HITACHI

ノンストップデータベース

HiRDB Version 10 解説

解説書

3020-6-551-C0

前書き

■ 対象製品

●適用 OS：AIX V7.2，AIX V7.3

P-1M62-35A1 HiRDB Server Version 10 10-12

P-1M62-1BA1 HiRDB/Run Time Version 10 10-12

P-1M62-1CA1 HiRDB/Developer's Kit Version 10 10-12

P-1M62-1DA1 HiRDB/Run Time Version 10(64) 10-12

P-1M62-1EA1 HiRDB/Developer's Kit Version 10(64) 10-12

P-F1M62-11A13 HiRDB Staticizer Option Version 10 10-00

P-F1M62-11A15 HiRDB Non Recover Front End Server Version 10 10-00

P-F1M62-11A16 HiRDB Advanced High Availability Version 10 10-00

P-F1M62-11A18 HiRDB Disaster Recovery Light Edition Version 10 10-00

P-F1M62-11A1A HiRDB Accelerator Version 10 10-00

●適用 OS：Red Hat Enterprise Linux 8 (64-bit x86_64)，Red Hat Enterprise Linux 9 (64-bit x86_64)，Red Hat Enterprise Linux 10 (64-bit x86_64)

P-8462-35A1 HiRDB Server Version 10 10-12

P-8462-1DA1 HiRDB/Run Time Version 10(64) 10-12

P-8462-1EA1 HiRDB/Developer's Kit Version 10(64) 10-12

P-F8462-11A13 HiRDB Staticizer Option Version 10 10-00

P-F8462-11A15 HiRDB Non Recover Front End Server Version 10 10-00

P-F8462-11A16 HiRDB Advanced High Availability Version 10 10-00

P-F8462-11A18 HiRDB Disaster Recovery Light Edition Version 10 10-00

P-F8462-11A1A HiRDB Accelerator Version 10 10-00

●適用 OS：Red Hat Enterprise Linux 8 (64-bit x86_64)，Red Hat Enterprise Linux 9 (64-bit x86_64)，Red Hat Enterprise Linux 10 (64-bit x86_64)

P-8362-1BA1 HiRDB/Run Time Version 10 10-12

P-8362-1CA1 HiRDB/Developer's Kit Version 10 10-12

P-8362-3CA1 HiRDB Developer's Suite Version 10 10-12

●適用 OS：Red Hat Enterprise Linux 7 (64-bit x86_64)，Red Hat Enterprise Linux 8 (64-bit x86_64)，Red Hat Enterprise Linux 9 (64-bit x86_64)

P-8462-C5A1 HiRDB Structured Data Access Facility Version 10 10-09

P-8462-AEA1 HiRDB Structured Data Access Facility/Developer's Kit Version 10(64) 10-09

P-F8462-C5A11 HiRDB Structured Data Access Facility Extension for XDM/SD type Version 10 10-04

●適用 OS：Windows Server 2019，Windows Server 2022，Windows Server 2025，Windows 10 Pro (x64)，Windows 10 Enterprise (x64)，Windows 11

P-2962-91A4 HiRDB Server Version 10 10-12

P-2962-7PA4 HiRDB Accelerator Version 10 10-00

P-2962-7HA4 HiRDB Non Recover Front End Server Version 10 10-00

P-2962-7JA4 HiRDB Advanced High Availability Version 10 10-00

●適用 OS：Windows Server 2019，Windows Server 2022，Windows Server 2025，Windows 10，Windows 11

P-2662-11A4 HiRDB/Run Time Version 10 10-12

P-2662-12A4 HiRDB/Developer's Kit Version 10 10-12

P-2662-32A4 HiRDB Developer's Suite Version 10 10-12

●適用 OS：Windows Server 2019，Windows Server 2022，Windows Server 2025，Windows 10 Home (x64)，Windows 10 Pro (x64)，Windows 10 Enterprise (x64)，Windows 11

P-2962-11A4 HiRDB/Run Time Version 10(64) 10-12

P-2962-12A4 HiRDB/Developer's Kit Version 10(64) 10-12

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替および外国貿易法の規制ならびに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

記載の会社名、製品名などは、それぞれの会社の商標もしくは登録商標です。

■ 発行

2026 年 9 月　3020-6-551-C0

■ 著作権

All Rights Reserved. Copyright (C) 2018, 2026, Hitachi, Ltd.

変更内容

変更内容(3020-6-551-C0)　HiRDB Version 10 10-12

追加・変更内容	変更個所
HiRDB Version 10 10-12 のサポート項目一覧を追加しました。	付録 A
プラットフォームごとの HiRDB の機能差の表に次の機能を追加しました。 • ODBC エラーログの特定エラーの出力抑止 • HiRDB ユニット間の KeepAlive 機能	付録 B

単なる誤字・脱字などはお断りなく訂正しました。

変更内容(3020-6-551-B0)　HiRDB Version 10 10-11

追加・変更内容
次の機能をサポートしたことによって，セキュリティの強化，ロール機能の利便性が向上しました。 • 監査対象イベントの定義で，オブジェクト種別（ROLE），セッション種別（ROLE）を指定できるようにしました。 • CURRENT_ROLE 値関数によって，現在使用しているロール名を検索できるようにしました。 • 埋込み型 UAP，ODBC から SET ROLE 文を発行できるようにしました。
HiRDB の適用 OS に Red Hat Enterprise Linux 10 を追加しました。

変更内容(3020-6-551-A0)　HiRDB Version 10 10-10

追加・変更内容
SELinux を enforcing モードにした環境で HiRDB を構築できるようにしました。これによって，セキュリティ要件の高いシステムでも HiRDB を使用できます。
表のアクセス権限を一括で管理できるロール機能をサポートしました。これによって，権限の管理や移行が容易になるだけでなく，アクセス制御の一貫性が向上し，セキュリティの強化にもつながります。
HiRDB の適用 OS に次の OS を追加しました。 • Windows Server 2025

変更内容(3020-6-551-90)　HiRDB Version 10 10-09

追加・変更内容
データベースの内容を複製した参照専用のデータベースを構築できるようにしました。 これによって，アプリケーション側で更新用 DB と参照用 DB（リードレプリカ）を使い分け，負荷分散及びシステム全体のスループット向上につながります。

はじめに

このマニュアルは，プログラムプロダクト　ノンストップデータベース　HiRDB Version 10 の機能の概要について説明したものです。

■ 対象読者

HiRDB Version 10（以降，**HiRDB** と表記します）を使ってリレーショナルデータベースシステムを構築または運用する方々を対象にしています。

このマニュアルの記述は，次に示す知識があることを前提にしています。

- Windows のシステム管理の基礎的な知識（Windows 版の場合）
- UNIX または Linux のシステム管理の基礎的な知識（UNIX 版の場合）
- SQL の基礎的な知識

■ 関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

HiRDB（UNIX 用マニュアル）

- HiRDB Version 10 システム導入・設計ガイド（UNIX(R)用）（3020-6-552）
- HiRDB Version 10 システム定義（UNIX(R)用）（3020-6-554）
- HiRDB Version 10 システム運用ガイド（UNIX(R)用）（3020-6-556）
- HiRDB Version 10 コマンドリファレンス（UNIX(R)用）（3020-6-558）
- インナレプリカ機能 HiRDB Staticizer Option Version 10（3020-6-563）
- HiRDB Version 10 ディザスタリカバリシステム　構築・運用ガイド（3020-6-564）
- HiRDB Version 10 構造型データベース機能（3020-6-578）
- HiRDB Version 10 構造型データベース機能（UAP 開発編）（3020-6-579）

HiRDB（Windows 用マニュアル）

- HiRDB Version 10 システム導入・設計ガイド（Windows(R)用）（3020-6-553）
- HiRDB Version 10 システム定義（Windows(R)用）（3020-6-555）
- HiRDB Version 10 システム運用ガイド（Windows(R)用）（3020-6-557）
- HiRDB Version 10 コマンドリファレンス（Windows(R)用）（3020-6-559）

HiRDB（Windows，UNIX 共通マニュアル）

- HiRDB Version 10 UAP 開発ガイド（3020-6-560）
- HiRDB Version 10 SQL リファレンス（3020-6-561）
- HiRDB Version 10 メッセージ（3020-6-562）
- HiRDB Version 10 XDM/RD E2 接続機能（3020-6-565）
- HiRDB Version 10 データベース暗号化機能（3020-6-566）
- HiRDB Version 10 バッチ高速化機能（3020-6-567）
- HiRDB Version 10 パフォーマンスガイド（3020-6-568）
- HiRDB XML 拡張機能 HiRDB XML Extension Version 10（3020-6-576）
- HiRDB 全文検索プラグイン HiRDB Text Search Plug-in Version 10（3020-6-577）
- HiRDB データ連動機能 HiRDB Datareplicator Version 10（3020-6-573）
- HiRDB データ連動拡張機能 HiRDB Datareplicator Extension Version 10（3020-6-574）
- データベース抽出・反映サービス機能 HiRDB Dataextractor Version 10（3020-6-575）

なお，本文中で使用している HiRDB Version 10 のマニュアル名は，Version 10，（UNIX(R)用），（Windows(R)用）を省略して表記しています。使用しているプラットフォームに応じて UNIX 用または Windows 用のマニュアルを参照してください。

関連製品

- JP1 Version 7i JP1/Integrated Manager - Console システム構築・運用ガイド（3020-3-F01）
- JP1 Version 7i JP1/Base（3020-3-F04）
- JP1 Version 7i JP1/Performance Management - Agent for HiRDB（3020-3-F61）
- JP1 Version 7i JP1/Performance Management/SNMP System Observer 拡張リソース管理編（3020-3-F70）
- JP1 Version 6 JP1/Base（3020-3-986）
- JP1 Version 6 JP1/Performance Management - Agent for HiRDB（3020-3-C68）
- JP1 Version 6 JP1/VERITAS NetBackup v4.5 Agent for HiRDB License（3020-3-D79）
- JP1 Version 5 システムイベントサービス JP1/System Event Service（3000-3-154）
- システムイベントサービス JP1/System Event Service（3000-3-080）
- 磁気テープマウント管理 JP1/Magnetic Tape Library（3000-3-575）
- 磁気テープ運用支援 JP1/Magnetic Tape Access（3000-3-578）
- 磁気テープ簡易アクセス法 EasyMT（3000-3-573）

- uCosminexus Grid Processing Server 使用の手引（3000-3-E05）
- OpenTP1 Version 6 分散アプリケーションサーバ TP1/LiNK 使用の手引（3000-3-951）
- OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1 システム定義（3000-3-D52）
- OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1 プログラム作成の手引（3000-3-D51）
- OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1 プログラム作成リファレンス COBOL 言語編（3000-3-D55）
- OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1 プログラム作成リファレンス C 言語編（3000-3-D54）
- 分散トランザクション処理機能 TP1/Server Base Enterprise Option 使用の手引（3000-3-982）
- トランザクション分散オブジェクト基盤 TPBroker ユーザーズガイド（3000-3-555）
- 高信頼化システム監視機能 HA モニタ AIX(R)編（3000-9-202）※
- 高信頼化システム監視機能 HA モニタ Linux(R)編（3000-9-132）※
- 高信頼化システム監視機能 HA モニタ Linux(R)（x86）編（3000-9-201）※
- 高信頼化システム監視機能 HA モニタ パブリッククラウド編（3000-9-204）
- Hitachi HA Toolkit（3000-9-115）
- COBOL85 使用の手引（3000-3-347）
- COBOL85 操作ガイド（3020-3-747）
- COBOL85 言語（3020-3-782）
- DBPARTNER2 Client 操作ガイド（3020-6-027）
- Hitachi Tuning Manager - Agent for RAID（3020-3-P44）
- Hitachi Tuning Manager - Storage Mapping Agent（3020-3-P45）
- VOS3 データマネジメントシステム XDM E2 系 XDM/RD E2 解説（6190-6-637）
- VOS3 データマネジメントシステム XDM E2 系 XDM/RD E2 SQL リファレンス（6190-6-656）
- VOS3 データマネジメントシステム XDM E2 系 メッセージ（XDM/RD E2）（6190-6-643）

注※

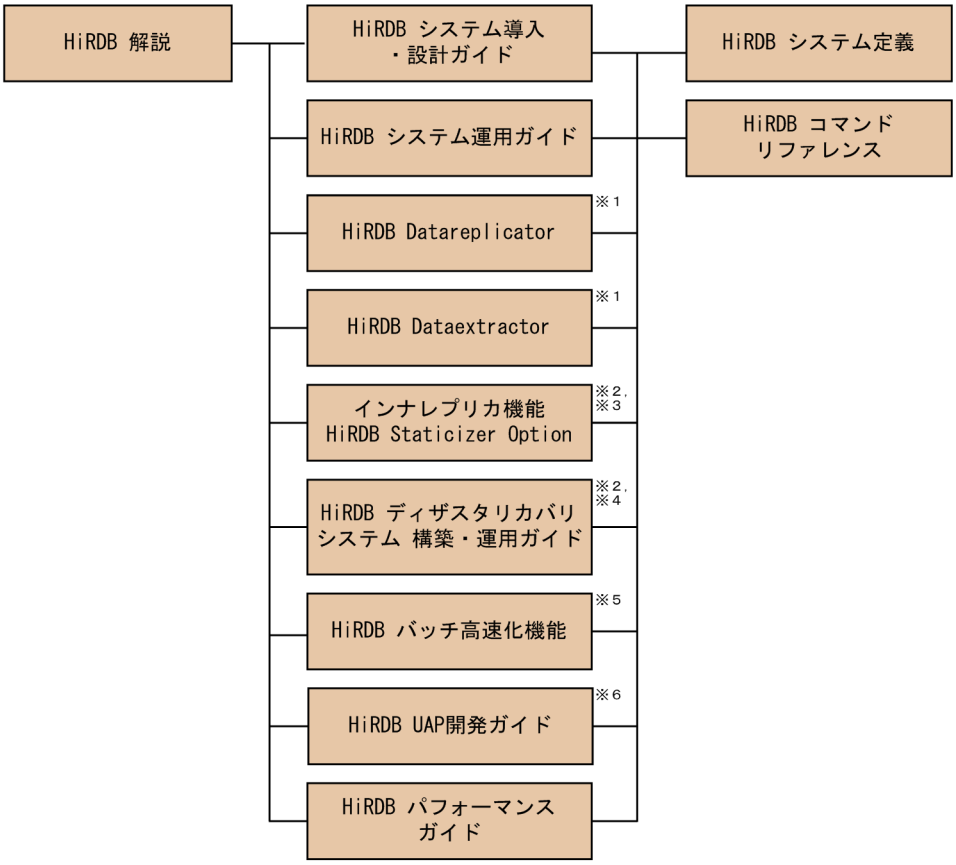
本文中で使用している HA モニタのマニュアル名は，AIX(R)編，Linux(R)編，および Linux(R)（x86）編を省略して表記しています。使用しているプラットフォームに応じて AIX 用または Linux 用のマニュアルを参照してください。

■ 利用者ごとの関連マニュアル

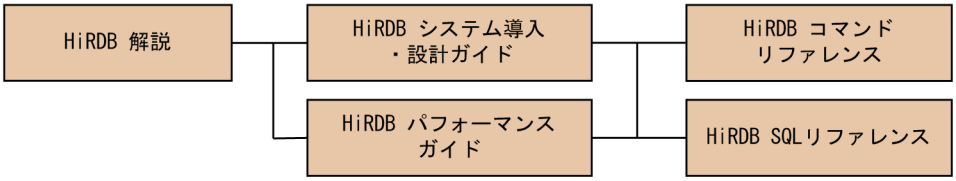
HiRDB のマニュアルをご利用になる場合，利用者ごとに次のようにお読みください。

また，より理解を深めるために，左側のマニュアルから順にお読みいただくことをお勧めします。

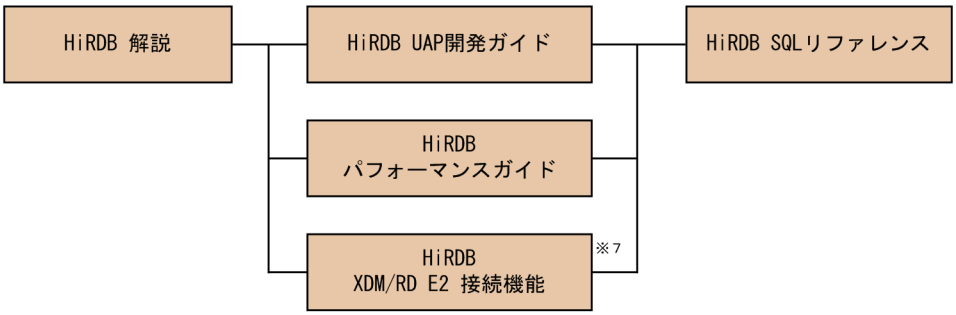
システム管理者が利用するマニュアル



表の作成者が利用するマニュアル



UAP作成者，およびUAP実行者が利用するマニュアル



注※１　レプリケーション機能を使用してデータ連携をする場合にお読みください。
注※２　UNIX用マニュアルです。Windows用はありません。
注※３　インナレプリカ機能を使用する場合にお読みください。
注※４　ディザスタリカバリシステムを構築する場合にお読みください。
注※５　インメモリデータ処理によるバッチ高速化を行う場合にお読みください。
注※６　OLTPシステムと連携する場合は必ずお読みください。
注※７　XDM/RD E2接続機能を使用して，XDM/RD E2のデータベースを操作する場合にお読みください。

■ このマニュアルでの表記

このマニュアルでは製品名称および名称について次のように表記しています。ただし，それぞれのプログラムについての表記が必要な場合はそのまま表記しています。

製品名称または名称	表記	
HiRDB Server Version 10	HiRDB/シングルサーバ	HiRDB または HiRDB サーバ
	HiRDB/パラレルサーバ	
HiRDB/Developer's Kit Version 10	HiRDB/Developer's Kit	HiRDB クライアント
HiRDB/Developer's Kit Version 10(64)		
HiRDB/Run Time Version 10	HiRDB/Run Time	
HiRDB/Run Time Version 10(64)		
HiRDB Accelerator Version 10	HiRDB Accelerator	
HiRDB Advanced High Availability Version 10	HiRDB Advanced High Availability	
HiRDB Control Manager	HiRDB CM	
HiRDB Control Manager Agent	HiRDB CM Agent	
HiRDB Dataextractor Version 10	HiRDB Dataextractor	
HiRDB Datareplicator Version 10	HiRDB Datareplicator	
HiRDB Disaster Recovery Light Edition Version 10	HiRDB Disaster Recovery Light Edition	
HiRDB Non Recover Front End Server Version 10	HiRDB Non Recover FES	
HiRDB Staticizer Option Version 10	HiRDB Staticizer Option	

