRDB files of RDAREA1 through RDAREA3.

■ Operations on the target machine

1. Use the pdfmkfs command to create HiRDB file system areas.
2. Use the pdfrstr command to restore the HiRDB files that you moved.

Example

```
pdfmkfs -n 30 -l 10 -k DB /area1                1
pdfmkfs -n 30 -l 10 -k DB /area2                1
pdfmkfs -n 30 -l 10 -k DB /area3                1
pdfrstr /tmp/bk_rdarea1 /area1                  2
pdfrstr /tmp/bk_rdarea2 /area2                  2
pdfrstr /tmp/bk_rdarea3 /area3                  2
```

Explanation

1. Creates the HiRDB files system areas for RDAREA1 to RDAREA3. Use the same path name as was used on the source machine. If you cannot use the same path name, create a symbolic link aliasing the original path to the directory on the target machine that contains the files.
2. Lists the HiRDB files of RDAREA1 to RDAREA3.

(8) Use the pdstart command to start HiRDB normally

```
pdstart
```

(9) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02-p /pdcopy/list02
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

15.8.3 Example 2 (Moving RDAREAs to a back-end server in a different unit)

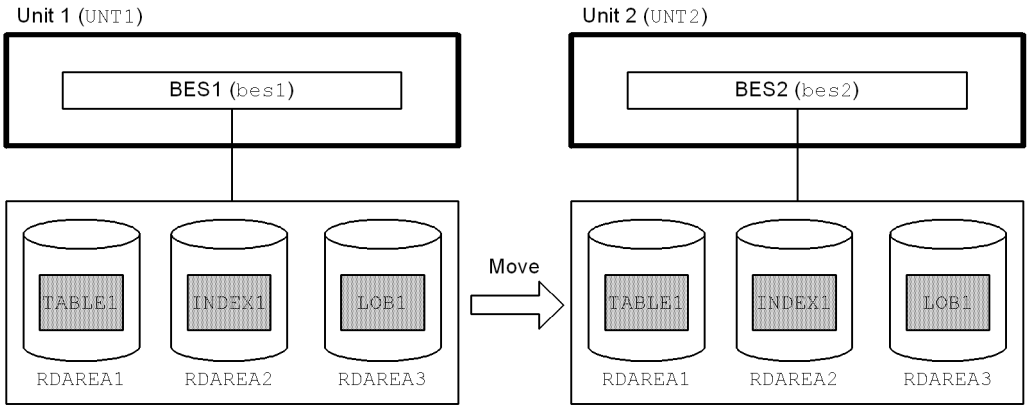
In this example, RDAREAs are moved to a back-end server in a different unit.

- RDAREA1 contains a non-row-partitioned table (TABLE1).
- An index (INDEX1) and a LOB column (LOB1) are defined for TABLE1. The index is stored in RDAREA2 and the LOB data is stored in RDAREA3.
- The HiRDB files configuring these RDAREAs are as follows:

  RDAREA1: /area1/rdarea1

  RDAREA2: /area2/rdarea2

  RDAREA3: /area3/rdarea3



Note: Parentheses enclose the unit and server names.

(1) Estimate memory requirements

Estimate memory requirements for the server machine on which the target unit is running. For details about estimating memory requirements, see the manual *HiRDB Version 8 Installation and Design Guide*.

The subsequent steps are identical to the steps in section 15.8.2, beginning with *15.8.2(2) Use the pdcopy command to back up the RDAREAs*.

15.8.4 Example 3 (Moving RDAREAs to a different back-end server in the same unit)

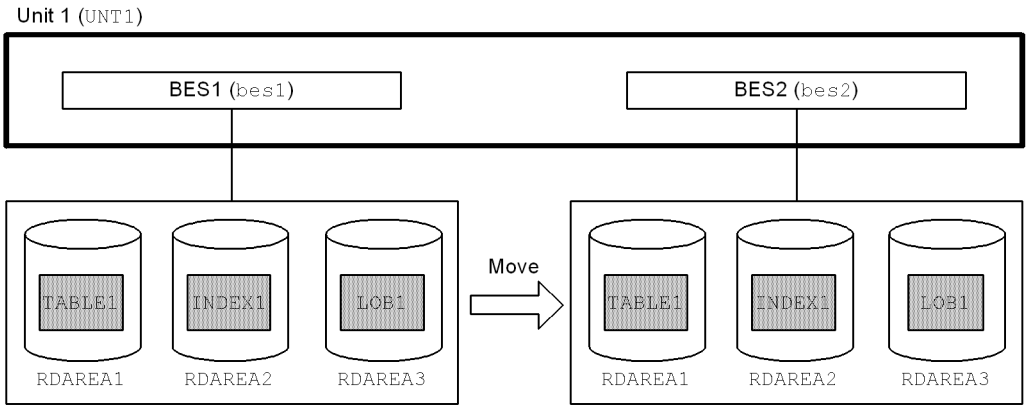
In this example, RDAREAs are moved to a different back-end server in the same unit.

- `RDAREA1` contains a non-row-partitioned table (`TABLE1`).
- An index (`INDEX1`) and a LOB column (`LOB1`) are defined for `TABLE1`. The index is stored in `RDAREA2` and the LOB data is stored in `RDAREA3`.
- The HiRDB files configuring these RDAREAs are as follows:

  `RDAREA1`: `/area1/rdarea1`

  `RDAREA2`: `/area2/rdarea2`

  `RDAREA3`: `/area3/rdarea3`



Note: Parentheses enclose the unit and server names.