製品名称または名称	表記
HiRDB Text Search Plug-in Version 9	HiRDB Text Search Plug-in
HiRDB XML Extension Version 9	HiRDB XML Extension
シングルサーバ	SDS
システムマネジャ	MGR
フロントエンドサーバ	FES
ディクショナリサーバ	DS
バックエンドサーバ	BES
Microsoft ActiveX	ActiveX
DataStage(R)	DataStage
JP1/Magnetic Tape Access	EasyMT
EasyMT	
JP1/Automatic Job Management System 2 - Scenario Operation	JP1/AJS2-SO
JP1/Automatic Job Management System 3	JP1/AJS3
JP1/Automatic Job Management System 2	
JP1/Cm2/Extensible SNMP Agent	JP1/ESA
JP1/Cm2/Extensible SNMP Agent for Mib Runtime	
JP1/NETM/Audit - Manager	JP1/NETM/Audit
JP1/Cm2/Network Node Manager	JP1/NNM
JP1/Performance Management	JP1/PFM
JP1/Performance Management - Agent Option for HiRDB	JP1/PFM-Agent for HiRDB
JP1/Performance Management - Agent Option for Platform	JP1/PFM-Agent for Platform
JP1/Performance Management/SNMP System Observer	JP1/SSO
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	
LifeKeeper for Linux v9.1	LifeKeeper
SIOS DataKeeper Cluster Edition	DataKeeper
Microsoft Office Excel	Microsoft Excel または Excel

製品名称または名称	表記	
DLT(TM)	DLT	
PowerBuilder(R)	PowerBuilder	
JP1/Magnetic Tape Library	MTguide	
JP1/VERITAS NetBackup BS v4.5	NetBackup	
JP1/VERITAS NetBackup v4.5		
PowerHA SystemMirror V7.1	PowerHA	
OpenTP1/Server Base Enterprise Option	TP1/EE	
Hitachi System Information Capture	HSIC	
Hitachi TrueCopy	TrueCopy	
Hitachi TrueCopy Asynchronous		
Hitachi TrueCopy basic		
Hitachi TrueCopy Software		
TrueCopy		
TrueCopy Asynchronous		
TrueCopy remote replicator		
Hitachi Universal Replicator Software	Universal Replicator	
Universal Replicator		
Microsoft .NET Framework	.NET Framework	
Microsoft Visual C++	Visual C++または C++言語	
Windows Server Failover Cluster	Microsoft Failover Cluster	MSFC
Microsoft Failover Cluster		
Microsoft Cluster Server		
Virtual-storage Operating System 3/Forefront System Product	VOS3/FS	VOS3
Virtual-storage Operating System 3/Leading System Product	VOS3/LS	
Virtual-storage Operating System 3/Unific System Product	VOS3/US	
Virtual-storage Operating System 3/eXtensible System Product	VOS3/XS	
VOS3 Database Connection Server	DB コネクションサーバ	
Extensible Data Manager/Base Extended Version 2 XDM 基本プログラム XDM/BASE E2	XDM/BASE E2	

製品名称または名称	表記	
XDM/Data Communication and Control Manager 3 XDM データコミュニケーションマネジメントシステム XDM/DCCM3	XDM/DCCM3	
XDM/Relational Database リレーショナルデータベースシステム XDM/RD	XDM/RD	XDM/RD
XDM/Relational Database Extended Version 2 リレーショナルデータベースシステム XDM/RD E2	XDM/RD E2	
AIX V7.2	AIX V7.2	AIX
AIX V7.3	AIX V7.3	
Red Hat Enterprise Linux 7 (64-bit x86_64)	Linux 7	Linux
Red Hat Enterprise Linux 8 (64-bit x86_64)	Linux 8	
Red Hat Enterprise Linux 9 (64-bit x86_64)	Linux 9	
Red Hat Enterprise Linux 10 (64-bit x86_64)	Linux 10	
Red Hat Enterprise Linux 7 (64-bit x86_64)	Linux (EM64T)	
Red Hat Enterprise Linux 8 (64-bit x86_64)		
Red Hat Enterprise Linux 9 (64-bit x86_64)		
Red Hat Enterprise Linux 10 (64-bit x86_64)		
Red Hat Enterprise Linux 7 (64-bit x86_64)	Linux 7 (64-bit x86_64)	Linux 7
Red Hat Enterprise Linux 8 (64-bit x86_64)	Linux 8 (64-bit x86_64)	Linux 8
Red Hat Enterprise Linux 9 (64-bit x86_64)	Linux 9 (64-bit x86_64)	Linux 9
Red Hat Enterprise Linux 10 (64-bit x86_64)	Linux 10 (64-bit x86_64)	Linux 10
Microsoft Windows Server 2019 Standard	Windows Server 2019 Standard	Windows Server 2019
Microsoft Windows Server 2019 Datacenter	Windows Server 2019 Datacenter	
Microsoft Windows Server 2022 Standard	Windows Server 2022 Standard	Windows Server 2022
Microsoft Windows Server 2022 Datacenter	Windows Server 2022 Datacenter	
Microsoft Windows Server 2025 Standard	Microsoft Windows Server 2025 Standard	Windows Server 2025
Microsoft Windows Server 2025 Datacenter	Microsoft Windows Server 2025 Datacenter	
Windows 10 Home		Windows 10

製品名称または名称	表記
Windows 10 Pro	
Windows 10 Enterprise	
Windows 10 Home (x64)	
Windows 10 Pro (x64)	
Windows 10 Enterprise (x64)	
Windows 11 Pro (x64)	Windows 11
Windows 11 Enterprise (x64)	
Windows 10 Home	Windows (x86)
Windows 10 Pro	
Windows 10 Enterprise	
Windows 10 Home (x64)	Windows (x64)
Windows 10 Pro (x64)	
Windows 10 Enterprise (x64)	
Windows 11 Pro (x64)	
Windows 11 Enterprise (x64)	

- Windows Server 2019，Windows Server 2022 および Windows Server 2025 を総称して Windows Server と表記します。また，Windows Server，Windows 10，および Windows 11 を総称して Windows と表記します。
- AIX および Linux を総称して UNIX と表記します。
- UNIX の場合，HiRDB 運用ディレクトリのパスを$PDDIR と表記します。
- Windows の場合，HiRDB 運用ディレクトリのパスを%PDDIR%と表記します。また，Windows のインストールディレクトリのパスを%windir%と表記します。
- TCP/IP が規定する hosts ファイル（UNIX の場合は/etc/hosts ファイルも含む）を hosts ファイルと表記します。また，Windows の場合は，通常，`%windir%¥system32¥drivers¥etc¥hosts` のことを指します。
- Linux の場合に使用できるブロック型スペシャルファイルは，キャラクタ型スペシャルファイルと同様の扱いになります。ブロック型スペシャルファイルを使用する場合は，本文中の「キャラクタ型スペシャルファイル」を「ブロック型スペシャルファイル」に読み替えてください。

■ このマニュアルで使用している略語

このマニュアルで使用している英略語の一覧を次に示します。

英略語	英字の表記
ACK	Acknowledgement
ACL	Access Control List
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
BMP	Basic Multilingual Plane
BOM	Byte Order Mark
CD-ROM	Compact Disc - Read Only Memory
CLOB	Character Large Object
CMT	Cassette Magnetic Tape
COBOL	Common Business Oriented Language
CORBA	Common ORB Architecture
CPU	Central Processing Unit
CSV	Comma Separated Values
DAO	Data Access Object
DAT	Digital Audio Tape
DB	Database
DBMS	Database Management System
DDL	Data Definition Language
DIC	Dictionary Server
DLT	Digital Linear Tape
DML	Data Manipulate Language
DNS	Domain Name System
DOM	Document Object Model

英略語	英字の表記
DS	Dictionary Server
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DWH	Data Warehouse
EBCDIK	Extended Binary Coded Decimal Interchange Kana
EJB	Enterprise JavaBeans
EUC	Extended UNIX Code
EX	Exclusive
FAT	File Allocation Table
FD	Floppy Disk
FES	Front End Server
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GUI	Graphical User Interface
HD	Hard Disk
HDP	Hitachi Dynamic Provisioning
HTML	Hyper Text Markup Language
ID	Identification number
IP	Internet Protocol
IPF	Itanium(R) Processor Family
JAR	Java Archive File
Java EE	Java Platform, Enterprise Edition
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

英略語	英字の表記
JPA	Java Persistence API
JRE	Java Runtime Environment
JSP	JavaServer Pages
JTA	Java Transaction API
JTS	Java Transaction Service
KEIS	Kanji processing Extended Information System
LAN	Local Area Network
LIP	Loop Initialization Process
LOB	Large Object
LRU	Least Recently Used
LTO	Linear Tape-Open
LU	Logical Unit
LVM	Logical Volume Manager
MGR	System Manager
MIB	Management Information Base
MRCF	Multiple RAID Coupling Feature
MSCS	Microsoft Cluster Server
MSFC	Microsoft Failover Cluster
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
OCI	Oracle Cloud Infrastructure
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
OLTP	On-Line Transaction Processing
OOCOBOL	Object Oriented COBOL

英略語	英字の表記
ORB	Object Request Broker
OS	Operating System
OTS	Object Transaction Service
PC	Personal Computer
PIC	Plug-in Code
PNM	Public Network Management
POSIX	Portable Operating System Interface for UNIX
PP	Program Product
PR	Protected Retrieve
PU	Protected Update
RAID	Redundant Arrays of Inexpensive Disk
RD	Relational Database
RDB	Relational Database
RDO	Remote Data Objects
ReFS	Resilient File System
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
SNTP	Simple Network Time Protocol
SR	Shared Retrieve
SU	Shared Update
TCP/IP	Transmission Control Protocol / Internet Protocol
TM	Transaction Manager

英略語	英字の表記
UAP	User Application Program
UOC	User Own Coding
VOS3	Virtual-storage Operating System 3
WS	Workstation
WSFC	Windows Server Failover Cluster
WWW	World Wide Web
XDM/BASE E2	Extensible Data Manager / Base Extended Version 2
XDM/DS	Extensible Data Manager / Data Spreader
XDM/RD E2	Extensible Data Manager / Relational Database Extended Version 2
XDM/XT	Extensible Data Manager / Data Extract
XML	Extensible Markup Language

■ ログの表記

●UNIX 版の場合

OS のログを syslogfile と表記します。syslogfile は，/etc/syslog.conf でログ出力先に指定しているファイルです。一般的には，次のファイルが syslogfile となります。

OS	ファイル
AIX	/var/adm/ras/syslog
Linux	/var/log/messages

●Windows 版の場合

Windows のイベントビューアで表示されるアプリケーションログをイベントログと表記します。イベントログは，次の方法で参照できます。

〈手順〉

1.［スタート］−［プログラム］−［管理ツール（共通）］−［イベントビューア］を選択します。

2.［ログ］−［アプリケーション］を選択します。

アプリケーションログが表示されます。「ソース」の列が「HiRDBSingleServer」または「HiRDBParallelServer」になっているのが HiRDB が出力したメッセージです。

なお，セットアップ識別子を指定してインストールした場合は，「HiRDBSingleServer」または「HiRDBParallelServer」にセットアップ識別子が付いた名称となります。

■ このマニュアルで使用している計算式の記号

このマニュアルで使用している計算式の記号の意味を次に示します。

記号	意味
↑↑	計算結果の値を小数点以下で切り上げることを示します。 （例）↑34÷3↑の計算結果は 12 となります。
↓↓	計算結果の小数点以下を切り捨てることを示します。 （例）↓34÷3↓の計算結果は 11 となります。
MAX または Max	計算結果の最も大きい値を選ぶことを示しています。 （例）MAX（3×6，4 ＋ 7）の計算結果は 18 となります。
MIN または Min	計算結果の最も小さい値を選ぶことを示しています。 （例）MIN（3×6，4 ＋ 7）の計算結果は 11 となります。
mod	mod（a，b）は，a を b で割った余りを示しています。 （例）mod（9，2）の計算結果は 1 となります。

■ Windows の操作説明での表記

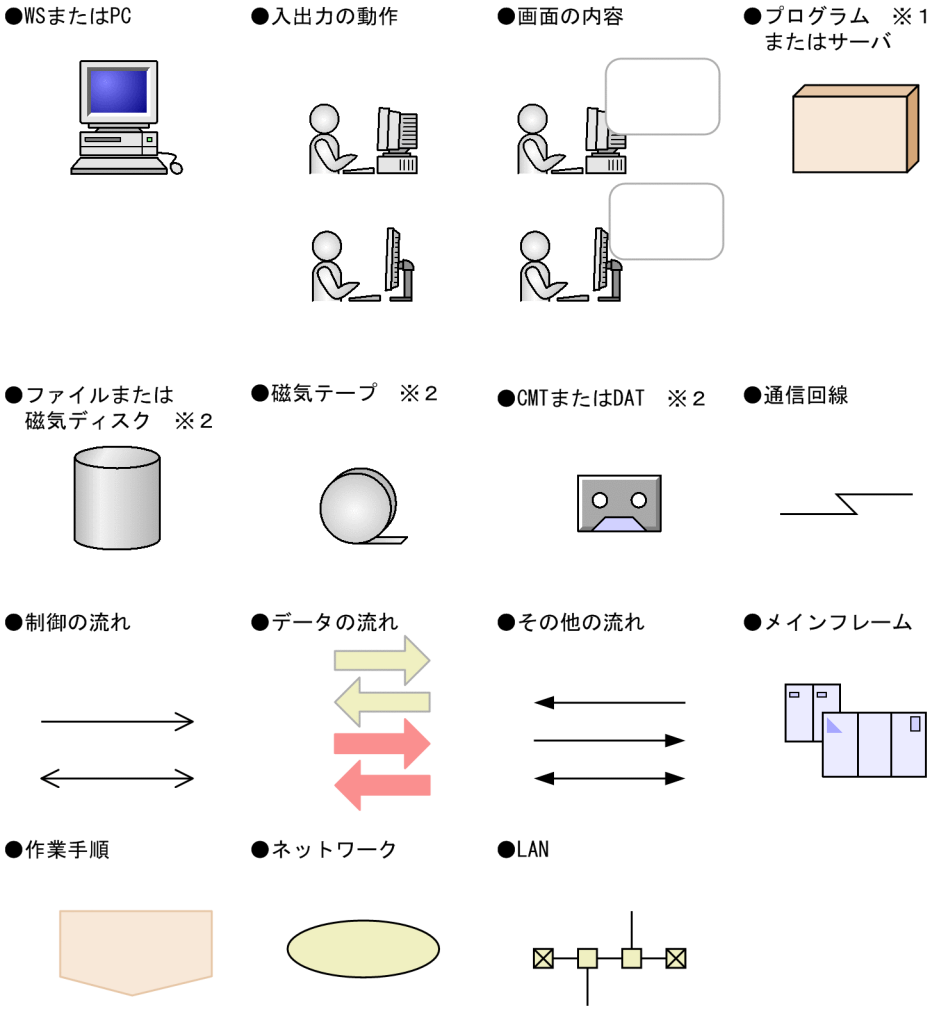
Windows の操作説明で使用している記号を次に示します。

記号	意味
[ ]	ボタンやテキストボックスなど，画面に表示されている要素を示します。
[ ] − [ ]	画面に表示されるメニューやアイコンなどを選択する操作を示します。

Windows の用語「ディレクトリ」と「フォルダ」は，「ディレクトリ」に統一して表記しています。

■ 図中で使用している記号

このマニュアルの図中で使用している記号を次のように定義します。



注※1

一部の図では，プログラムを単に四角で囲んで（影を付けないで）記載しています。

注※2

入力データファイル，アンロードファイルおよびバックアップファイルには，磁気ディスク装置のほかに磁気テープ装置，カセット磁気テープ装置（CMT）およびデジタルオーディオテープ装置（DAT）が使用できますが，このマニュアルでは磁気ディスク装置だけを記載しています。

■ UNIX の NFS（ネットワークファイルシステム）使用に関する注意

NFS については，ネットワークを介したファイルアクセスであるため，性能遅延，データ欠損などが発生する場合があり，信頼性の面から使用を推奨しません。

■ Windows のパス名に関する注意

- パス名を絶対パスで指定する場合はドライブ名を指定してください。

（例）C:¥win32app¥hitachi¥hirdb_s¥spool¥tmp

- コマンドの引数，制御文ファイル，および HiRDB システム定義ファイル中に空白または丸括弧を含むパス名を指定する場合は，前後を引用符（”）で囲んでください。

（例）pdinit -d ”C:¥Program Files(x86)¥hitachi¥hirdb_s¥conf¥mkinit”

ただし，バッチファイルもしくはコマンドプロンプト上で set コマンドを使用して環境変数を設定する場合，またはインストールディレクトリを指定する場合は引用符は不要です。引用符で囲むと，引用符も環境変数の値に含まれます。

（例）set PDCLTPATH=C:¥Program Files¥hitachi¥hirdb_s¥spool

- HiRDB はネットワークドライブのファイルを使用できないため，HiRDB のインストール，および環境構築はローカルドライブで行ってください。また，ユティリティの入出力ファイルなども，ローカルドライブ上のファイルを使用してください。
- パス名には，ショートパス名（例えば，C:¥PROGRA~1 など）は使用しないでください。

■ KB（キロバイト）などの単位表記について

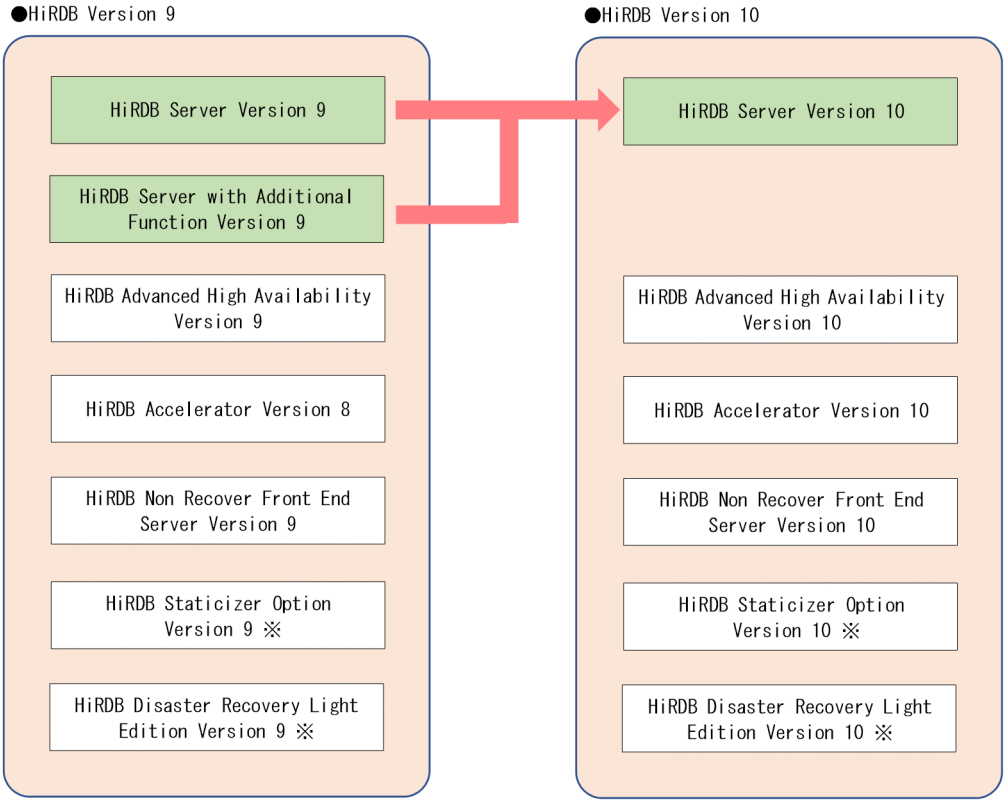
1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