(1) Estimate memory requirements

Estimate memory requirements for the target back-end server. For details about estimating memory requirements, see the manual *HiRDB Version 8 Installation and Design Guide*.

(2) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up.

Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(3) Use the pdhold command to place the RDAREAs being moved in shutdown and closed status

```
pdhold -r RDAREA1,RDAREA2,RDAREA3 -c
```

(4) Create a control statements file for the pdmod command

Create a control statements file (`/pdmod/move01`) containing a `move rdarea` statement for the `pdmod` command. The following shows the contents of this control statements file:

```
move rdarea RDAREA1,RDAREA2,RDAREA3                 1
   to bes2;                                         2
```

Explanation

1. Specifies the names of the RDAREAs being moved.
2. Specifies the name of the target server.

(5) Use the pdmod command to move the RDAREAs

```
pdmod -a /pdmod/move01
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command.

(6) Use the pdstop command to terminate HiRDB normally

```
pdstop
```

(7) Use the pdstart command to start HiRDB normally

```
pdstart
```

(8) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

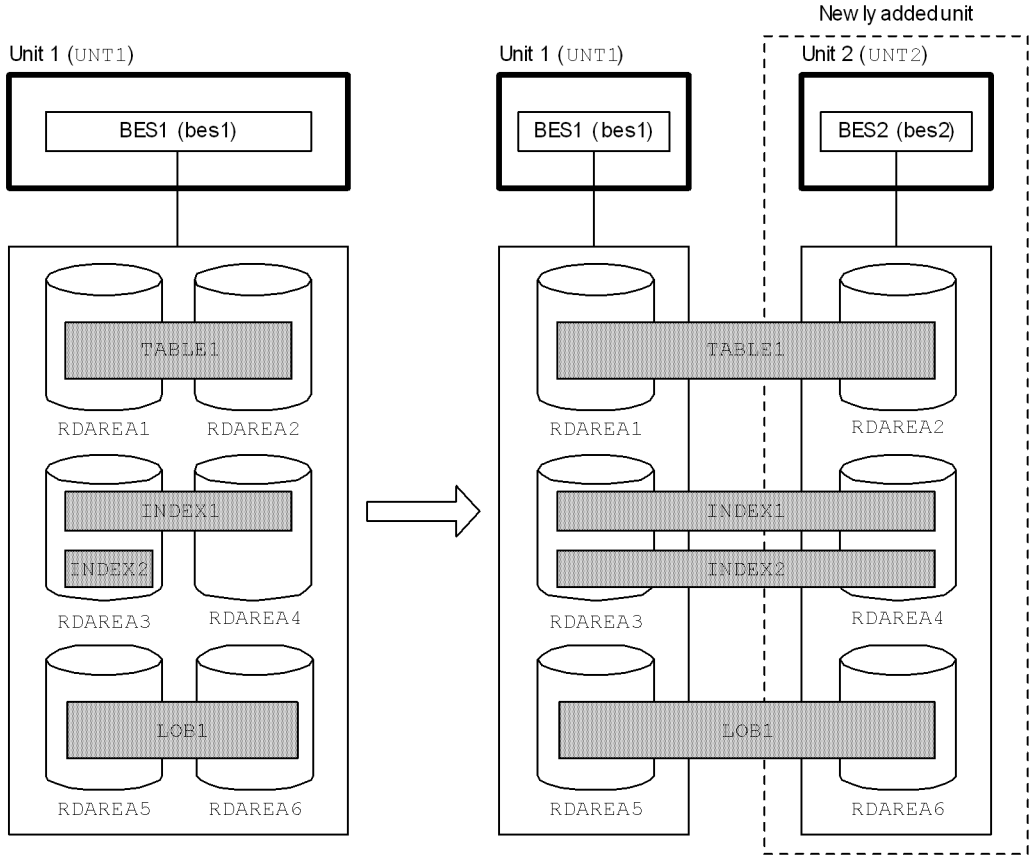
-p: Specifies the output destination of the pdcopy command's processing results listing.

15.8.5 Example 4 (Moving RDAREAs containing a row-partitioned table)

In this example, RDAREAs are moved to a back-end server created on a new server machine. A table that is row-partitioned within a server (TABLE1) is converted to one that is row-partitioned among servers.

- RDAREA1 and RDAREA2 contain a row-partitioned table (TABLE1).
- A partitioning key index (INDEX1) and a non-partitioning key index (INDEX2) are defined for TABLE1. A LOB column (LOB1) is also defined. INDEX1 is stored in RDAREA3 and RDAREA4, INDEX2 is stored in RDAREA3, and LOB data is stored in RDAREA5 and RDAREA6.
- The HiRDB files configuring these RDAREAs are as follows:

```
RDAREA1: /area1/rdarea1
RDAREA2: /area2/rdarea2
RDAREA3: /area3/rdarea3
RDAREA4: /area4/rdarea4
RDAREA5: /area5/rdarea5
RDAREA6: /area6/rdarea6
```



Note: Parentheses enclose the unit and server names.

(1) Add unit 2

For details about adding a unit, see *11.1 Adding a unit*.

(2) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master

directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(3) Use DROP INDEX to delete the non-partitioning key index

```
DROP INDEX INDEX2;
```

Because you cannot move an RDAREA that contains a non-partitioning key index, use DROP INDEX to delete INDEX2 (the non-partitioning key index).

(4) Use the pdhold command to place the RDAREAs being moved in shutdown and closed status

```
pdhold -r RDAREA2,RDAREA4,RDAREA6 -c
```

(5) Create a control statements file for the pdmod command

Create a control statements file (/pdmod/move01) containing a move rdarea statement for the pdmod command. The following shows the contents of this control statements file:

```
move rdarea RDAREA2,RDAREA4,RDAREA6                    1
   to bes2;                                            2
```

Explanation

1. Specifies the names of the RDAREAs being moved.
2. Specifies the name of the target server.

(6) Use the pdmod command to move the RDAREAs

```
pdmod -a /pdmod/move01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command.

(7) Use the pdstop command to terminate HiRDB normally

```
pdstop
```

(8) Transfer the HiRDB files comprising the RDAREAs being moved

Use one of the methods explained below to transfer the HiRDB files of the RDAREAs being moved to the new server machine.

(a) Transfer by HiRDB file system area

This method requires that the following be satisfied:

- The HiRDB file system area is a regular file.
- The HiRDB file system area contains only the HiRDB file being moved.

Use the operating system's `rcp` or `ftp` command to transfer the HiRDB file system areas.

```
rcp /area2 host2:/area2
rcp /area4 host2:/area4
rcp /area6 host2:/area6
```

Explanation

Transfers the HiRDB file system areas for `RDAREA2`, `RDAREA4`, and `RDAREA6`.

(b) Transfer by HiRDB file

If you cannot use the method described in (a), use the procedure described below to move the HiRDB files.

■ Operations on the source machine

1. Use the `pdfbkup` command to back up the HiRDB files to be moved.
2. Use the `rcp` or `ftp` command to move the backups you made of the HiRDB files.
3. Use the `pdfrm` command to delete the originals of the HiRDB files that you moved.

Example

```
pdfbkup /area2/rdarea2 /tmp/bk_rdarea2          1
pdfbkup /area4/rdarea4 /tmp/bk_rdarea4          1
pdfbkup /area6/rdarea6 /tmp/bk_rdarea6          1
rcp /tmp/"bk_*" host2:/tmp/                     2
pdfrm /area2/rdarea2                            3
pdfrm /area4/rdarea4                            3
pdfrm /area6/rdarea6                            3
```

Explanation

1. Backs up the HiRDB files of RDAREA2, RDAREA4, and RDAREA6.
2. Moves the backups of the HiRDB files.
3. Deletes the HiRDB files of RDAREA2, RDAREA4, and RDAREA6.

■ Operations on the target machine

1. Use the pdfmkfs command to create HiRDB file system areas.
2. Use the pdfrstr command to restore the HiRDB files that you moved.

Example

```
pdfmkfs -n 30 -l 10 -k DB /area2                1
pdfmkfs -n 30 -l 10 -k DB /area4                1
pdfmkfs -n 30 -l 10 -k DB /area6                1
pdfrstr /tmp/bk_rdarea2 /area2                  2
pdfrstr /tmp/bk_rdarea4 /area4                  2
pdfrstr /tmp/bk_rdarea6 /area6                  2
```

Explanation

1. Creates the HiRDB files system areas RDAREA2, RDAREA4, and RDAREA6. Use the same path name as was used on the source machine. If you cannot use the same path name, create a symbolic link aliasing the original path to the directory on the target machine that contains the files.
2. Lists the HiRDB files of RDAREA2, RDAREA4, and RDAREA6

(9) Use the pdstart command to start HiRDB normally

```
pdstart
```

(10) Use CREATE INDEX to re-create the non-partitioning key index

```
CREATE INDEX INDEX2 on TABLE1(C2) IN ((RDAREA3),(RDAREA4));
```

Re-create the non-partitioning key index that you deleted in step (3).

(11) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

15.8.6 Example 5 (Moving inner replica RDAREAs)

In this example, an inner replica group RDAREA is moved to a back-end server created on a new server machine.

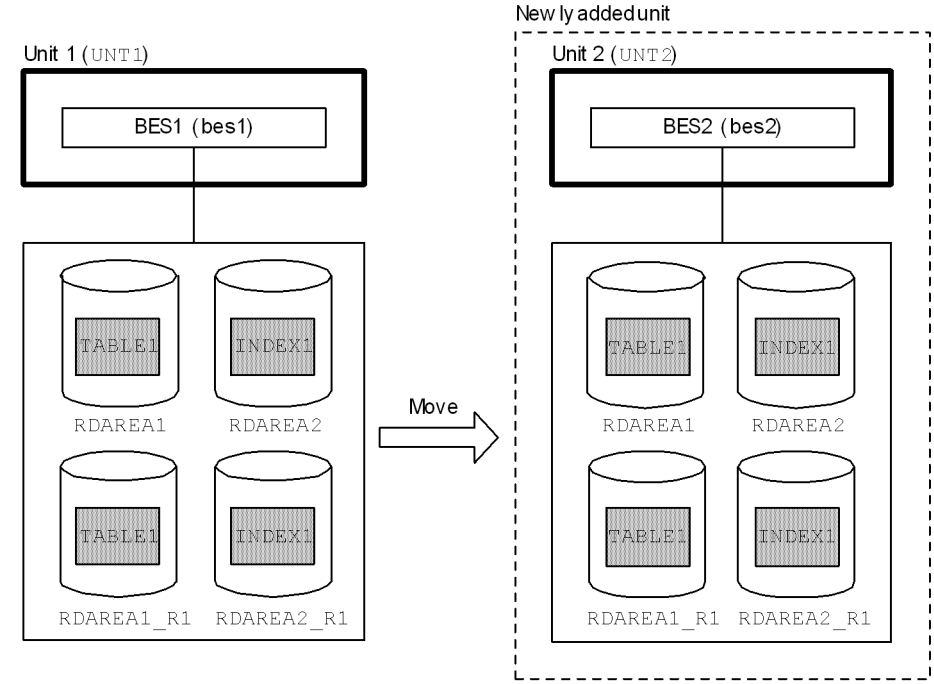
- RDAREA1 contains a non-row-partitioned table (TABLE1).
- An index (INDEX1) is defined for TABLE1. INDEX1 is stored in RDAREA2.
- The replica RDAREA of RDAREA1 is RDAREA1_R1, and the replica RDAREA of RDAREA2 is RDAREA2_R1.
- The HiRDB files configuring these RDAREAs are as follows:

  RDAREA1: /area1/rdarea1

  RDAREA2: /area2/rdarea2

  RDAREA1_R1: /area3/rdarea1

  RDAREA2_R1: /area4/rdarea2



Note: Parentheses enclose the unit and server names.

(1) Add unit 2

For details about adding a unit, see *11.1 Adding a unit*.

(2) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(3) Create a control statements file for the pdmod command

This file is used to register the generation number of the replica HiRDB file system areas on the new back-end server. Create a control statements file (`/pdmod/gen01`) containing a `create generation` statement for the `pdmod` command. The following shows the contents of this control statements file:

```
create generation for HiRDB file system area
       "/area3"                                    1
       server name bes2                            2
       generation number 1                         3
       reproduce "/area1";                         4
create generation for HiRDB file system area
       "/area4"                                    1
       server name bes2                            2
       generation number 1                         3
       reproduce "/area2";                         4
```

Explanation

1. Specifies the names of the replica HiRDB file system areas.

   When the `pdmod` command is executed, the `KFPX24251-W` message is issued. However, this does not pose a problem because you will create the HiRDB file system areas on the target server machine later in step (11).

2. Specifies the name of the server to which the replica HiRDB files system areas are being moved.
3. Specifies the generation number of the replica HiRDB file system areas.
4. Specifies the original HiRDB file system areas.

(4) Use the pdmod command to register the generation number of the replica HiRDB file system areas

```
pdmod -a /pdmod/gen01
```

Explanation

`-a`: Specifies the name of the control statements file for the `pdmod` command.

(5) Use the pdhold command to place the RDAREAs being moved in shutdown and closed status

```
pdhold -r RDAREA1,RDAREA2,RDAREA1_R1,RDAREA2_R1 -c
```

(6) Create a control statements file for the pdmod command

Create a control statements file (/pdmod/move01) containing a move rdarea statement for the pdmod command. The following shows the contents of this control statements file:

```
move rdarea RDAREA1,RDAREA2,RDAREA1_R1,RDAREA2_R1          1
   to bes2;                                                2
```

Explanation

1. Specifies the names of the RDAREAs being moved.
2. Specifies the name of the target server.

(7) Use the pdmod command to move the RDAREAs

```
pdmod -a /pdmod/move01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command.

(8) Use the pdstop command to terminate HiRDB normally

```
pdstop
```

(9) Transfer the HiRDB files comprising the RDAREAs being moved

Use one of the following methods to transfer the HiRDB files of the RDAREAs being moved to the new server machine.

(a) Transfer by HiRDB file system area

This method requires that the following be satisfied:

- The HiRDB file system area is a regular file.
- The HiRDB file system area contains only the HiRDB file being moved.

Use the operating system's rcp or ftp command to transfer the HiRDB file system areas.

```
rcp /area1 host2:/area1
rcp /area2 host2:/area2
rcp /area3 host2:/area3
rcp /area4 host2:/area4
```

Explanation

Transfers the HiRDB file system areas for RDAREA1 through RDAREA2, and for RDAREA1_R1 through RDAREA2_R1.

(b) Transfer by HiRDB file

If you cannot use the method described in (a), use the procedure described below to move the HiRDB files.

■ Operations on the source machine

1. Use the pdfbkup command to back up the HiRDB files to be moved.
2. Use the rcp or ftp command to move the backups you made of the HiRDB files.
3. Use the pdfrm command to delete the originals of the HiRDB files that you moved.

Example

```
pdfbkup /area1/rdarea1 /tmp/bk_rdarea1            1
pdfbkup /area2/rdarea2 /tmp/bk_rdarea2            1
pdfbkup /area3/rdarea1 /tmp/bk_rdarea1_r1         1
pdfbkup /area4/rdarea2 /tmp/bk_rdarea2_r1         1
rcp /tmp/"bk_*" host2:/tmp/                       2
pdfrm /area1/rdarea1                              3
pdfrm /area2/rdarea2                              3
pdfrm /area3/rdarea1                              3
pdfrm /area4/rdarea2                              3
```

Explanation

1. Backs up the HiRDB files of RDAREA1 through RDAREA2, and of RDAREA1_R1 through RDAREA2_R1.
2. Moves the backups of the HiRDB files.
3. Deletes the HiRDB files of RDAREA1 to RDAREA2, and of RDAREA1_R1 to RDAREA2_R1

■ Operations on the target machine

1. Use the pdfmkfs command to create HiRDB file system areas.

2. Use the pdfrstr command to restore the HiRDB files that you moved.

Example