■ HiRDB Version 9 と HiRDB Version 10 の製品体系の違い

HiRDB Version 10 では，製品体系を次のように変更しました。

- HiRDB Server with Additional Function を廃止し，HiRDB Server with Additional Function の機能を HiRDB Server に統合しました。

HiRDB Version 9 と HiRDB Version 10 の製品体系の違いを次に示します。



注※　UNIX 版でだけ使用できる製品です。

■ バージョン 09-50 以降のシステム定義のオペランド省略値表記について

バージョン 09-50 以降，システム定義のオペランドの省略値として，推奨値を省略時に仮定する推奨モードと，特定のバージョンの省略値を仮定する互換モードを選択できるようになりました。通常は推奨モードをお勧めしているため，本文中での推奨モードのオペランド省略値を下線付きで表記している個所があります。

目次

前書き　2
変更内容　4
はじめに　5

1　概要　31
1.1　HiRDB の特長　32
1.1.1　HiRDB システムの概要　32
1.1.2　HiRDB を導入するメリット　35
1.2　HiRDB システムの構成　38
1.2.1　HiRDB/シングルサーバの構成　38
1.2.2　HiRDB/パラレルサーバの構成　39
1.2.3　マルチ HiRDB の構成　42
1.3　データベースへのアクセス形態　43
1.3.1　HiRDB クライアント下の UAP を実行してデータベースにアクセスする形態　43
1.3.2　HiRDB サーバ上の UAP を実行してデータベースにアクセスする形態　43
1.3.3　OLTP の UAP を実行してデータベースにアクセスする形態　44
1.3.4　ODBC 対応のアプリケーションからデータベースにアクセスする形態　44
1.3.5　OLE DB 対応のアプリケーションからデータベースにアクセスする形態　45
1.3.6　JDBC 対応のアプリケーションからデータベースにアクセスする形態　46
1.3.7　ADO.NET 対応のアプリケーションからデータベースにアクセスする形態　47

2　HiRDB の付加 PP 及び関連製品との連携　49
2.1　HiRDB の付加 PP　50
2.1.1　HiRDB Advanced High Availability　50
2.1.2　HiRDB Staticizer Option【UNIX 版限定】　51
2.1.3　HiRDB Non Recover FES　55
2.1.4　HiRDB Disaster Recovery Light Edition【UNIX 版限定】　55
2.1.5　HiRDB Accelerator　56
2.2　データ連携製品との連携　57
2.2.1　HiRDB Datareplicator，HiRDB Dataextractor との連携　57
2.3　OLTP 製品との連携　62
2.3.1　連携できる OLTP 製品　62
2.3.2　HiRDB XA ライブラリ　62
2.3.3　HiRDB XA ライブラリが提供する機能　63
2.3.4　システムの構成　64

2.3.5 HiRDB のトランザクションマネジャへの登録 66
2.4 運用支援製品との連携 67
2.4.1 HiRDB SQL Executer 67
2.4.2 HiRDB Control Manager 68
2.4.3 HiRDB SQL Tuning Advisor 69
2.4.4 JP1/Performance Management - Agent Option for HiRDB 71
2.4.5 JP1/NETM/Audit 72
2.5 PC 上でのデータベースアクセス製品との連携 75
2.6 マルチメディア情報を扱う製品との連携 76
2.6.1 マルチメディア情報を扱う製品 76
2.7 Cosminexus との連携 79

3 データベースの論理構造 80
3.1 RD エリア 81
3.1.1 RD エリアの種類 81
3.1.2 RD エリアの作成方法 84
3.2 スキーマ 86
3.3 表 87
3.3.1 表の基本構造 87
3.3.2 表の正規化 89
3.3.3 FIX 属性 90
3.3.4 主キー（プライマリキー） 90
3.3.5 クラスタキー 90
3.3.6 サプレスオプション 91
3.3.7 ノースプリットオプション 91
3.3.8 表の横分割 92
3.3.9 表のマトリクス分割 100
3.3.10 表の分割格納条件の変更 104
3.3.11 改竄防止表 105
3.3.12 採番業務で使用する表 107
3.3.13 繰返し列 107
3.3.14 ビュー表 109
3.3.15 共用表 110
3.3.16 圧縮表 111
3.3.17 一時表 114
3.4 インデクス 118
3.4.1 インデクスの基本構造 118
3.4.2 インデクスの横分割 120
3.4.3 インデクスページスプリット 125

3.4.4 除外キー値 128
3.4.5 データを格納している表にインデクスを定義する場合 128
3.4.6 インデクスキー値無排他 128
3.5 オブジェクトリレーショナルデータベースへの拡張 133
3.5.1 抽象データ型 133
3.5.2 サブタイプと継承 139
3.5.3 隠蔽 144

4 データベースの物理構造 147
4.1 データベースの物理構造の仕組み 148
4.2 セグメントの設計 150
4.2.1 セグメントの状態 150
4.2.2 セグメントの設計方針 150
4.2.3 セグメント内の空きページ比率 150
4.2.4 セグメントの確保と解放 151
4.2.5 空き領域の再利用 152
4.3 ページの設計 155
4.3.1 ページの状態 155
4.3.2 ページの設計方針 155
4.3.3 ページ内の未使用領域の比率 155
4.3.4 ページの確保と解放 156

5 SQL によるデータベースアクセス 158
5.1 HiRDB で使える SQL 159
5.1.1 HiRDB の SQL の種類 159
5.1.2 SQL を実行する方法 159
5.2 データの基本操作 161
5.2.1 カーソル 161
5.2.2 データの検索 161
5.2.3 データの更新 162
5.2.4 データの削除 162
5.2.5 データの挿入 163
5.2.6 特定データの探索 164
5.2.7 データの演算 166
5.2.8 データの加工 166
5.2.9 抽象データ型を含む表データの操作 167
5.3 ストアドプロシジャ，ストアドファンクション 172
5.3.1 ストアドプロシジャ，ストアドファンクションの概要 172
5.4 Java ストアドプロシジャ，Java ストアドファンクション 178

5.4.1 Java ストアドプロシジャ，Java ストアドファンクションを使用できる環境 178
5.4.2 外部 Java ストアドルーチンの特長 178
5.4.3 システム構成（Java 仮想マシンの位置づけ） 179
5.4.4 外部 Java ストアドルーチンを実行するためには 180
5.4.5 外部 Java ストアドルーチンの実行手順 180
5.5 C ストアドプロシジャ，C ストアドファンクション 183
5.5.1 外部 C ストアドルーチンの特長 183
5.5.2 外部 C ストアドルーチンの実行手順 183
5.6 トリガ 186
5.6.1 トリガの概要 186
5.7 整合性制約 188
5.7.1 非ナル値制約 188
5.7.2 一意性制約 188
5.8 参照制約 189
5.8.1 参照制約の概要 189
5.9 検査制約 192
5.9.1 検査制約の概要 192
5.10 検査保留状態 193
5.10.1 検査保留状態の設定又は解除の契機 193
5.10.2 検査保留状態の表に対して制限される操作 193
5.11 データベースをアクセスする処理性能の向上 195
5.11.1 ブロック転送機能 195
5.11.2 グループ分け高速化機能 196
5.11.3 配列を使用した機能 197
5.11.4 ホールダブルカーソル 199
5.11.5 SQL の最適化 199
5.12 絞込み検索 204
5.13 自動採番機能 207
5.14 DB アクセス部品を使用したデータベースアクセス 208
5.14.1 ODBC ドライバ 208
5.14.2 HiRDB OLE DB プロバイダ 208
5.14.3 HiRDB データプロバイダ for .NET Framework 208
5.14.4 JDBC ドライバ 209
5.14.5 SQLJ 209

6 HiRDB のアーキテクチャ 210

6.1 HiRDB の環境設定 211
6.2 HiRDB ファイルシステム領域 213
6.2.1 HiRDB ファイルシステム領域の概要 213

6.3 システムファイル 217
6.3.1 システムログファイル 217
6.3.2 シンクポイントダンプファイル 218
6.3.3 ステータスファイル 220
6.3.4 システムファイルの構成単位 221
6.4 作業表用ファイル 224
6.4.1 作業表用ファイルの概要 224
6.5 HiRDB システム定義 226
6.5.1 HiRDB/シングルサーバの HiRDB システム定義の体系 226
6.5.2 HiRDB/パラレルサーバの HiRDB システム定義の体系 227
6.5.3 HiRDB システム定義ファイルの作成方法 228
6.5.4 システム構成変更コマンド（pdchgconf コマンド） 229
6.6 HiRDB の開始・終了 230
6.6.1 開始モードと終了モード 230
6.6.2 HiRDB の自動開始 232
6.6.3 縮退起動（HiRDB/パラレルサーバ限定） 232
6.7 ディレードリラン 233
6.8 データベースのアクセス処理方式 235
6.8.1 グローバルバッファ 235
6.8.2 インメモリデータ処理 238
6.8.3 プリフェッチ機能 240
6.8.4 非同期 READ 機能 241
6.8.5 デファードライト処理 241
6.8.6 デファードライト処理の並列 WRITE 機能 241
6.8.7 コミット時反映処理 242
6.8.8 グローバルバッファの LRU 管理方式 242
6.8.9 スナップショット方式によるページアクセス 243
6.8.10 グローバルバッファの先読み入力 244
6.8.11 ローカルバッファ 245
6.8.12 BLOB データのファイル出力機能 247
6.8.13 BLOB データ，BINARY データの部分的な更新・検索 250
6.8.14 位置付け子機能 253
6.9 トランザクション制御 255
6.9.1 HiRDB への接続と切り離し 255
6.9.2 複数接続機能 255
6.9.3 トランザクションの開始と終了 256
6.9.4 コミットとロールバック 256
6.9.5 OLTP 環境での UAP のトランザクション管理 260
6.9.6 自動再接続機能 260

6.10 排他制御 261
6.10.1 排他制御の単位 261
6.10.2 排他制御モード 262
6.10.3 HiRDB による自動的な排他制御 263
6.10.4 ユーザの設定による排他制御の変更 263
6.10.5 排他制御の期間 264
6.10.6 デッドロック 264
6.11 データベースの更新ログを取得しないときの運用 265
6.11.1 データベースの更新ログを取得しないときの運用方法 265

7 データベースの管理 268
7.1 データベースの回復 269
7.1.1 データベース回復の概要 269
7.1.2 データベースを回復できる時点 269
7.2 データベースの障害に備えた運用 271
7.2.1 バックアップの取得 271
7.2.2 システムログのアンロード（アンロードログファイルの作成） 273
7.2.3 差分バックアップ機能 276
7.2.4 バックアップ閉塞 278
7.2.5 ユーザ LOB 用 RD エリアのバックアップ取得時間の短縮（更新凍結コマンド） 279
7.2.6 NetBackup 連携機能 281
7.3 表及びインデクスの再編成 283
7.3.1 表の再編成 283
7.3.2 インデクスの再編成 287
7.4 使用中空きページ及び使用中空きセグメントの再利用 290
7.4.1 使用中空きページの再利用 290
7.4.2 使用中空きセグメントの再利用 293
7.5 RD エリアの追加，拡張，及び移動 294
7.5.1 RD エリアの追加 294
7.5.2 RD エリアの拡張 294
7.5.3 RD エリアの自動増分 294
7.5.4 RD エリアの移動（HiRDB/パラレルサーバ限定） 300
7.6 空白変換機能 302
7.6.1 空白変換レベルの概要 302
7.7 DECIMAL 型の符号正規化機能 305
7.7.1 符号付きパック形式の符号部の仕様 305
7.7.2 符号付きパック形式の符号部の変換規則 305
7.7.3 適用基準 306
7.7.4 環境設定 306

8 障害対策に関する機能　307
8.1 系切り替え機能　308
8.1.1 系切り替え機能とは　308
8.1.2 系切り替え機能の運用方法　315
8.1.3 系切り替え機能の形態　316
8.1.4 システム構成例　317
8.1.5 系の切り替え時間を短縮する機能（ユーザサーバホットスタンバイ，高速系切り替え機能）　323
8.1.6 系切り替え失敗時の自動再起動機能【UNIX 版限定】　324
8.1.7 系切り替えの実行時間監視機能【UNIX 版限定】　325
8.2 回復不要 FES　327
8.2.1 回復不要 FES とは　327
8.2.2 回復不要 FES を使用したシステムの構成例　328
8.3 リアルタイム SAN レプリケーション（ディザスタリカバリ）【UNIX 版限定】　329
8.3.1 リアルタイム SAN レプリケーションとは　329
8.3.2 前提プラットフォーム及び前提製品　331
8.3.3 リモートサイトへのデータ反映方式　332
8.3.4 全同期方式　332
8.3.5 全非同期方式　333
8.3.6 ハイブリッド方式　334
8.3.7 ログ同期方式　336

9 セキュリティ対策に関する機能　339
9.1 機密保護機能　340
9.1.1 ユーザ権限の種類　340
9.1.2 機密保護機能の運用方法　342
9.2 セキュリティ監査機能　343
9.2.1 セキュリティ監査機能とは　343
9.2.2 監査対象になるイベント　348
9.3 CONNECT 関連セキュリティ機能　353
9.3.1 CONNECT 関連セキュリティ機能とは　353
9.3.2 パスワードの文字列制限　353
9.3.3 連続認証失敗回数の制限　355
9.4 HiRDB 接続時のパスワード秘匿化機能　356
9.4.1 HiRDB 接続時のパスワード秘匿化機能の概要　356
9.4.2 HiRDB 接続時のパスワード秘匿化機能の適用範囲　356
9.5 IP アドレスによる接続制限　357
9.6 パスワードの有効期間　359

10 プラグイン 360
10.1 HiRDB のプラグインの概要 361
10.2 プラグインの業務への適用 362
10.3 HiRDB のプラグインの機能 363
10.3.1 全文検索プラグイン（HiRDB Text Search Plug-in） 363
10.3.2 空間検索プラグイン（HiRDB Spatial Search Plug-in） 364
10.3.3 HiRDB XML Extension 364
10.4 プラグイン使用時に HiRDB で設定する項目 366
10.4.1 プラグインのセットアップ／登録 366
10.4.2 レジストリ機能の初期設定 366
10.4.3 プラグイン使用時の表の定義 367
10.4.4 プラグインインデクスの遅延一括作成 367

付録 369
付録 A 今回のサポート項目一覧 370
付録 A.1 10-12 370
付録 B プラットフォームごとの HiRDB の機能差 372
付録 C データディクショナリ表 375
付録 D HiRDB クライアントと HiRDB サーバの接続可否 378
付録 E サポートを終了した機能の一覧 381
付録 F 用語解説 382

索引 433

1 概要

この章では，HiRDB の特長，システム構成，アクセス形態，HiRDB の付加 PP，及び HiRDB の関連製品について説明します。

1.1　HiRDB の特長

HiRDB とは，業務の規模に応じたリレーショナルデータベースを構築できるようにする，データベース管理システム（DBMS）の製品です。

HiRDB は，独立して動作できるサーバマシンをネットワークで相互結合して接続できます。また，一つのプロセサが一つのディスク上のデータベースだけを処理する方式（**Shared Nothing 方式**）を採用しているため，高い柔軟性を持ち，サーバマシン 1 台で動作するシングルノード構成からサーバマシン複数台で動作する並列プロセサ構成まで，幅広いアーキテクチャに適用できます。さらに，後からサーバマシンを追加してシステムを拡張するなど，スケーラブルなシステム構築ができるようにしています。

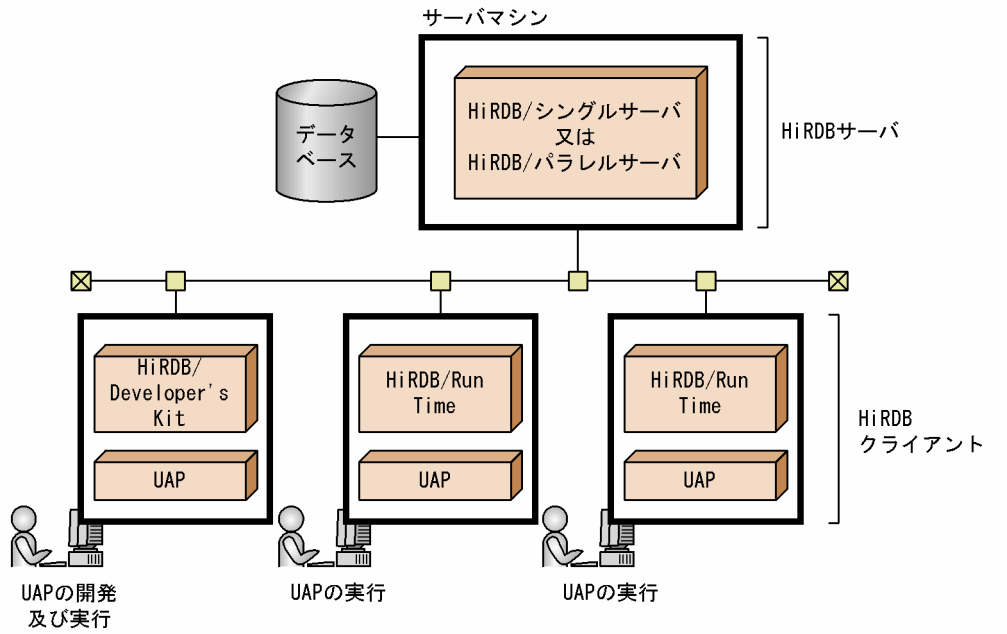
HiRDB を並列プロセサ構成にした場合，データの検索や更新の要求を内部で並列に実行でき，大きなスループットと迅速なターンアラウンドを実現しています。

また，顧客主導型の経営を実現するために必要なデータウェアハウスの構築にも HiRDB とミドルウェア群（DataStage など）で対応しています。

1.1.1　HiRDB システムの概要

HiRDB はクライアント／サーバシステムのネットワーク環境で使用します。データベースを配置するサーバ側システムを **HiRDB サーバ**，UAP を開発及び実行するクライアント側システムを **HiRDB クライアント**といいます。また，HiRDB サーバと HiRDB クライアントをまとめて **HiRDB システム**といいます。HiRDB システムの構成を次の図に示します。

図 1–1　HiRDB システムの構成



(1) HiRDB サーバ

HiRDB サーバには，**HiRDB/シングルサーバ**と **HiRDB/パラレルサーバ**があります。システム形態又は業務内容に合わせてどちらかを選択してください。HiRDB サーバの稼働プラットフォームは次のどれかになります。

- AIX
- Linux
- Windows

(a) HiRDB/シングルサーバ

HiRDB/シングルサーバは 1 台のサーバマシンでデータベースシステムを構築するときに使用します。処理性能が安定しており，運用も HiRDB/パラレルサーバに比べてシンプルであるため，小中規模のデータベースを構築するのに適したシステムです。

(b) HiRDB/パラレルサーバ

HiRDB/パラレルサーバを使用すると，複数のサーバマシンを一つのデータベースシステムとして構築でき，一つの表を複数のサーバマシンに分割して格納できます。検索処理を各サーバで並行して実行できるため，処理性能を向上できます。また，大量データの追加又は更新や，データベースのバックアップも並列に処理できるので，データベースが大規模化しても性能を維持できます。

マシン性能を十分に活用できる Shared Nothing 方式を採用しているので，データ量の増加に対してもサーバ台数を増やすことで安定した性能を維持できます。また，ソートやジョインなどの負荷の掛かる処理は空いている別のサーバへ処理を代行させて，特定のサーバへの負荷が集中しないようにバランスの取れた並列処理を実現しています。

(2) HiRDB クライアント

HiRDB クライアントには **HiRDB/Developer's Kit** と **HiRDB/Run Time** があります。HiRDB クライアントの稼働プラットフォームは次のどれかになります。

- AIX
- Linux
- Windows

(a) HiRDB/Developer's Kit

HiRDB/Developer's Kit は，UAP の開発（プリプロセス，コンパイル及びリンケージ）及び実行に必要なプログラムです。UAP の開発言語には，C，C++，COBOL85，OOCOBOL，COBOL 2002，及び Java（SQLJ）を使用できます。

なお，HiRDB サーバには HiRDB/Developer's Kit の機能が含まれています。したがって，HiRDB サーバで UAP を開発及び実行する場合は HiRDB/Developer's Kit は不要ですが，クライアント上で UAP を開発及び実行する場合は必要になります。

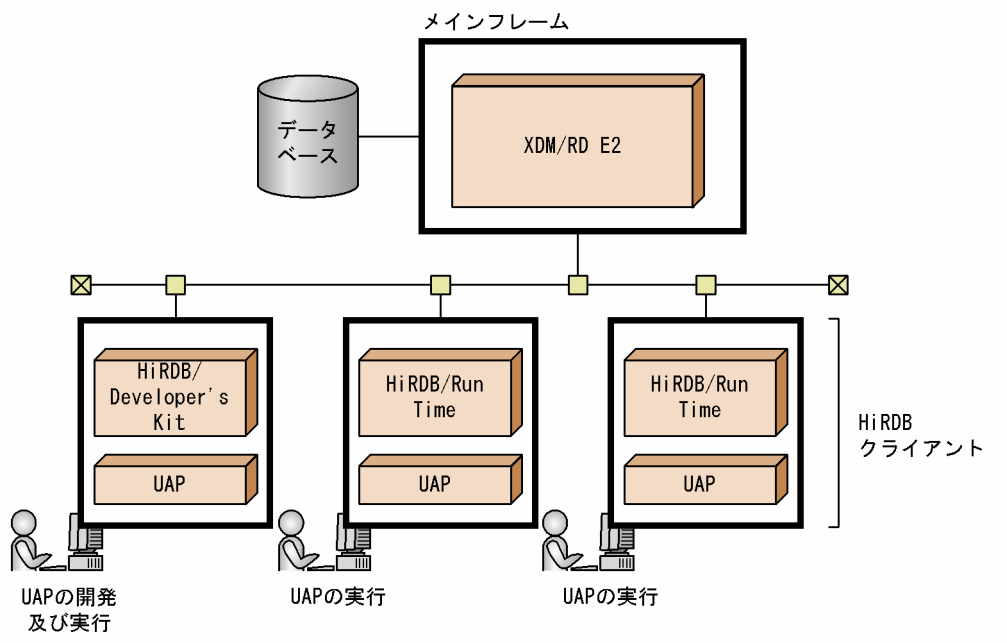
(b)　HiRDB/Run Time

HiRDB/Run Time は，作成した UAP を実行するだけのランタイムプログラムです。HiRDB/Run Time では，UAP の開発（プリプロセス，コンパイル及びリンケージ）はできません。UAP の実行だけができます。

(c)　HiRDB クライアントから XDM/RD E2 への接続

HiRDB クライアントで XDM/RD E2 のデータベースにアクセスする UAP の開発及び実行ができます。その UAP を使用して HiRDB クライアントから XDM/RD E2 のデータベースにアクセスできます。これを **XDM/RD E2 接続機能**といいます。XDM/RD E2 接続機能の概要を次の図に示します。

図 1–2　XDM/RD E2 接続機能の概要



〔説明〕

XDM/RD E2 接続機能を使用すると，HiRDB クライアント下で実行する UAP から XDM/RD E2 のデータベースを直接アクセスできます。

PC 上の UAP で ODBC 関数などを使用すれば XDM/RD E2 にアクセスできますが，適用言語の制限などがあります。XDM/RD E2 接続機能を使用すると，UAP 中に SQL を直接記述できるため，多様な処理ができるようになり，UAP の開発効率も向上します。

XDM/RD E2 接続機能については，マニュアル「HiRDB XDM/RD E2 接続機能」を参照してください。

1.1.2　HiRDB を導入するメリット

HiRDB は並列一括更新と並列リカバリ技術を実装しているリレーショナルデータベースです。HiRDB の特長を次に示します。

(1)　スケーラビリティに優れている

HiRDB には，サーバマシン 1 台で動作する HiRDB/シングルサーバと，複数台のサーバマシンで動作する HiRDB/パラレルサーバがあります。業務の特長によって，HiRDB/シングルサーバにするか，HiRDB/パラレルサーバにするかを選択できます。さらに，HiRDB では，HiRDB/シングルサーバを HiRDB/パラレルサーバに変更したり，HiRDB/パラレルサーバのサーバマシンを増やしたりできます。これによって，業務の規模の拡大に合わせて段階的にシステムを拡張できます。

HiRDB では，並列処理に適した Shared Nothing 方式を採用しているため，プロセサ数に比例して，処理性能を向上できます。つまり，増やしたサーバマシンの台数分だけ，処理性能を向上できます。サーバマシンを追加した場合にフレキシブルハッシュ分割機能を使用すると，HiRDB によってハッシュ関数が自動的に変更され，追加したサーバマシンにも自動的にデータが格納されます。

(2)　高性能システムの実現

(a)　並列処理による性能向上

表の並列検索・更新

データベースの検索又は更新処理を複数のサーバマシンに分散でき，表のデータも各サーバマシンに分割できます。このため，表を並列に検索又は更新でき，サーバマシンの台数分だけ処理性能が向上します。

負荷の高いデータベース処理の分散

HiRDB では，ソートやジョインなどの負荷が高い処理を別のサーバマシンに割り当て，データへのアクセスと並列に処理できます。これによって，検索結果の出力までの時間を短縮できます。

大量データのバッチ処理時間の短縮

HiRDB では，システム構築時などの大量データの格納も並列処理できるため，処理時間を短縮できます。

データベースの再編成の並列化

HiRDB では，データベースの再編成をサーバごとに並列に実行できるため，再編成の処理に掛かる時間を短縮できます。

並列バックアップ／リカバリ処理

HiRDB では，一つのコマンドの実行によって，バックアップの取得や，障害発生時のデータベースの回復処理をサーバごとに並列に実行できます。このため，バックアップ又はデータベースの回復に掛かる時間を短縮できます。

(b)　グローバルバッファによるきめ細かなバッファ制御

HiRDB では，アクセス頻度が高いインデクスのデータなどを専用のバッファに割り当てられます。これによって，インデクス検索やデータの全件検索などの様々な処理が混在する環境でも，バッファ間の干渉をなくし，安定したレスポンスを確保できます。

(c)　シンクポイントダンプ処理方式による性能の向上

HiRDB では，ある時点までの更新情報をデータベースに反映し，その時点での回復情報をファイルに取得しています。この処理を**シンクポイントダンプ処理**といいます。従来のシステムでは，シンクポイントダンプ処理中はトランザクションの受け付けが中断されるため，処理性能が低下していました。HiRDB はシンクポイントダンプ処理中でもトランザクションの受け付けを制限しないため，シンクポイントダンプ処理が原因で処理性能が低下しないようにしています。

(d)　高速リランによる短時間でのシステム回復

HiRDB では定期的なシンクポイントダンプ処理によって，障害時の回復範囲を小さくし，短時間で回復処理を完了できます。

また，システム回復時には，ロールバックと新規トランザクションの受け付けを同時に開始する**ディレードリラン方式**を採用しているので，速やかにシステムを再開始できます。

(3)　高信頼システムの実現

(a)　システムやデータベースの回復に必要な情報のファイルへの取得

システム再開始時に必要な情報の取得

HiRDB では障害が発生した場合に備えて，HiRDB 稼働中に，再開始に必要な情報（システムステータス情報）をファイルに格納します。このファイルを**ステータスファイル**といいます。

データベース回復時に必要な情報の取得

HiRDB では障害が発生した場合に備えて，データベースを回復するときに必要なデータベースの更新履歴情報（**システムログ**）をファイルに格納します。このファイルを**システムログファイル**といいます。システムログファイルによって，障害が発生した場合でも障害直前の状態にデータベースを正しく回復できます。

ファイルの二重化

HiRDB ではステータスファイル及びシステムログファイルを二重化して，システム及びデータベースの回復に必要な情報を取得できます。二重化することで，片方のファイルに異常が発生してももう一方のファイルを使用できるため，システムの信頼性を向上できます。

(b)　システムの自動再開始

軽度の障害であれば，HiRDB ではステータスファイルを使用して自動的にシステムを再開始します。このとき，オペレータの操作は不要です。

(c)　系切り替え機能による不稼働時間の短縮

業務処理中のサーバマシンとは別に待機用のサーバマシンを準備すれば，業務処理中のサーバマシンに障害が発生した場合にも，スムーズに待機用のサーバマシンに業務処理を切り替えられます。これを**系切り替え機能**といいます。

(4)　運用・操作性の向上

(a)　特定のサーバマシンからの集中管理

HiRDB/パラレルサーバの場合，特定のサーバマシンからシステムを一元的に集中管理できます。例えば，あるサーバマシンでコマンド又はユティリティを実行して，すべてのサーバマシン又は特定のサーバマシン上の HiRDB を開始又は終了できます。

(b)　HiRDB の環境設定支援【Windows 版限定】

Windows 版の HiRDB では，環境設定を支援するツールを提供しています。提供しているツールを次に示します。

- **簡易セットアップツール**
  GUI で HiRDB の環境を設定できます。運用ディレクトリ及びセットアップディレクトリを指定すれば容易に環境設定ができる「標準セットアップ」と，より詳細な設定ができる「カスタムセットアップ」のどちらかを選択できます。また，作成又は編集したシステム定義を更新できます。
- **バッチファイル**
  バッチファイルを実行すると，基本的な HiRDB の環境を自動的に設定できます。

(5)　オープンな環境での柔軟なシステムの実現

(a)　X/Open の XA インタフェースの装備

HiRDB では，X/Open の XA インタフェースを使用して OLTP と連携できます。HiRDB のトランザクション処理をトランザクションマネジャで制御するために，HiRDB XA ライブラリを提供しています。

(b)　ODBC，JDBC，OLE DB インタフェースの装備

HiRDB は業界標準の ODBC，JDBC，OLE DB に対応しているので，ODBC，JDBC，OLE DB に従ったアプリケーションを使用できます。また，ADO（ADO.NET にも対応），DAO，及び RDO も使用できます。

(6)　ノンストップサービスへの対応

ネットビジネスが盛んになり，24 時間 365 日ノンストップでオンライン業務を行いたいというニーズが増えています。HiRDB は 24 時間連続稼働を想定した機能を提供しています。24 時間連続稼働を想定した機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

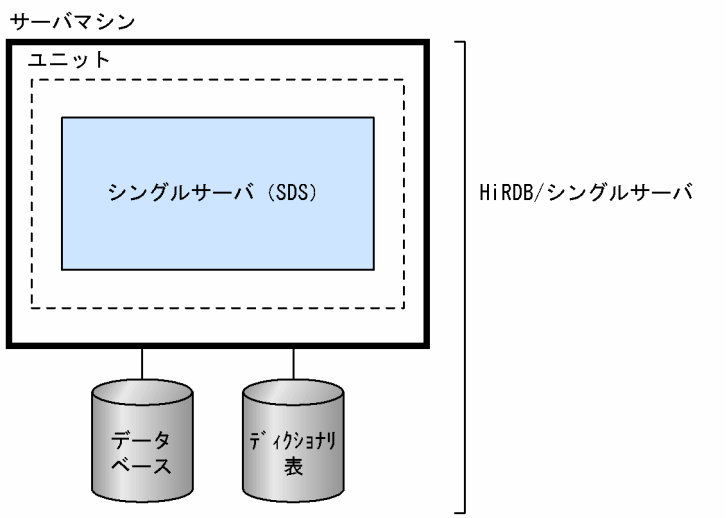
1.2　HiRDB システムの構成

ここでは，HiRDB/シングルサーバ，HiRDB/パラレルサーバ，及びマルチ HiRDB の構成について説明します。

1.2.1　HiRDB/シングルサーバの構成

HiRDB/シングルサーバは 1 ユニット（1 シングルサーバ）で構成されます。HiRDB/シングルサーバの構成を次の図に示します。

図 1–3　HiRDB/シングルサーバの構成



(1)　ユニット

HiRDB/シングルサーバは，次に示すサーバから構成されます。

- シングルサーバ

ユニットはサーバの実行制御及び監視をします。概念的には，ユニットとはサーバを格納する器のようなものです。

(2)　シングルサーバ（SDS）

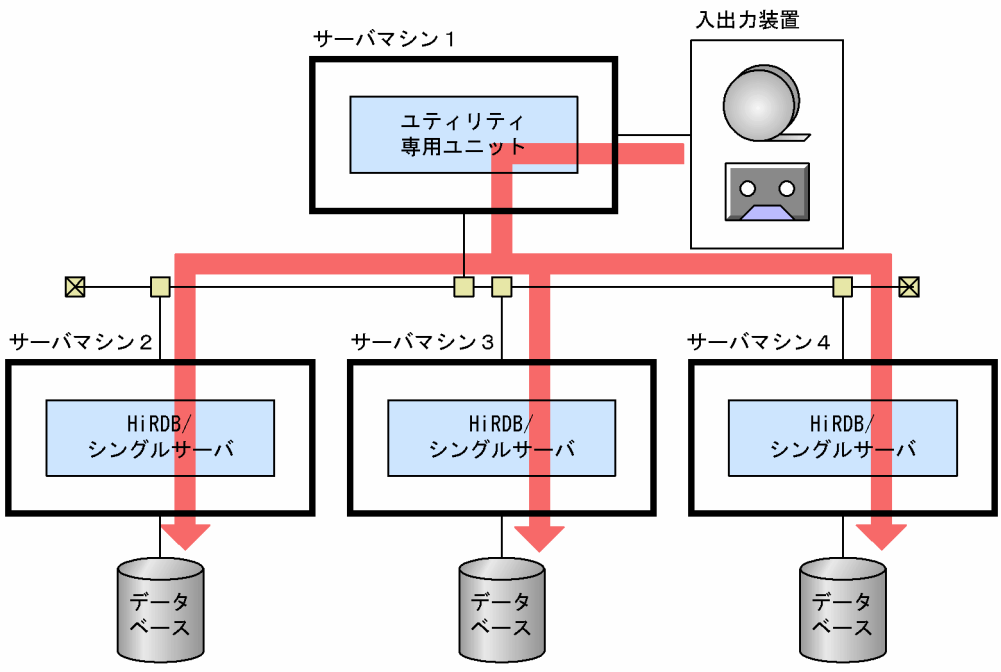
シングルサーバとは，データベース（表及びインデクス），データベースの情報を持つデータディクショナリ（ディクショナリ表）を管理するサーバのことです。

(3)　ユティリティ専用ユニット【UNIX 版限定】

HiRDB/シングルサーバがあるサーバマシンに CMT や磁気テープなどの入出力装置がない場合，入出力装置があるサーバマシンを**ユティリティ専用ユニット**として使用できます。例えば，複数の HiRDB/シングルサーバを使用していて，入出力装置があるサーバマシンが限られている場合にユティリティ専用ユニット

を設定します。ユティリティ専用ユニットを設定する場合の，HiRDB/シングルサーバのシステム構成例を次の図に示します。

図 1–4　HiRDB/シングルサーバのシステム構成例（ユティリティ専用ユニットを設置する場合）



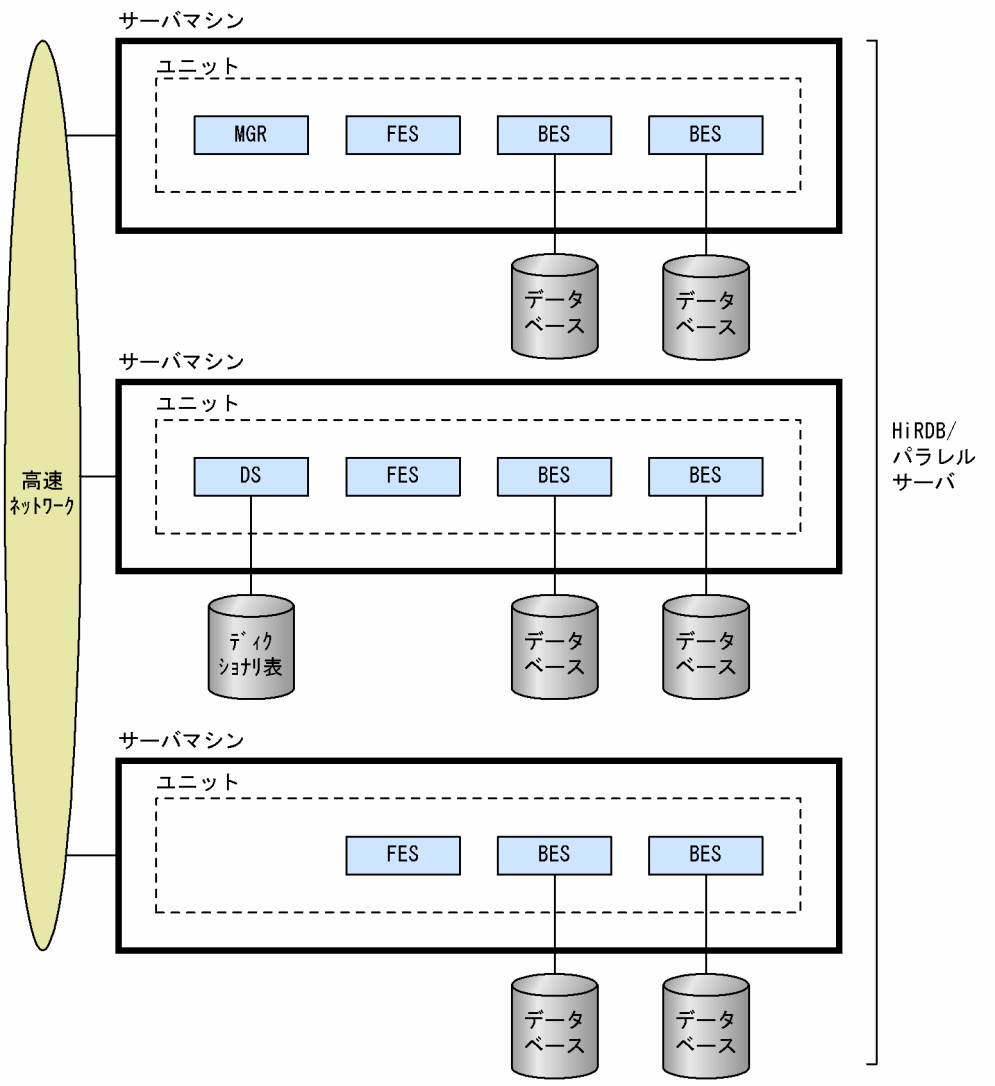
〔説明〕

- サーバマシン 2～4 には，CMT や磁気テープなどの入出力装置がありません。サーバマシン 1 に入出力装置があるので，サーバマシン 1 をユティリティ専用ユニットとします。
- CMT 又は磁気テープ中のソースデータをデータベースに格納する場合，サーバマシン 1（ユティリティ専用ユニット）の入出力装置を使用します。