```
pdfmkfs -n 30 -l 10 -k DB /area1              1
pdfmkfs -n 30 -l 10 -k DB /area2              1
pdfmkfs -n 30 -l 10 -k DB /area3              1
pdfmkfs -n 30 -l 10 -k DB /area4              1
pdfrstr /tmp/bk_rdarea1 /area1                2
pdfrstr /tmp/bk_rdarea2 /area2                2
pdfrstr /tmp/bk_rdarea1_r1 /area3             2
pdfrstr /tmp/bk_rdarea2_r1 /area4             2
```

Explanation

1. Creates the HiRDB files system areas for RDAREA1 to RDAREA2, and for RDAREA1_R1 to RDAREA2_R1. Use the same path name as was used on the source machine. If you cannot use the same path name, create a symbolic link aliasing the original path to the directory on the target machine that contains the files.
2. Lists the HiRDB files of RDAREA1 to RDAREA2, and of RDAREA1_R1 to RDAREA2_R1

(10) Use the pdstart command to start HiRDB normally

```
pdstart
```

(11) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

15.8.7 Example 6 (Moving RDAREAs containing an abstract data type)

In this example, RDAREAs containing an abstract data type (SGMLTEXT type) are moved to a back-end server created on a new server machine. A table that is row-partitioned within a server (TABLE1) is converted to one that is row-partitioned among servers.

- RDAREA1 and RDAREA2 contain a row-partitioned table (TABLE1).
- An abstract data type (SGMLTEXT type) and a plug-in index are defined for TABLE1. The SGMLTEXT data is stored in RDAREA3 and RDAREA4, and the plug-in index is stored in RDAREA5 and RDAREA6.
- The HiRDB files configuring these RDAREAs are as follows:

  RDAREA1: /area1/rdarea1

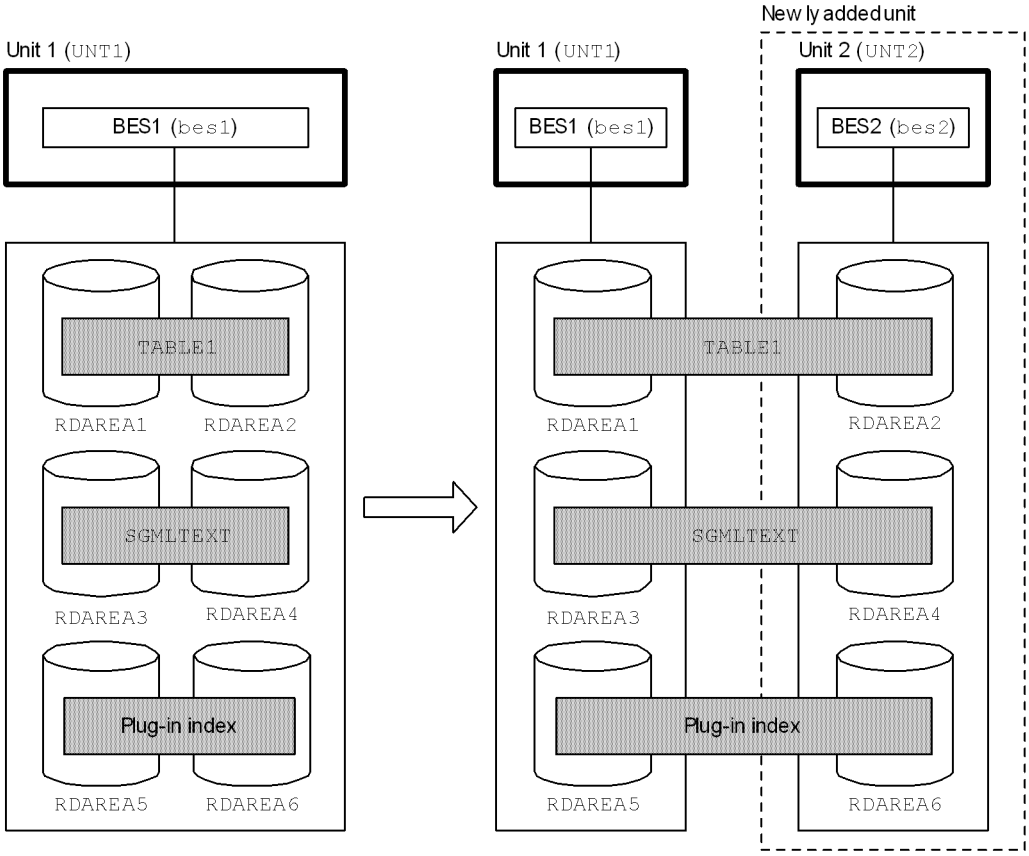
  RDAREA2: /area2/rdarea2

  RDAREA3: /area3/rdarea3

  RDAREA4: /area4/rdarea4

  RDAREA5: /area5/rdarea5

  RDAREA6: /area6/rdarea6



Note: Parentheses enclose the unit and server names.

(1) Add unit 2

For details about adding a unit, see *11.1 Adding a unit*.

(2) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup01 -p /pdcopy/list01
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an

RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

(3) Use the pdhold command to place the RDAREAs being moved in shutdown and closed status