1.2.2　HiRDB/パラレルサーバの構成

HiRDB/パラレルサーバは複数のユニット（複数のサーバ）で構成されます。HiRDB/パラレルサーバの構成を次の図に示します。

図 1–5　HiRDB/パラレルサーバの構成



〔説明〕

- この HiRDB/パラレルサーバは 3 台のサーバマシンで構成される例です。
- フロントエンドサーバを複数設置するマルチフロントエンドサーバにしています。
- バックエンドサーバは各サーバマシンに二つ設置しています。

(1)　ユニット

HiRDB/パラレルサーバは次に示すサーバから構成されます。

- システムマネジャ
- フロントエンドサーバ
- ディクショナリサーバ
- バックエンドサーバ

ユニットはサーバの実行制御，監視，及びサーバ間通信を管理します。概念的にはユニットとはサーバを格納する器のようなものです。

(2) システムマネジャ（MGR）

システムマネジャとは HiRDB の開始及び終了処理を制御するサーバです。また，システム構成情報の管理やサーバの障害の検出などもします。システムマネジャはシステムで一つ必要になります。

(3) フロントエンドサーバ（FES）

フロントエンドサーバとはデータベースへのアクセス方法を決定し，バックエンドサーバに実行内容を指示するサーバです。また，SQL の解析処理，SQL の最適化処理，各バックエンドサーバへ処理の指示，検索結果の編集処理などもしています。

フロントエンドサーバはシステムで一つ以上（最大 1,024 個）必要になります。複数のフロントエンドサーバを設置する形態を**マルチフロントエンドサーバ**といいます。SQL 処理の CPU 負荷が高く，一つのフロントエンドサーバで処理しきれない場合にマルチフロントエンドサーバにします。マルチフロントエンドサーバにすると，フロントエンドサーバが稼働するマシンの処理負荷を分散できます。

(4) ディクショナリサーバ（DS）

ディクショナリサーバとはデータベースの定義情報であるデータディクショナリ（ディクショナリ表）を一括管理するサーバです。ディクショナリサーバはシステムで一つ必要になります。

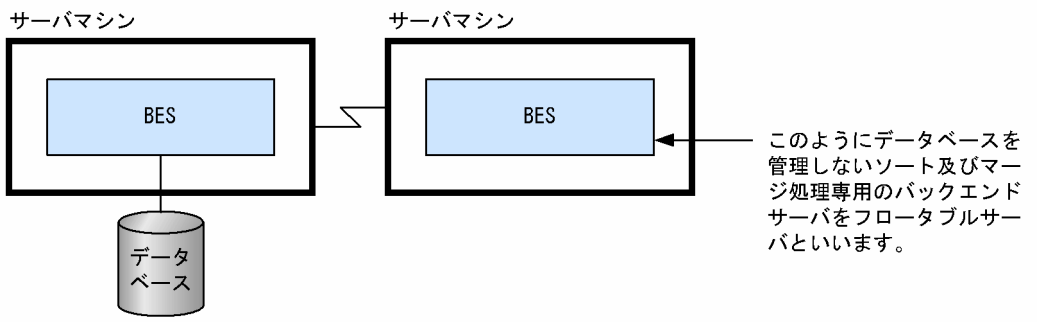
(5) バックエンドサーバ（BES）

バックエンドサーバとはデータベースを管理するサーバです。バックエンドサーバは，フロントエンドサーバからの実行指示に従って，データベースのアクセス，排他制御，演算処理などをします。また，検索結果に対してソート，マージ及び結合処理もします。

バックエンドサーバはシステムで一つ以上（最大 16,382 個）必要になります。バックエンドサーバを複数設定して，一つの表を複数のバックエンドサーバに分割して管理できます。

性能を向上させたい場合は，HiRDB/パラレルサーバ内に処理の負荷が高いソートやジョイン専用のバックエンドサーバ（データベースを管理しないバックエンドサーバ）を設定します。このようなバックエンドサーバを**フロータブルサーバ**といいます。フロータブルサーバを次の図に示します。

図 1–6　フロータブルサーバ



1.2.3　マルチ HiRDB の構成

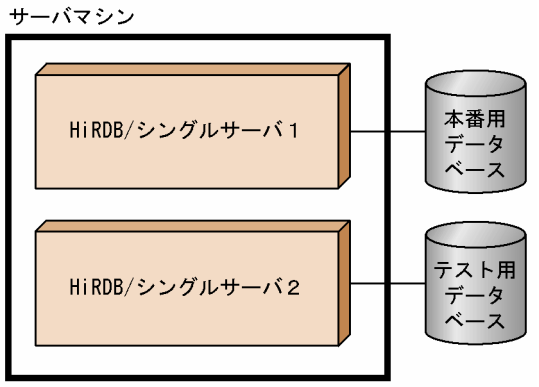
一つのサーバマシンに複数の HiRDB サーバをインストールして，別々のシステムとして運用できます。このシステム形態を**マルチ HiRDB** といいます。例えば，次に示す運用の場合にマルチ HiRDB の導入を検討してください。

- 本番用システムとテスト用システムを同じサーバマシンで運用
- 業務内容が異なるシステムを同じサーバマシンで運用

マルチ HiRDB では，HiRDB/シングルサーバと HiRDB/パラレルサーバの組み合わせを自由にできます。

HiRDB/シングルサーバでのマルチ HiRDB の構成を次の図に示します。

図 1–7　HiRDB/シングルサーバでのマルチ HiRDB の構成



〔説明〕

HiRDB/シングルサーバのマルチ HiRDB です。HiRDB/シングルサーバ 1 を本番用システムとし，HiRDB/シングルサーバ 2 をテスト用システムとしています。

1.3　データベースへのアクセス形態

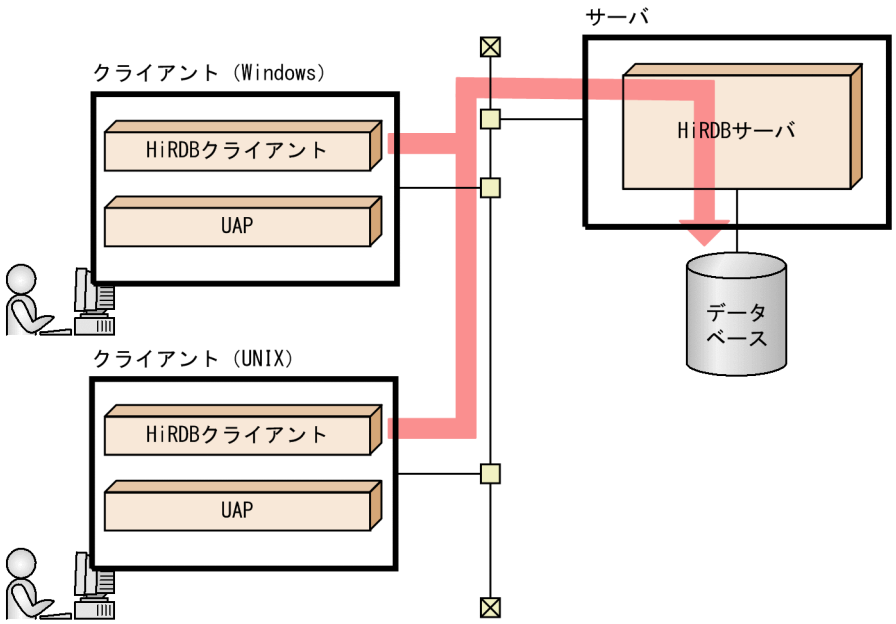
HiRDB のデータベースにアクセスするには HiRDB が提供する **SQL** を使用します。ここでは，SQL を UAP から実行する形態について説明します。

1.3.1　HiRDB クライアント下の UAP を実行してデータベースにアクセスする形態

基本的な形態です。HiRDB クライアント下の UAP を実行してデータベースにアクセスします。

ただし，HiRDB クライアントのバージョンによっては，接続できる HiRDB サーバのプラットフォームが限定されます。詳細については，「HiRDB クライアントと HiRDB サーバの接続可否」を参照してください。HiRDB クライアント下の UAP を実行してデータベースにアクセスする形態を次の図に示します。

図 1–8　HiRDB クライアント下の UAP を実行してデータベースにアクセスする形態

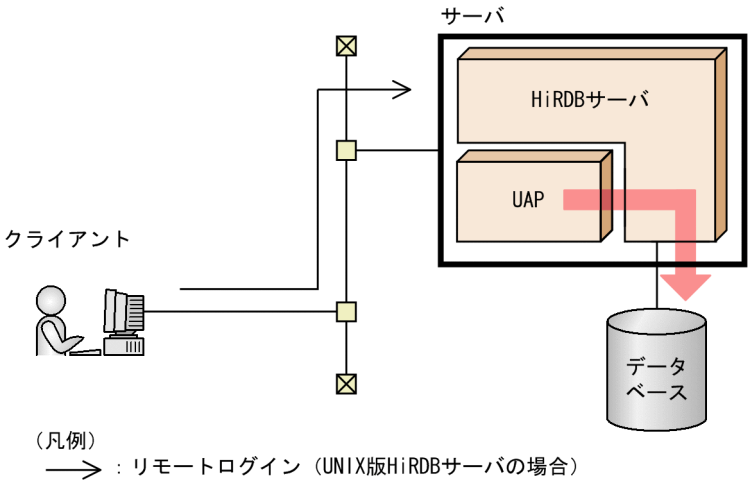


1.3.2　HiRDB サーバ上の UAP を実行してデータベースにアクセスする形態

HiRDB サーバ上の UAP を実行してデータベースにアクセスします。また，UNIX 版 HiRDB サーバの場合でクライアント側システムに HiRDB クライアントがないときは，HiRDB サーバにリモートログインします。

HiRDB サーバ上の UAP を実行してデータベースにアクセスする形態を次の図に示します。

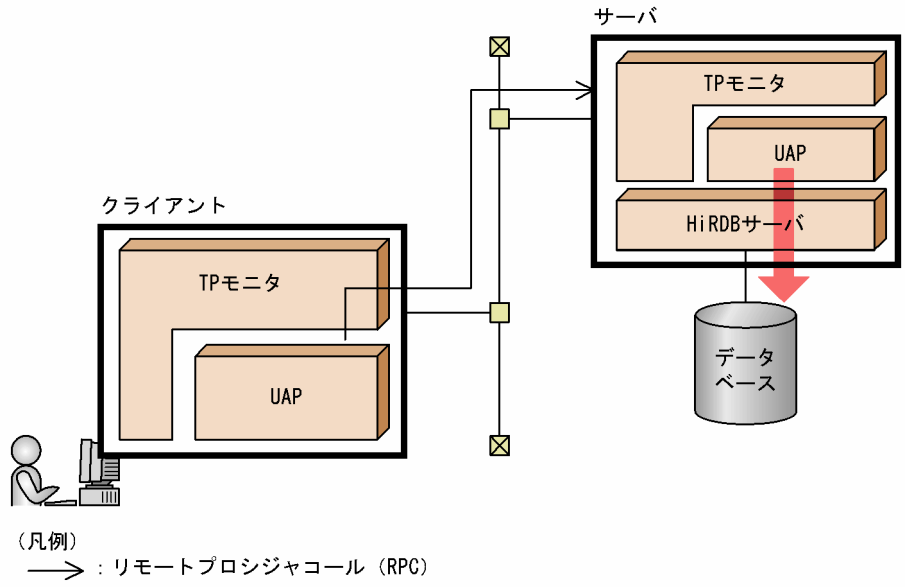
図 1–9　HiRDB サーバ上の UAP を実行してデータベースにアクセスする形態



1.3.3　OLTP の UAP を実行してデータベースにアクセスする形態

OLTP（TP モニタ）のマシンにある UAP からサービスを要求して，HiRDB のデータベースにアクセスする形態です。クライアント側にはサービスを要求する UAP，サーバ側にはサービスを提供する UAP を用意します。OLTP の UAP を実行してデータベースにアクセスする形態を次の図に示します。

図 1–10　OLTP の UAP を実行してデータベースにアクセスする形態

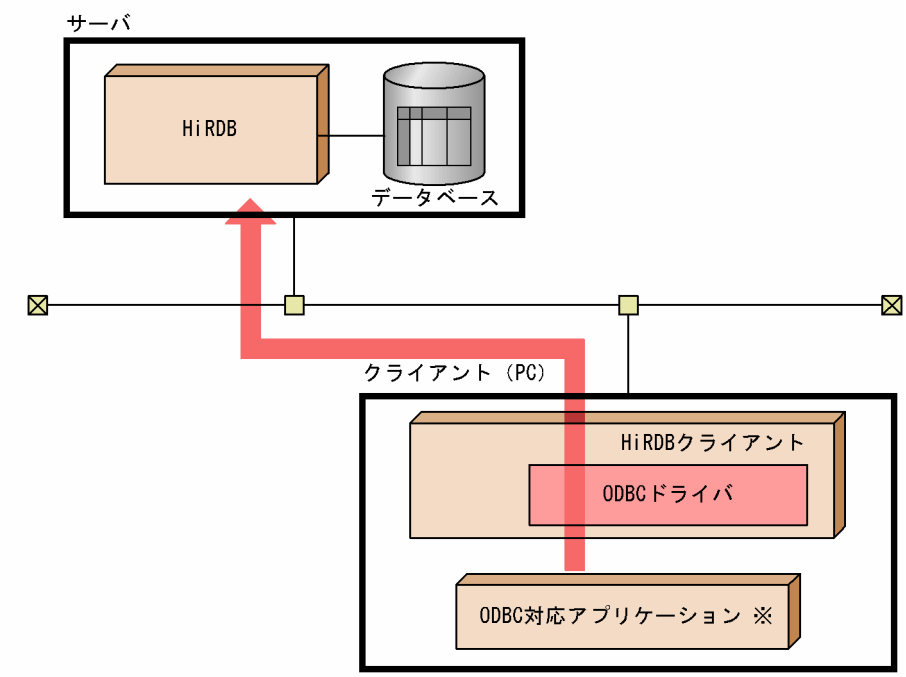


1.3.4　ODBC 対応のアプリケーションからデータベースにアクセスする形態

HiRDB では **ODBC ドライバ**を提供しています。HiRDB クライアントに ODBC ドライバをインストールすると，ODBC 対応のアプリケーションから HiRDB のデータベースにアクセスできるようになります。ODBC 対応のアプリケーションには，Microsoft Access や PowerBuilder など市販のソフトウェアがあります。

また，HiRDB から提供される ODBC 関数を使用した UAP からも HiRDB のデータベースにアクセスできます。ODBC 対応のアプリケーションからデータベースにアクセスする形態を次の図に示します。

図 1–11　ODBC 対応のアプリケーションからデータベースにアクセスする形態



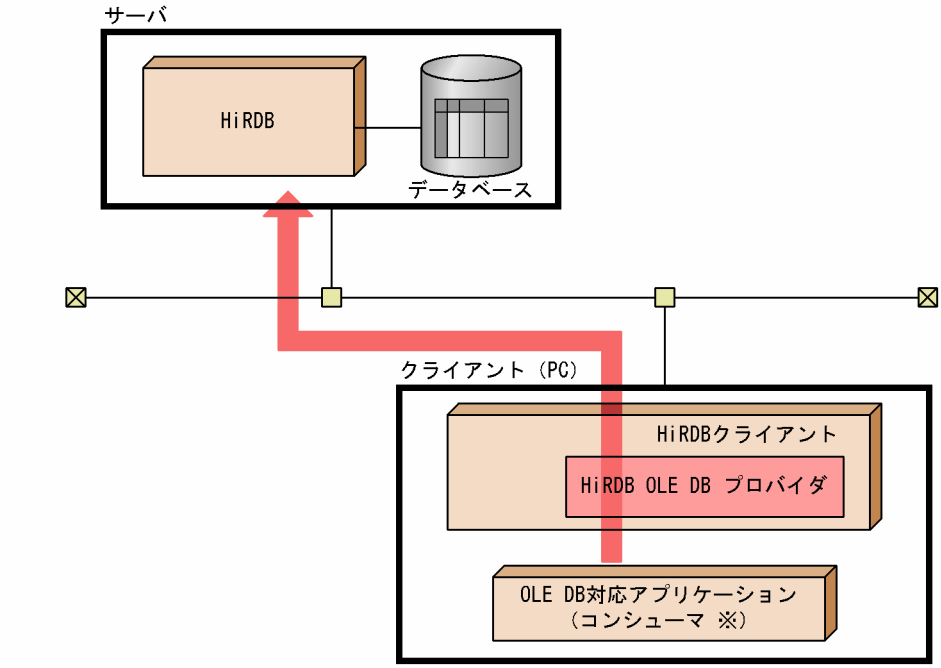
注※　ODBC対応のUAPも含まれます。

ODBC 対応のアプリケーションからの HiRDB へのアクセスについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

1.3.5　OLE DB 対応のアプリケーションからデータベースにアクセスする形態

HiRDB では **OLE DB プロバイダ**を提供しています。HiRDB クライアントのインストール時に OLE DB プロバイダを選択してインストールすると，OLE DB 対応のアプリケーションから，HiRDB のデータベースにアクセスできるようになります。OLE DB 対応のアプリケーションからデータベースにアクセスする形態を次の図に示します。

図 1–12　OLE DB 対応のアプリケーションからデータベースにアクセスする形態



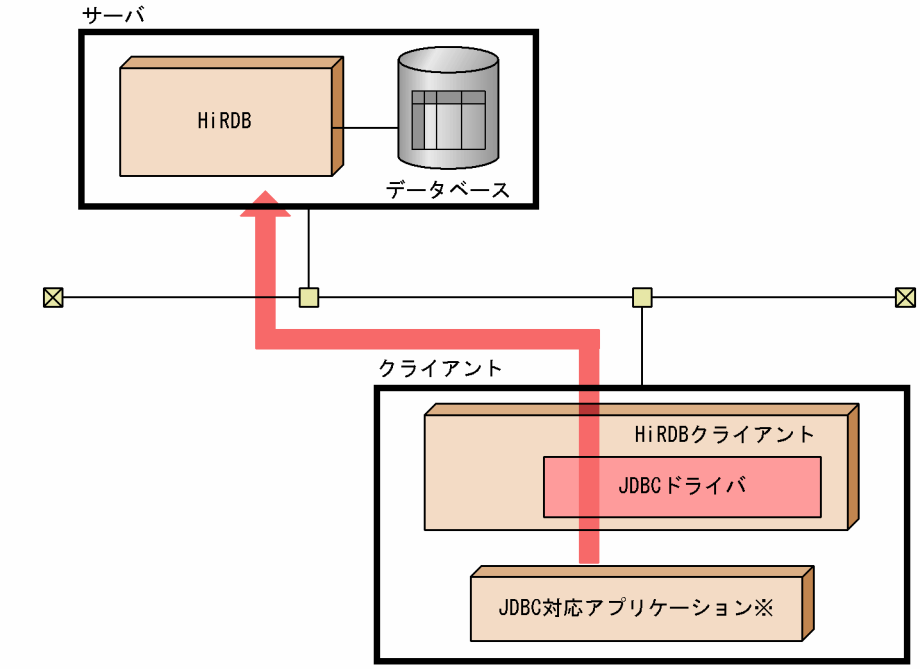
注※　コンシューマとはOLE DBのメソッドとインタフェースを呼び出すプログラムのことです。