```
pdhold -r RDAREA2,RDAREA4,RDAREA6 -c
```

(4) Create a control statements file for the pdmod command

Create a control statements file (/pdmod/move01) containing a move rdarea statement for the pdmod command. The following shows the contents of this control statements file:

```
move rdarea RDAREA2,RDAREA4,RDAREA6                    1
   to bes2;                                            2
```

Explanation

1. Specifies the names of the RDAREAs being moved.
2. Specifies the name of the target server.

(5) Use the pdmod command to move the RDAREAs

```
pdmod -a /pdmod/move01
```

Explanation

-a: Specifies the name of the control statements file for the pdmod command.

(6) Use the pdstop command to terminate HiRDB normally

```
pdstop
```

(7) Transfer the HiRDB files comprising the RDAREAs being moved

Use one of the following methods to transfer the HiRDB files of the RDAREAs being moved to the new server machine.

(a) Transfer by HiRDB file system area

This method requires that the following be satisfied:

- The HiRDB file system area is a regular file.
- The HiRDB file system area contains only the HiRDB file being moved.

Use the operating system's `rcp` or `ftp` command to transfer the HiRDB file system areas.

```
rcp /area2 host2:/area2
rcp /area4 host2:/area4
rcp /area6 host2:/area6
```

Explanation

Transfers the HiRDB file system areas for `RDAREA2`, `RDAREA4`, and `RDAREA6`.

(b) Transfer by HiRDB file

If you cannot use the method described in (a), use the procedure described below to move the HiRDB files.

■ Operations on the source machine

1. Use the `pdfbkup` command to back up the HiRDB files to be moved.
2. Use the `rcp` or `ftp` command to move the backups you made of the HiRDB files.
3. Use the `pdfrm` command to delete the originals of the HiRDB files that you moved.

Example

```
pdfbkup /area2/rdarea2 /tmp/bk_rdarea2          1
pdfbkup /area4/rdarea4 /tmp/bk_rdarea4          1
pdfbkup /area6/rdarea6 /tmp/bk_rdarea6          1
rcp /tmp/"bk_*" host2:/tmp/                     2
pdfrm /area2/rdarea2                            3
pdfrm /area4/rdarea4                            3
pdfrm /area6/rdarea6                            3
```

Explanation

1. Backs up the HiRDB files of `RDAREA2`, `RDAREA4`, and `RDAREA6`.
2. Moves the backups of the HiRDB files.
3. Deletes the HiRDB files of `RDAREA2`, `RDAREA4`, and `RDAREA6`.

■ Operations on the target machine

1. Use the pdfmkfs command to create HiRDB file system areas.
2. Use the pdfrstr command to restore the HiRDB files that you moved.

Example

```
pdfmkfs -n 30 -l 10 -k DB /area2              1
pdfmkfs -n 30 -l 10 -k DB /area4              1
pdfmkfs -n 30 -l 10 -k DB /area6              1
pdfrstr /tmp/bk_rdarea2 /area2                2
pdfrstr /tmp/bk_rdarea4 /area4                2
pdfrstr /tmp/bk_rdarea6 /area6                2
```

Explanation

1. Creates the HiRDB files system areas RDAREA2, RDAREA4, and RDAREA6. Use the same path name as was used on the source machine. If you cannot use the same path name, create a symbolic link aliasing the original path to the directory on the target machine that contains the files.
2. Lists the HiRDB files of RDAREA2, RDAREA4, and RDAREA6.

(8) Use the pdstart command to start HiRDB normally