OLE DB 対応のアプリケーションからの HiRDB へのアクセスについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

1.3.6　JDBC 対応のアプリケーションからデータベースにアクセスする形態

HiRDB では **JDBC ドライバ**を提供しています。HiRDB クライアントのインストール時に JDBC ドライバを選択してインストールすると，JDBC 対応アプリケーションから HiRDB のデータベースをアクセスできるようになります。JDBC 対応のアプリケーションからデータベースにアクセスする形態を次の図に示します。

図 1–13　JDBC 対応のアプリケーションからデータベースにアクセスする形態



注※　Javaストアドプロシジャ，Javaストアドファンクションを呼び出すプログラムも含まれます。

JDBC 対応のアプリケーションからの HiRDB へのアクセス（Java ストアドプロシジャ，Java ストアドファンクション）については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

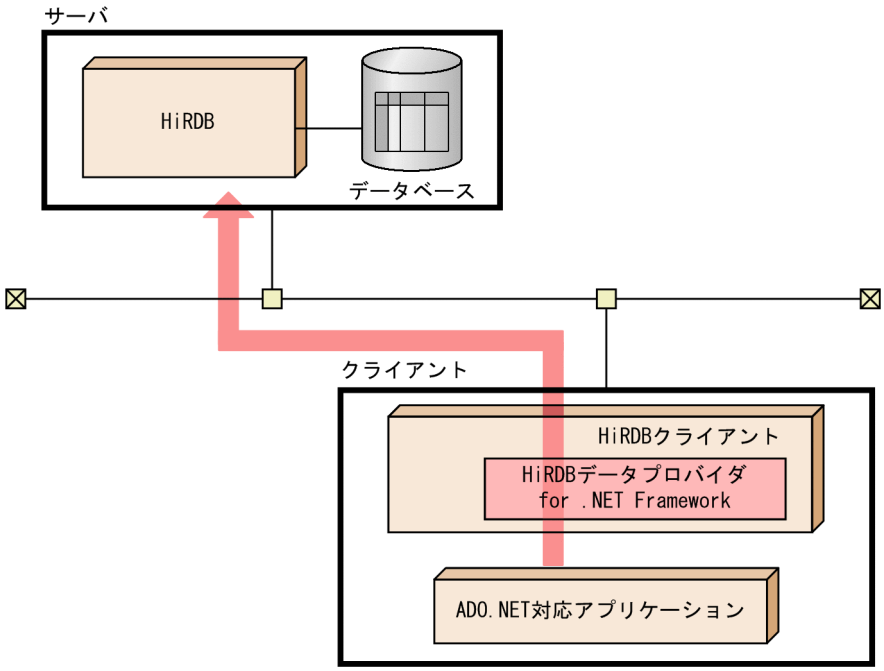
1.3.7　ADO.NET 対応のアプリケーションからデータベースにアクセスする形態

HiRDB では，ADO.NET を使用して HiRDB をアクセスするために必要な **HiRDB データプロバイダ for .NET Framework** を提供しています。HiRDB データプロバイダ for .NET Framework は，ADO.NET 仕様に準拠したデータプロバイダです。HiRDB クライアントのインストール時に HiRDB データプロバイダ for .NET Framework を選択してインストールすると，ADO.NET 対応アプリケーションから HiRDB のデータベースをアクセスできるようになります。

HiRDB データプロバイダ for .NET Framework は，.NET Framework の System.Data 空間で提供されている共通基本インタフェース群を実装しています。また，HiRDB データプロバイダ for .NET Framework 独自の拡張機能として，配列インサート及び繰返し列へのアクセスを実装しています。

ADO.NET 対応のアプリケーションからデータベースにアクセスする形態を次の図に示します。

図 1–14　ADO.NET 対応のアプリケーションからデータベースにアクセスする形態



ADO.NET 対応のアプリケーションからの HiRDB へのアクセスについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

2 HiRDB の付加 PP 及び関連製品との連携

この章では，HiRDB の付加 PP 及び関連製品と連携して実現できる機能について説明します。

2.1　HiRDB の付加 PP

ここでは，次に示す HiRDB の付加 PP を使用して実現できる機能又は操作について説明します。

- HiRDB Advanced High Availability
- HiRDB Staticizer Option【UNIX 版限定】
- HiRDB Non Recover FES
- HiRDB Disaster Recovery Light Edition【UNIX 版限定】
- HiRDB Accelerator

2.1.1　HiRDB Advanced High Availability

HiRDB Advanced High Availability を導入すると，次に示す機能及びコマンドが使用できるようになります。

- スタンバイレス型系切り替え機能
- システム構成変更コマンド（pdchgconf コマンド）
- グローバルバッファの動的変更（pdbufmod コマンド）
- 表のマトリクス分割
- 分割格納条件の変更

(1)　スタンバイレス型系切り替え機能

スタンバイレス型系切り替え機能とは，障害が発生したときにほかのユニットに処理を引き継いでサービスを続行する方式の系切り替えです。待機用のリソースをスタンバイする必要がないため，従来の系切り替え機能（スタンバイ型系切り替え機能）に比べて経済的です。従来の系切り替え機能では，障害時に業務を引き継ぐためのサーバ，CPU，メモリリソース一式を事前に確保する必要がありますが，スタンバイレス型系切り替え機能ではその必要がありません。別のサーバを代替サーバとして登録し，障害発生時にはほかのユニットで処理を引き継ぎます。処理を引き継いだユニットの処理性能が低下することがありますが，リソースを有効に使えるため，システム全体のコストを低減できます。

スタンバイレス型系切り替え機能には 1：1 スタンバイレス型系切り替え機能と影響分散スタンバイレス型系切り替え機能があります。スタンバイレス型系切り替え機能については，「系切り替え機能」を参照してください。

(2)　システム構成変更コマンド（pdchgconf コマンド）

HiRDB システム定義を変更する場合，HiRDB を終了する必要がありましたが，**システム構成変更コマンド**（pdchgconf コマンド）を使用すると，HiRDB を終了する必要がなくなります。このため，HiRDB を稼働したままユニット又はサーバ構成を変更したり，システムファイルを追加したりできます。システム構成変更コマンドについては，「システム構成変更コマンド（pdchgconf コマンド）」を参照してください。

(3)　グローバルバッファの動的変更（pdbufmod コマンド）

グローバルバッファを追加，変更，又は削除する場合，HiRDB システム定義の pdbuffer オペランドを変更する必要があるため，HiRDB を一度終了する必要がありました。pdbufmod コマンドを使用すると，HiRDB を稼働したままグローバルバッファを追加，変更，又は削除できます。これを**グローバルバッファの動的変更**といいます。グローバルバッファの動的変更については，「グローバルバッファの動的変更」を参照してください。

(4)　表のマトリクス分割

表のマトリクス分割とは，複数の列をキーとしてキーレンジ分割する機能のことです。複数の列をキーとして分割することで SQL の並列実行性を高めたり，複数のキーによる検索での検索範囲をより絞り込んで高速に処理したりできます。また，データベース格納領域（RD エリア）をより小さく分割することで，データベースの再編成，バックアップの取得，データベースの回復などの運用時間を短縮できます。表のマトリクス分割については，「表のマトリクス分割」を参照してください。

(5)　分割格納条件の変更

キーレンジ分割※で横分割した表の分割格納条件を，ALTER TABLE で変更できます。表の分割格納条件を変更することで，古いデータを格納していた RD エリアを再利用でき，作業時間を短くできます。分割格納条件の変更については，「表の分割格納条件の変更」を参照してください。

注※

次に示す分割方法の場合に，表の分割格納条件を ALTER TABLE で変更できます。

- 境界値指定
- 格納条件指定（格納条件の比較演算子に＝だけを使用している場合）

2.1.2　HiRDB Staticizer Option【UNIX 版限定】

HiRDB Staticizer Option を導入すると，**インナレプリカ機能**が使用できるようになり，ノンストップサービスに対応したデータベースシステムを構築できます。

(1)　インナレプリカ機能とは

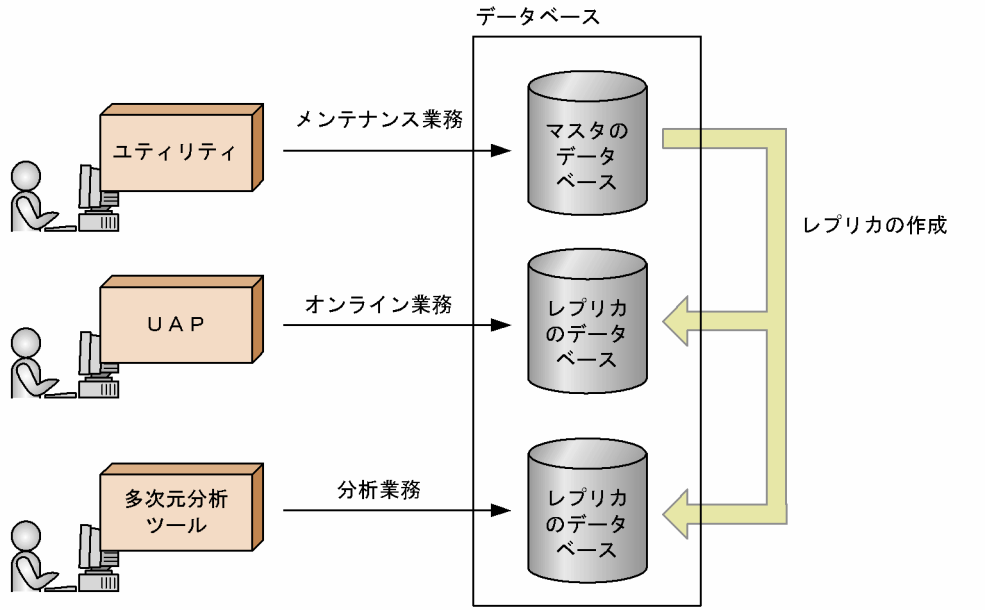
24 時間 365 日ノンストップでオンライン業務を実行していると，データベースのメンテナンス処理（データベースの再編成，データロード，バックアップの取得など）を実施する時間がなくなってしまいます。また，オンライン業務中に，集計・分析処理用のデータマートを作成すると，追加・更新データの抽出がオンライン業務のレスポンスを低下させることがあります。このような状況に対応するためにインナレプリカ機能を提供しています。

ミラーリング機能を持つディスクシステムやソフトウェアを使用して，マスタのデータベースをコピーしたレプリカのデータベースを作成します。オンライン業務はレプリカのデータベースを使用し，データベー

スのメンテナンス処理，又はデータマートの作成処理はマスタのデータベースを使用します。このように，インナレプリカ機能を使用すると，オンライン業務とそのほかの処理（データベースのメンテナンス処理，データマート作成処理）を並行して実施できます。

インナレプリカ機能の概要を次の図に示します。インナレプリカ機能の詳細，及び前提となる製品については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。

図 2–1　インナレプリカ機能の概要



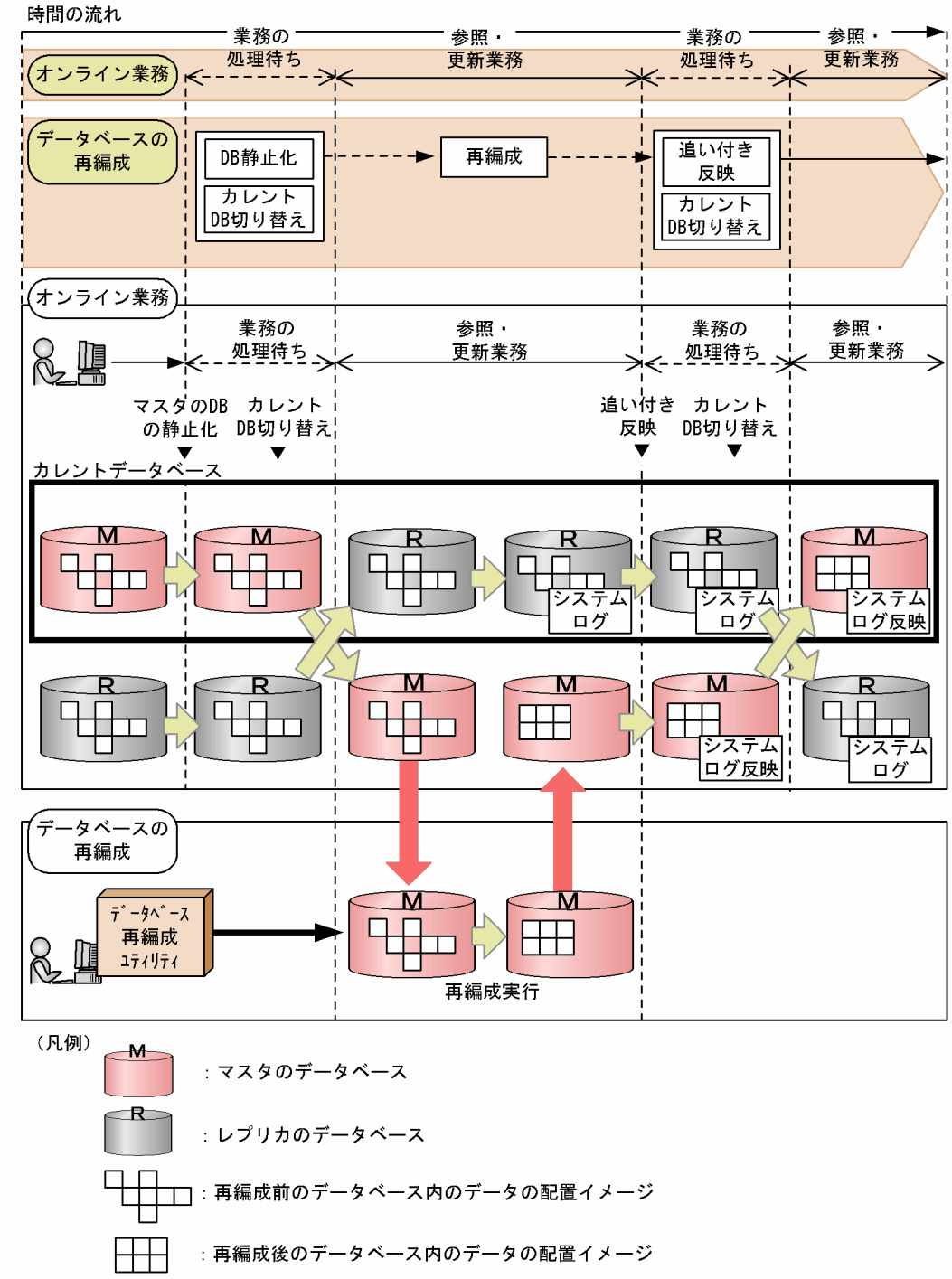
(2)　インナレプリカ機能の適用例

インナレプリカ機能の適用例を次に示します。

(a)　データベースの再編成

インナレプリカ機能を使用すると，データベースの再編成中にデータベースを参照及び更新できます。これを**更新可能なオンライン再編成**といいます。インナレプリカ機能を使用しないと，再編成中のデータベースを参照及び更新できません。また，業務処理をレプリカのデータベースで実行するため，オンライン業務に対する影響を最小限に抑えられます。インナレプリカ機能を使用したデータベースの再編成を次の図に示します。

図 2–2　インナレプリカ機能を使用したデータベースの再編成



〔説明〕

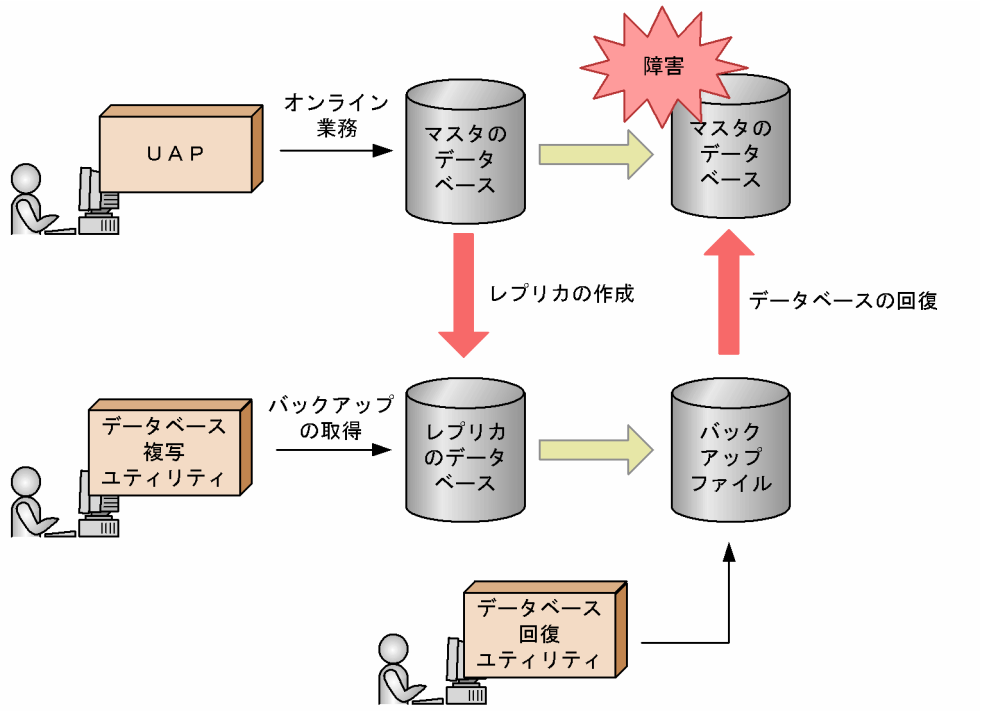
マスタのデータベースに対してデータベース再編成ユティリティ（pdrorg コマンド）でデータベースを再編成し，レプリカのデータベースはオンライン業務に使用します。

再編成を行う前にデータベースの静止化及びカレントデータベースの切り替えを行います。この間はデータベースを参照及び更新できません。カレントデータベースを切り替えた後，マスタのデータベースに対して再編成を実行し，レプリカのデータベースをオンライン業務に使用します。再編成が終了した後に，レプリカのデータベースのシステムログを入力情報にして再編成中に行われた更新処理をマスタのデータベースに反映します。これを**追い付き反映処理**といいます。

(b)　バックアップの取得

インナレプリカ機能を使用すると，バックアップの取得中にデータベースを参照及び更新できます。インナレプリカ機能を使用したバックアップの取得を次の図に示します。

図 2–3　インナレプリカ機能を使用したバックアップの取得



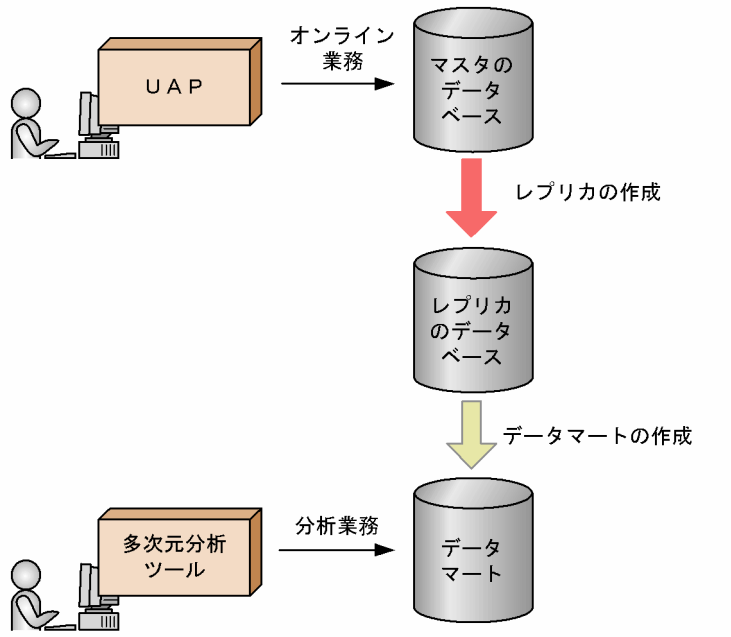
〔説明〕

レプリカのデータベースを使用してバックアップを取得します。バックアップの取得中はマスタのデータベースを参照及び更新できます。マスタのデータベースに障害が発生した場合は，レプリカのデータベースのバックアップを使用してマスタのデータベースを回復できます。

(c)　データマートの作成

インナレプリカ機能を使用すると，レプリカのデータベースから分析業務に使用するデータマートを作成できます。このため，オンライン業務と分析業務を同時に実行できます。インナレプリカ機能を使用したデータマートの作成を次の図に示します。

図 2–4　インナレプリカ機能を使用したデータマートの作成



〔説明〕

オンライン業務中に，レプリカのデータベースからデータの集計，分析処理用のデータマートを作成できます。多彩な視点から分析するにはデータマートのデータ量が増加して，データマートの作成に時間が掛かります。インナレプリカ機能を使用すれば，オンライン業務と並行してデータマートを作成できるため，時間，オンライン業務への負荷などを意識しないで作成できます。

2.1.3　HiRDB Non Recover FES

HiRDB Non Recover FES を導入すると，回復不要 FES が使用できるようになります。回復不要 FES を使用すると，フロントエンドサーバがあるユニットに障害が発生して異常終了したときに未決着状態になっていたトランザクションを HiRDB が自動的に決着するため，異常終了したフロントエンドサーバがあるユニットを再開始しなくても，データベースの更新を再開できます。回復不要 FES については，「回復不要 FES」を参照してください。

2.1.4　HiRDB Disaster Recovery Light Edition【UNIX 版限定】

HiRDB Disaster Recovery Light Edition を導入すると，ログ同期方式のリアルタイム SAN レプリケーションが使用できるようになります。ログ同期方式は，全同期方式，全非同期方式，及びハイブリッド方式に比べ，通信量を削減できます。ログ同期方式のリアルタイム SAN レプリケーションについては，「ログ同期方式」を参照してください。

2.1.5　HiRDB Accelerator

HiRDB Accelerator を導入すると，次に示す機能が使用できるようになります。

- バッチ高速化機能

(1)　バッチ高速化機能

HiRDB Accelerator を導入すると，インメモリデータ処理によってバッチ処理の高速化を実現できます。インメモリデータ処理では，RD エリア内の全データをメモリ上に一括して読み込みます。バッチ処理の実行中はメモリ上のデータだけを更新して，ディスク上のデータは更新しません。バッチ処理が完了した後にメモリ上の更新データを一括してディスクに書き込みます。バッチ処理中はディスク入出力が発生しないため，その分バッチ処理に掛かる時間を短縮できます。インメモリデータ処理については，「インメモリデータ処理」を参照してください。

2.2　データ連携製品との連携

ここでは，データ連携製品を使用して実現できる機能について説明します。

2.2.1　HiRDB Datareplicator，HiRDB Dataextractor との連携

HiRDB Datareplicator，HiRDB Dataextractor と連携すると，**レプリケーション機能**が使用できるようになります。レプリケーション機能とは，分散配置したデータベースの内容をほかのデータベースに反映する機能のことです。レプリケーション機能を使用すると，データベースの情報をほかのシステムのデータベースに反映して，分散システム環境でのデータ管理を支援できます。なお，レプリケーション機能には次に示す二つの機能があります。

- データ連動機能
- データベース抽出・反映サービス機能

(1)　データ連動機能

データ連動機能とは，メインフレームの DBMS 又はほかの HiRDB システムのデータベース更新情報を自動的に自ノードの HiRDB のデータベースに反映する機能です。データ連動機能を使用する場合は，HiRDB の関連製品である **HiRDB Datareplicator** が必要です。データ連動機能の特長を次に示します。

- 一定の時間間隔で，基幹データベースの更新内容を部門データベースに逐次反映します。これによって，基幹業務の最新データを部門データベースに利用できます。
- 基幹データベースから部分的にデータを抽出したり，基幹業務の更新情報の履歴を時系列順に部門データベースに反映したりできます。これによって，データウェアハウスに適したデータを提供できます。

注意事項

HiRDB Datareplicator がサポートしていない HiRDB の機能を使用した場合，データ連動機能が使えなくなることがあります。また，列の属性によっては，HiRDB のデータ連動機能の対象とならない列があります。これらの詳細については，マニュアル「HiRDB データ連動機能 HiRDB Datareplicator」を参照してください。また，最新の情報については，HiRDB のホームページで公開しているオンラインマニュアルを参照してください。

(2)　データベース抽出・反映サービス機能

データベース抽出・反映サービス機能とは，メインフレームの DBMS 又はほかの HiRDB システムに蓄積したデータを自ノードの HiRDB のデータベースに移行する機能です。データベース抽出・反映サービス機能を使用する場合は，HiRDB の関連製品である **HiRDB Dataextractor** が必要です。データベース抽出・反映サービス機能の特長を次に示します。

- 基幹データベースのある時点の情報を部門データベースに一括して反映します。これによって，部門データベースの表を初期作成したり，全データを最新の状態に更新したりできます。
- データの抽出時に条件を指定することで，基幹データベースから部分的にデータを抽出し，各業務に適した部門データベースを作成できます。
- データベース抽出・反映サービス機能を使うことで，データを抽出する UAP の作成，文字コード変換，ファイル転送などの作業が不要になります。

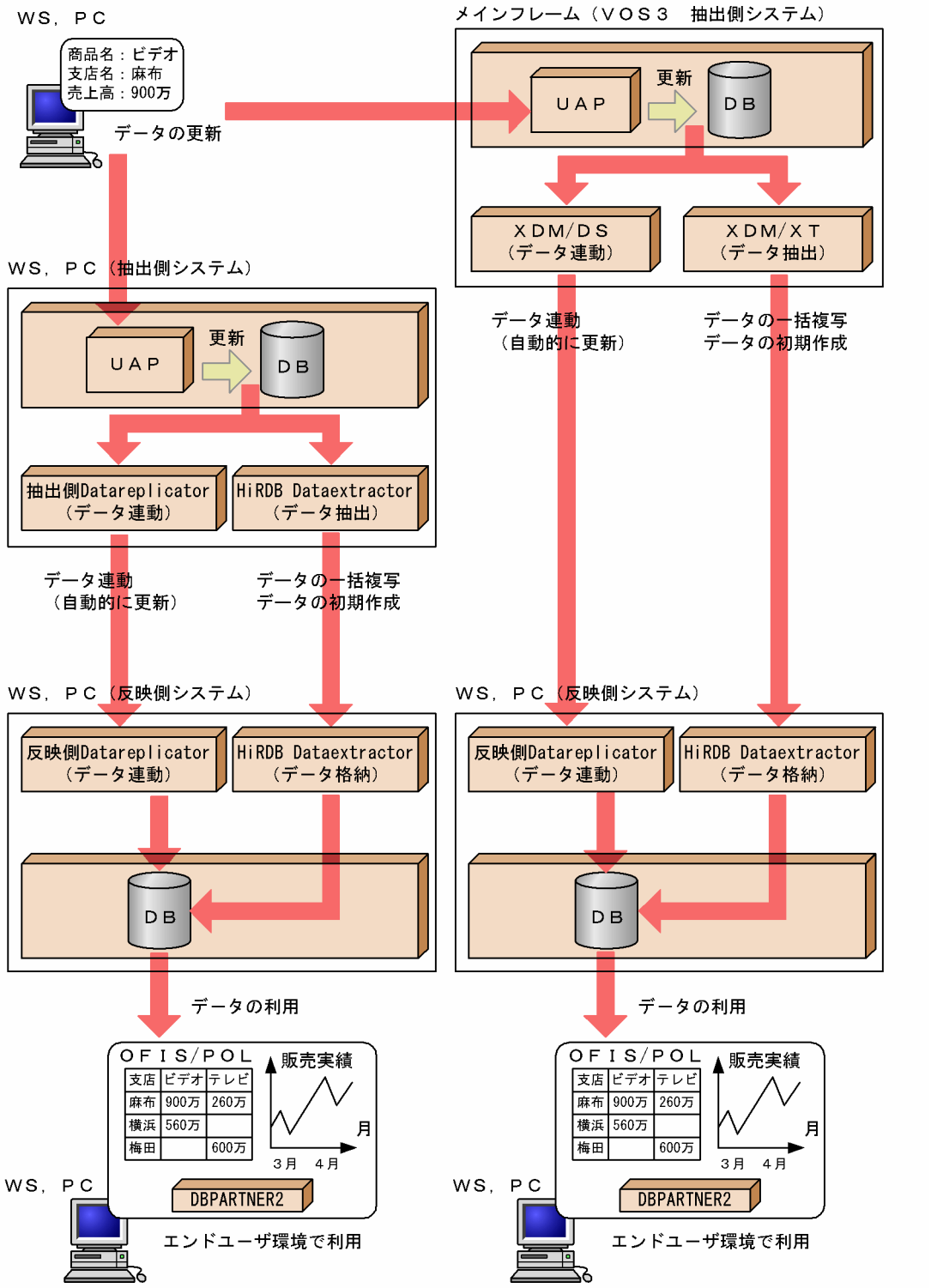
注意事項

列の属性によっては，HiRDB のデータ抽出・反映サービス機能の対象とならない列があります。データベース抽出・反映サービス機能の対象にならない列の属性については，マニュアル「データベース抽出・反映サービス機能 HiRDB Dataextractor」を参照してください。

(3)　レプリケーション機能の適用例

レプリケーション機能の適用例を次の図に示します。

図 2-5　レプリケーション機能の適用例



レプリケーション機能の適用例の詳細については，マニュアル「HiRDB データ連動機能 HiRDB Datareplicator」，及び「データベース抽出・反映サービス機能 HiRDB Dataextractor」を参照してください。

(4)　レプリケーション機能に必要な製品

(a)　HiRDB のシステムで必要な製品

データ連動機能を使用するには，HiRDB Datareplicator が必要です。データベース抽出・反映サービス機能を使用するには，HiRDB Dataextractor が必要です。ただし，相手側システムが RDB1 E2 又は VOS1 の PDM II E2 の場合は，HiRDB Dataextractor は不要です。

(b)　HiRDB とレプリケーション機能で連携できる DBMS の製品

HiRDB とレプリケーション機能で連携できる DBMS の製品を次の表に示します。

表 2–1　HiRDB とレプリケーション機能で連携できる DBMS の製品

連携できる DBMS の製品名（適用 OS）	必要なレプリケーション機能の製品	
	データ連動	データベース抽出・反映サービス機能
XDM/RD E2（VOS3）	XDM/DS	XDM/XT
XDM/SD E2（VOS3）		
ADM（VOS3）	VOS3 Database Datareplicator 又は XDM/DS	
PDM II E2（VOS3）※1	VOS3 Database Datareplicator 又は XDM/DS	
	ファイル転送プログラム※2	
PDM II E2（VOS1）	ファイル転送プログラム※2	PDM II Dataextractor
TMS-4V/SP（VOS3）	VOS3 Database Datareplicator 又は XDM/DS	XDM/XT
RDB1 E2（VOS1）	ファイル転送プログラム※2	RDB1 Dataextractor

注※1

VOS3 の PDM II E2 の場合，データ連動製品を使用してデータ連動をすることも，ファイル転送プログラムを使ってデータ連動をすることもできます。

注※2

ファイル転送プログラムとして XFIT と XFIT の関連製品が必要です。詳細については，マニュアル「HiRDB データ連動機能 HiRDB Datareplicator」及び該当するマニュアルを参照してください。

(5)　データ連動機能を使用する場合の HiRDB の環境設定

データ連動機能を使用する場合（HiRDB のデータベースの更新に連動してデータを抽出する場合だけ）は，HiRDB システム定義の次に示すオペランドを指定する必要があります。

- pd_rpl_hdepath：抽出側 HiRDB Datareplicator 運用ディレクトリ名を指定します。
- pd_rpl_init_start：HiRDB 連携機能の開始時期を指定します。
- pd_log_rpl_no_standby_file_opr：システムログファイルがスワップ先にできない状態になった場合の運用方法を指定します。

2.3　OLTP 製品との連携

HiRDB は OLTP と連携してトランザクション処理を実現できます。ここでは，OLTP 製品との連携の概要について説明します。OLTP との連携方法については，マニュアル「HiRDB システム導入・設計ガイド」を，OLTP と連携するときの UAP の作成方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

2.3.1　連携できる OLTP 製品

HiRDB は次に示す OLTP 製品と連携できます。

- OpenTP1
- TPBroker for C++
- TUXEDO
- OpenTP1/Server Base Enterprise Option（以降，TP1/EE と表記）

ただし，適用 OS によってはこれらの OLTP 製品と連携できないことがあります。適用 OS による OLTP 連携の適用可否を次の表に示します。

表 2–2　適用 OS による OLTP 連携の適用可否

HiRDB の種類	OLTP 製品の種類			
	OpenTP1	TPBroker for C++	TUXEDO	TP1/EE
AIX 版	○	×	×	○
Linux 版	○	×	×	○
Windows 版	○	○	○	×

（凡例）

○：適用できます。

×：適用できません。

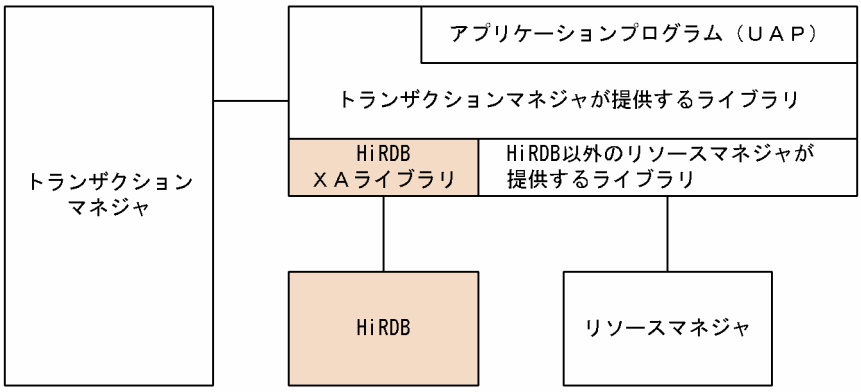
2.3.2　HiRDB XA ライブラリ

HiRDB は OLTP と連携するときに X/Open XA インタフェースを使用しています。**X/Open XA インタフェース**とは，分散トランザクション処理（DTP：Distributed Transaction Processing）システムのトランザクションマネジャ（TM：Transaction Manager）とリソースマネジャ（RM：Resource Manager）の接続インタフェースを規定した X/Open の標準仕様です。XA インタフェースを使用すると，リソースマネジャのトランザクション処理をトランザクションマネジャで制御できます。リソースマネジャのトランザクショ

ン処理をトランザクションマネジャで制御するには，リソースマネジャが提供するライブラリとトランザクションマネジャが提供するライブラリを UAP にリンケージします。

HiRDB の UAP の処理をトランザクションマネジャで制御するために，HiRDB は **HiRDB XA ライブラリ**を提供しています。HiRDB XA ライブラリは，X/Open DTP ソフトウェアアーキテクチャの XA インタフェースの仕様に準拠しています。X/Open DTP モデルでの HiRDB の位置づけを次の図に示します。

図 2–6　X/Open DTP モデルでの HiRDB の位置づけ



なお，X/Open XA インタフェースを使用する UAP から回復不要 FES に接続すると，SQL がエラーリターンします。クライアント環境定義の PDFESHOST 及び PDSERVICEGRP を指定して，回復不要 FES ではないフロントエンドサーバに接続してください。

2.3.3　HiRDB XA ライブラリが提供する機能

HiRDB XA ライブラリが提供する機能を次の表に示します。

表 2–3　HiRDB XA ライブラリが提供する機能

機能	説明
トランザクションの移行	トランザクションのコミット処理を UAP が HiRDB にアクセスしたときと異なるプロセスで実行する機能です。ここでいう UAP とは，HiRDB XA ライブラリを使用して HiRDB に接続する UAP のことです。 トランザクションの移行を使用するかどうかは，クライアント環境定義の PDXAMODE オペランドで指定します。トランザクションの移行については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。
一相最適化	二相コミット制御を一相に最適化する機能です。
読み取り専用	プリペア要求で HiRDB のリソースが更新されていない場合，トランザクションマネジャが二相目にコミット要求をしないで最適化する機能です。
動的トランザクションの登録	UAP を実行する直前に，HiRDB が動的にトランザクションを登録する機能です。

機能	説明
複数接続機能	一つのプロセスから HiRDB サーバに対して複数の CONNECT を別々に実行する機能です。X/Open XA インタフェース環境下での複数接続機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注