```
pdstart
```

(9) Use the pdcopy command to back up the RDAREAs

```
pdcopy -m /rdarea/mast/mast01 -M r -a -b /pdcopy/backup02 -p /pdcopy/list02
```

Explanation

-m: Specifies the name of the first HiRDB file in the master directory RDAREA.

-M: Specifies the backup acquisition mode.

-a: Specifies that all RDAREAs are to be backed up. Because the master directory RDAREA and the data dictionary RDAREAs are updated when an RDAREA is moved, we recommend that you back up all RDAREAs at this time.

-b: Specifies the name of the backup file.

-p: Specifies the output destination of the pdcopy command's processing results listing.

15.9 Re-using 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. This section explains how to reuse used free pages and segments. The following topics are explained in this section:

- Page and segment status
- Reusing used free pages
- Reusing used free segments

15.9.1 Page and segment status

Before reading about reusing used free pages and used free segments in this section, you must be familiar with page statuses and segment statuses. Table 15-4 lists and describes pages statuses, and Table 15-5 lists and describes segment statuses.

Table 15-4: 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 *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*.
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.

Table 15-5: 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 *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*.
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 be used only by a table or an index that stored data in it. Other

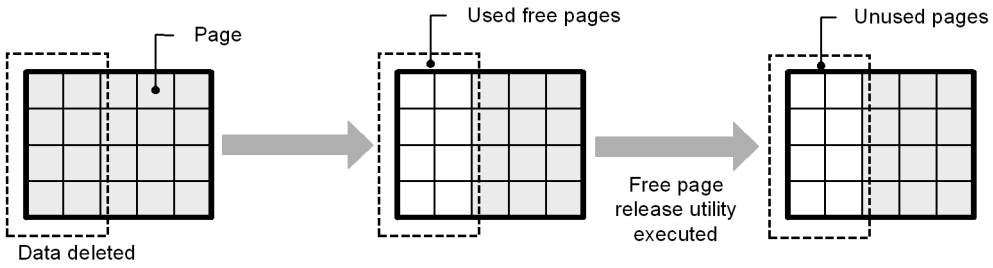
tables or indexes cannot use such a segment.

15.9.2 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 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` command), you can convert used free pages to unused pages and reuse them. This is called *releasing used free pages*. Figure 15-2 provides an overview of releasing used free pages.

Figure 15-2: 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.

(2) Benefits of releasing used free pages

(a) Benefits of releasing used free pages of tables

Table 15-6 lists and describes the benefits of releasing used free pages of a table.

Table 15-6: Benefits of 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

Benefit	Description	Degree of benefit
Improvement in performance for 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 a search). Thus, converting to unused pages improves 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 in a used page, HiRDB performs a process called *page compaction*. In page compaction, data in the affected page is repacked to secure sufficient contiguous free space to enable storage of the new data. When used free pages are released, page compaction is 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.

Particularly beneficial is application of this facility on tables that satisfy the conditions shown in Table 15-7.

Table 15-7: Tables that benefit from release of used free pages

Condition	Reason
A table that has no updates of variable-length character strings.	Because branch rows do not occur in such tables, the probability of being able to release used free pages increases. Branch rows are stored on separate pages from their base rows, and base rows of other data are also stored on those pages. This means that, if there are branch rows, even if all rows of a particular value (such as registered duration) are deleted, the number of pages in which data is stored may not be reduced to zero. However, if there are no branch rows, when data inserted during a certain period is deleted, because the space in that contiguous area (page) becomes free, the probability of being able to release used free pages increases.
A table that has no changes in the number of repetition column elements.	
A table that has no changes of NULL values to real data or real data to NULL values.	

Condition	Reason
A table that has no cluster key index defined.	Because the storage position of a data item is determined from its key value when a cluster key index is defined, even if used free pages are released, the possibility that those pages will not be used is high, and little benefit will be achieved.
A table that has no LOB data.	This facility cannot be used on data stored in LOB RDAREAs.
A table for which data has already been deleted from the same page.	Because such a page is easy to convert to used free pages, the probability of being able to release used free pages increases.
A table that is a `REUSE` table (table on which the free space reusage facility has been used).	Free pages released in a `REUSE` table by the free page release utility can be quickly reused in the free page reuse mode. If the table is not a `REUSE` table, free pages released by the free page release utility cannot be used until new segments in the RDAREA can no longer be allocated.

Based on the conditions described in Table 15-7, the optimal table on which to apply this facility is a `REUSE` table with the `FIX` attribute for which no cluster key index is defined.

(b) Benefits of releasing used free pages of indexes

Table 15-8 lists and describes the benefits of releasing used free pages of an index.

Table 15-8: Benefits of releasing used free pages of an index

Benefit	Description	Degree of benefit
Potential reduction of space shortages in RDAREAs storing indexes	If space runs out even though free pages (used free pages) exist, release the used free pages. Note that this tends to reduce the occurrence of space shortages even for RDAREAs that store indexes with key values that are updated or deleted frequently.	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 a 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:

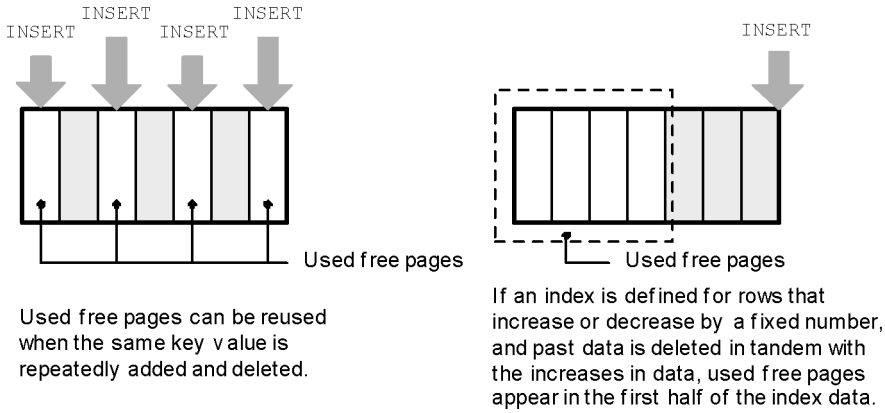
Great: Particularly beneficial.

Good: Always beneficial.

Varies: The degree of benefit varies depending on conditions.

Use of 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 tandem with increases in data, a large number of used free pages that are not reused appear in the first half of the index pages. Figure 15-3 shows the processing of a used free pages being created for index pages.

Figure 15-3: Processing 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.

Reference note:

This processing applies only to used free pages; it performs no page compaction on pages whose storage efficiency has deteriorated. Other than for free used pages, no benefit results if applied when there is a large number of pages whose storage efficiency has deteriorated.

(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 you are 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 have to interrupt normal operations in order to

release free pages.

Use the execution results of the database condition analysis utility to evaluate whether to reorganize tables and 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.

(4) Method of operation

(a) Estimate the system log file capacity

Because updated logs of the database are obtained when used free pages are released, we recommend that you re-estimate the system log file capacity. For details about estimating system log file capacity, see the manual *HiRDB Version 8 Installation and Design Guide*.

(b) Check the page status

Use the database condition analysis utility to assess periodically the page storage efficiency, the number of used free pages, and other statuses. A page whose usage ratio is 0% is a used free page. The page usage ratio and the number of used free pages is displayed in `Used Page Ratio` of *Condition analysis (logical analysis) by RDAREA* and *Condition analysis by table or index*.

If the data storage efficiency has deteriorated due to increases in the number of used free pages, consider releasing used free pages.

(c) Execute the free page release utility

You use the free page release utility to release used free pages.

Before releasing used free pages from a table, consider whether or not to perform page compaction. In the following cases, page compaction provides no benefits, so execute the free page release utility without performing page compaction:

- If most of the used pages are full pages or used free pages
- If all pages are free pages

(d) Check the results

Check the execution results of the free page release utility to determine if used free pages have been released as expected. If there is a large number of used free segments, consider releasing used free segments as well. For details about releasing used free segments, see *15.9.3 Reusing used free segments*.

(e) If the free page release utility terminates abnormally

If the free page release utility terminates abnormally, all used free pages being released up to the one immediately prior to the abnormal termination have been released. Although doing nothing more at this time would not create a problem, re-executing the free page release utility will release the remaining pages.

(5) Notes

Execution of the free page release utility on an index does not cause UAPs to wait for a long time while the utility is executing. However, if a UAP accesses an index that is being processed by this utility, completion processing of the UAP transaction may be delayed.

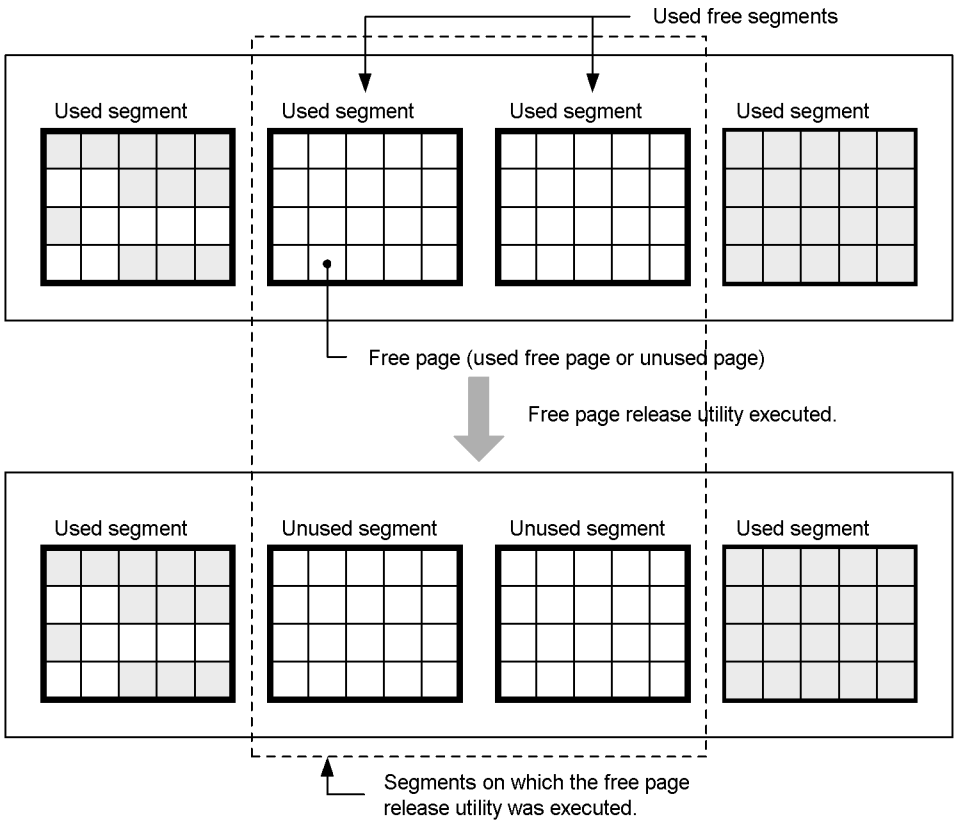
Note that you can specify a maximum wait time in the `-w` option of the free page release utility. If the wait status is not released within the specified wait time, the `KFPH25002-E` message is issued and the free page release utility terminates abnormally. If the `-w` option is omitted, the system continues waiting until the UAP transaction is completed.

15.9.3 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 15-4 provides an overview of releasing used free segments.

Figure 15-4: Releasing used free segments



(2) Benefits of 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 tables can do so. Releasing used free segments converts used free segments to unused segments, which enables other tables to use those segments.

(3) Method of operation

Whether or not to release used free segments is determined from the execution results of the free page release utility or the database condition analysis utility. Release used free segments if there are a large number of them.

(a) Estimate the system log file capacity

Because updated logs of the database are obtained when used free segments are released, we recommend that you re-estimate the system log file capacity. For details about estimating system log file capacity, see the manual *HiRDB Version 8 Installation*

and Design Guide.

(b) Execute the free page release utility

You use the free page release utility to release used free segments. Check the execution results of the free page release utility to determine if used free segment have been released as expected.

Reference note:

A table or index whose used free segments are being released (the free page release utility is executing) cannot be accessed. Because of this, any UAP that attempts to access an RDAREA being processed is placed in wait status. To have the utility return an error without such a UAP being placed in wait status, use the `pdhold` command to shut down the RDAREA before executing the utility.

(c) If the free page release utility terminates abnormally

If the free page release utility terminates abnormally, the system returns to its status before the utility was executed (before any operation was performed). In such a case, simply re-execute the free page release utility. Although not re-executing the free page release utility does not create a problem, no used free segments are released if you do not re-execute.

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)