HiRDB XA ライブラリでは，非同期 XA 呼び出し（トランザクションマネジャが非同期に HiRDB XA ライブラリを呼び出す機能）を提供していません。

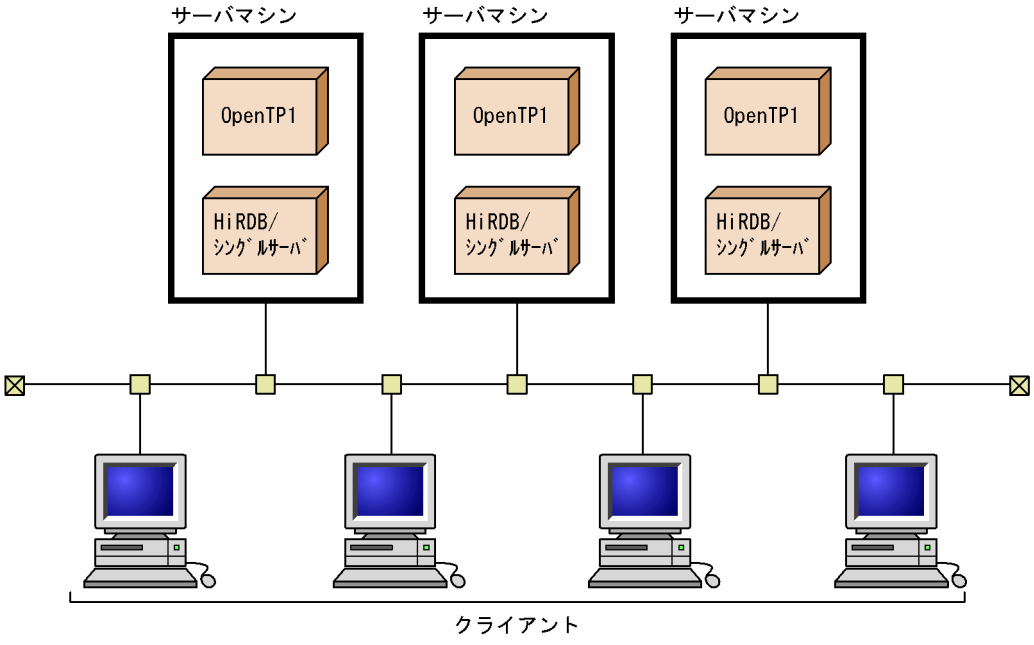
2.3.4　システムの構成

OLTP を使用したシステムの構成について説明します。ここでは，OpenTP1 を例にして説明します。

(1)　HiRDB/シングルサーバとの連携

HiRDB/シングルサーバと OpenTP1 を連携すると，複数の HiRDB/シングルサーバを統合化して処理を実行できます。この場合，データベースはキーレンジ分割などで分割配置し，各サーバマシンで稼働する OpenTP1 が，各 HiRDB/シングルサーバへ処理を振り分けます。複数の HiRDB/シングルサーバを統合化する場合には，OpenTP1 との連携をお勧めします。HiRDB/シングルサーバと OpenTP1 の連携を次の図に示します。

図 2–7　HiRDB/シングルサーバと OpenTP1 の連携



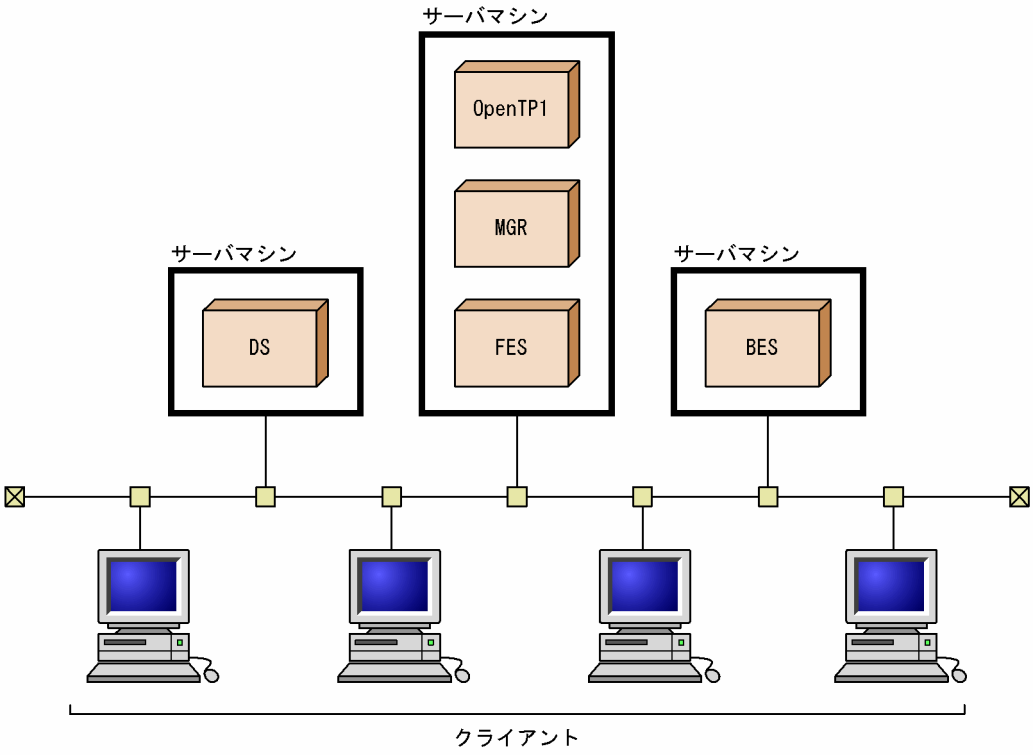
〔説明〕

三つのサーバマシンに，それぞれ HiRDB/シングルサーバと OpenTP1 を用意し，トランザクション処理を各 HiRDB/シングルサーバに振り分ける構成です。

(2)　HiRDB/パラレルサーバとの連携

HiRDB/パラレルサーバと OpenTP1 を連携すると，高性能なトランザクション処理を実現できます。HiRDB/パラレルサーバと OpenTP1 の連携を次の図に示します。

図 2–8　HiRDB/パラレルサーバと OpenTP1 の連携



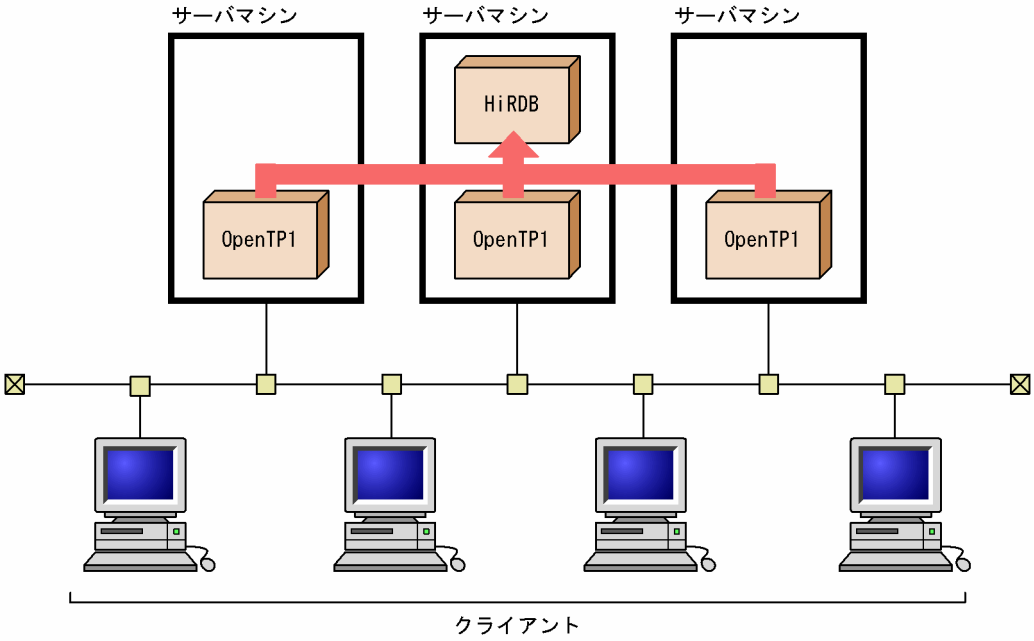
〔説明〕

システムマネジャ，フロントエンドサーバがあるサーバマシンに OpenTP1 を用意し，トランザクション処理を管理する構成です。

(3)　複数の OpenTP1 との連携

複数の OpenTP1 との連携を次の図に示します。

図 2–9　複数の OpenTP1 との連携



〔説明〕

一つの HiRDB と三つの OpenTP1 でクライアント／サーバ型で通信し，トランザクション処理を管理する構成です。

2.3.5　HiRDB のトランザクションマネジャへの登録

HiRDB をリソースマネジャとして，トランザクションマネジャに登録するには，次に示す方法があります。

動的登録

HiRDB をトランザクションマネジャに動的登録すると，トランザクション内で最初の SQL 文を発行したときに，UAP がトランザクションマネジャの制御下に入ります。UAP が HiRDB を含む複数のリソースマネジャをアクセスする場合，UAP が HiRDB をアクセスするとは限らない場合などに，トランザクションマネジャからの HiRDB に対するトランザクション制御のオーバヘッドを削減できます。

静的登録

HiRDB をトランザクションマネジャに静的登録すると，UAP が SQL 文を発行するかどうかに関係なく，トランザクションの開始時に常にトランザクションマネジャの制御下に入ります。

トランザクションマネジャが OpenTP1 の場合，UAP と HiRDB とのコネクションが切断されたとき（ユニットの異常終了，サーバプロセスの異常終了などのとき）に，OpenTP1 にはトランザクション開始時に再接続をする機能があるため，UAP の再起動が不要になります。

HiRDB をトランザクションマネジャに登録する方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

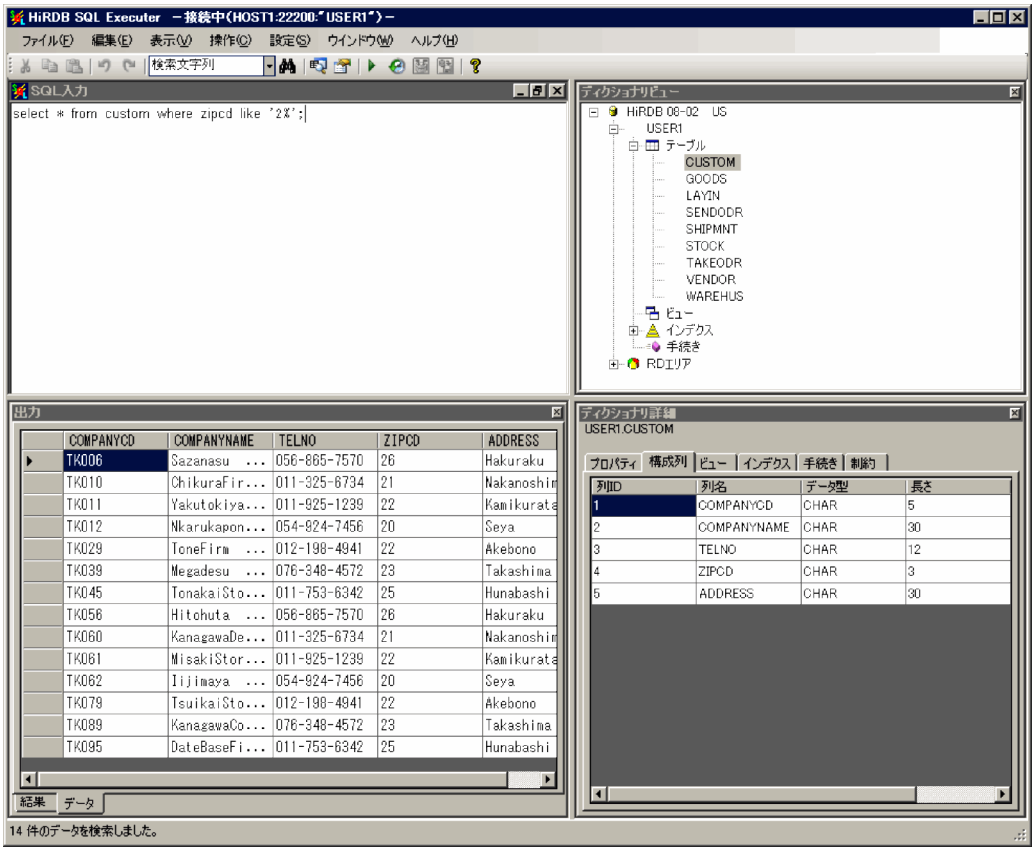
2.4　運用支援製品との連携

ここでは次に示す HiRDB の運用支援製品について説明します。

- HiRDB SQL Executer
- HiRDB Control Manager
- HiRDB SQL Tuning Advisor
- JP1/Performance Management - Agent Option for HiRDB
- JP1/NETM/Audit

2.4.1　HiRDB SQL Executer

HiRDB SQL Executer を使用すると SQL を対話形式で実行できます。データの検証，確認，又は変更や簡単な定型業務処理が実行できるため，データベースのメンテナンスや単純な確認の業務に向いています。なお，HiRDB SQL Executer を HiRDB クライアントで使用する場合は，HiRDB/Developer's Kit 又は HiRDB/Run Time が前提になります。前提となる製品については，HiRDB SQL Executer のリリースノートを参照してください。HiRDB SQL Executer の画面を次に示します。



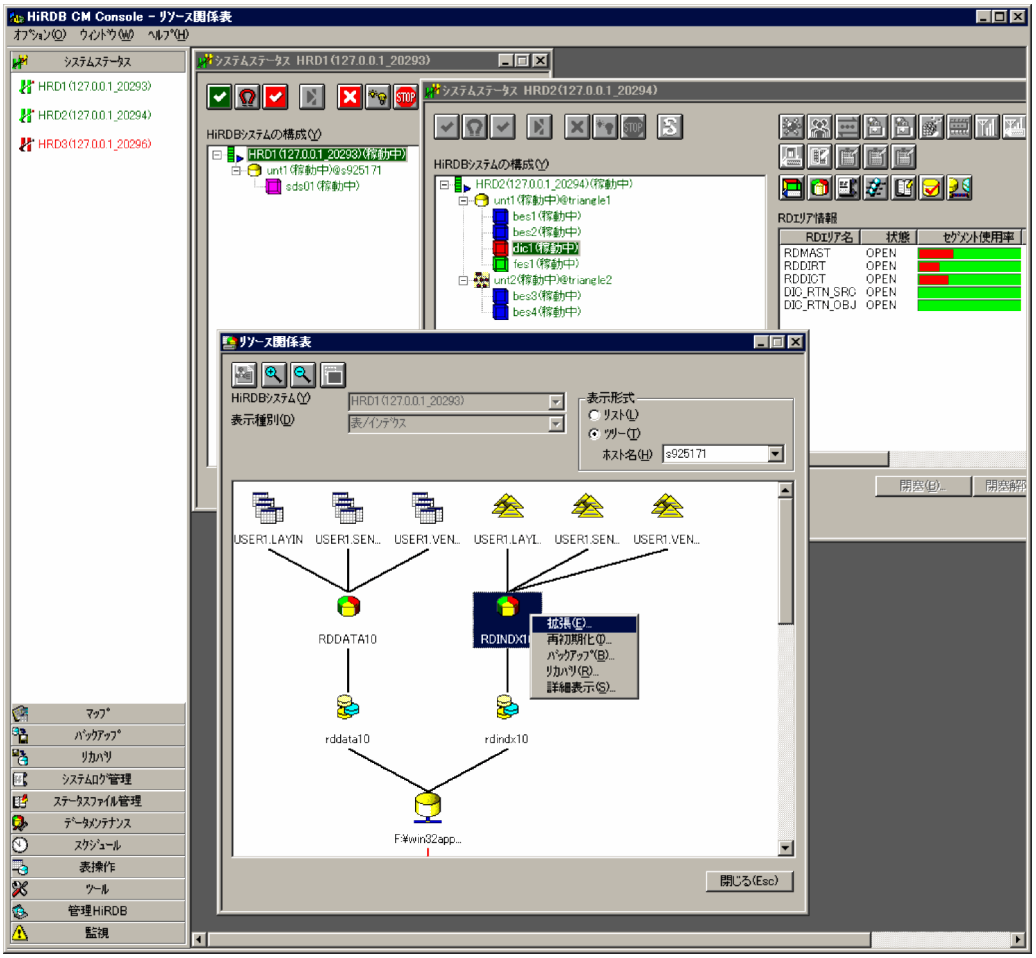
2.4.2 HiRDB Control Manager

HiRDB Control Manager を使用すると，HiRDB サーバの各種状態表示・状態変更，データベースのバックアップ・回復といった基本運用を，PC からのマウスの操作（GUI）で実行できます。また，複数の HiRDB システムを一つの GUI で統合的に運用することもできます。これによって，システム運用の負担を軽減できます。

HiRDB Control Manager では次に示す運用操作を支援しています。

- 複数 HiRDB の同時管理
- HiRDB の各種状態表示
- HiRDB の開始・終了
- データベース複写ユティリティ，データベース回復ユティリティの実行
- RD エリアの操作
- 表の操作
- システムログファイルの操作
- カタログ登録とスケジュール実行
- XDM/RD 稼働状態の表示及び RD エリア・表構成情報の表示
- 表からディスクボリュームまでの関係表示と，表示画面からの操作

HiRDB Control Manager の画面を次に示します。



2.4.3 HiRDB SQL Tuning Advisor

HiRDB SQL Tuning Advisor は，SQL の性能上のボトルネックとなっている所を特定する作業，及びその対策を支援する製品です。HiRDB サーバ及びクライアントから出力される実行記録を解析し，その結果を分かり易く表示したり，ガイダンスを示したりします。

HiRDB SQL Tuning Advisor には次の特長があります。

- 性能上問題になる SQL を効率良く見付け出せます。
- 性能上問題になる SQL のチューニングを効率良く行えます。

HiRDB SQL Tuning Advisor の主な機能を次に示します。

- SQL トレース解析機能

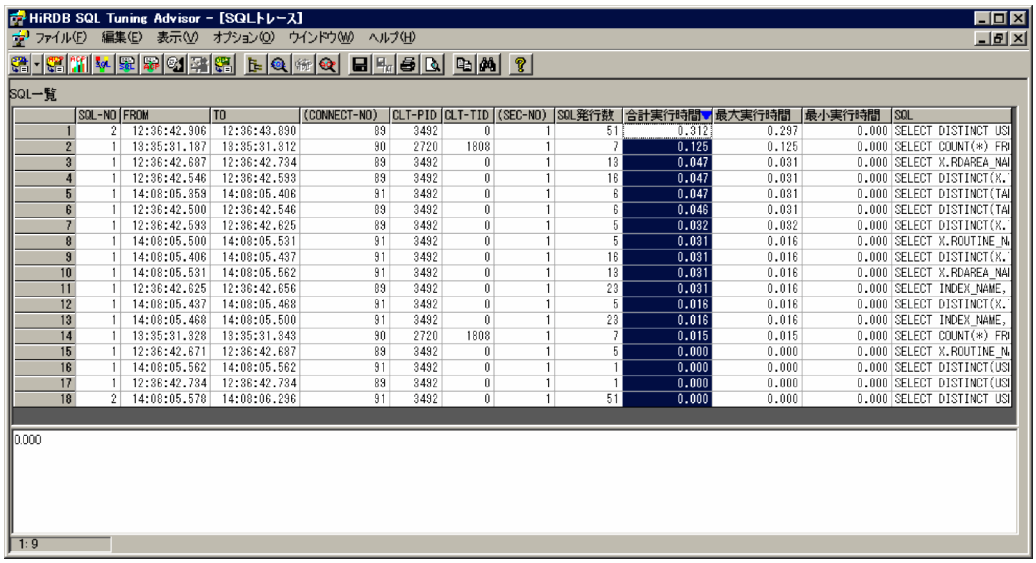
  HiRDB クライアントが生成する SQL トレース情報を解析し，UAP，SQL などの単位で集計した時間を画面上に表示します。SQL 処理時間の集計，長時間実行 SQL の抽出などの作業を支援し，ネックとなっている SQL を特定できます。
- アクセスパス解析機能

HiRDB サーバが生成するアクセスパス情報を解析し，画面上に表示します。表間の結合関係をグラフィカルに表示することもできます。結合方法などについて，性能的に問題のある所（テーブルスキャン，クロスジョインなど）を検出して警告するガイダンスも表示できます。

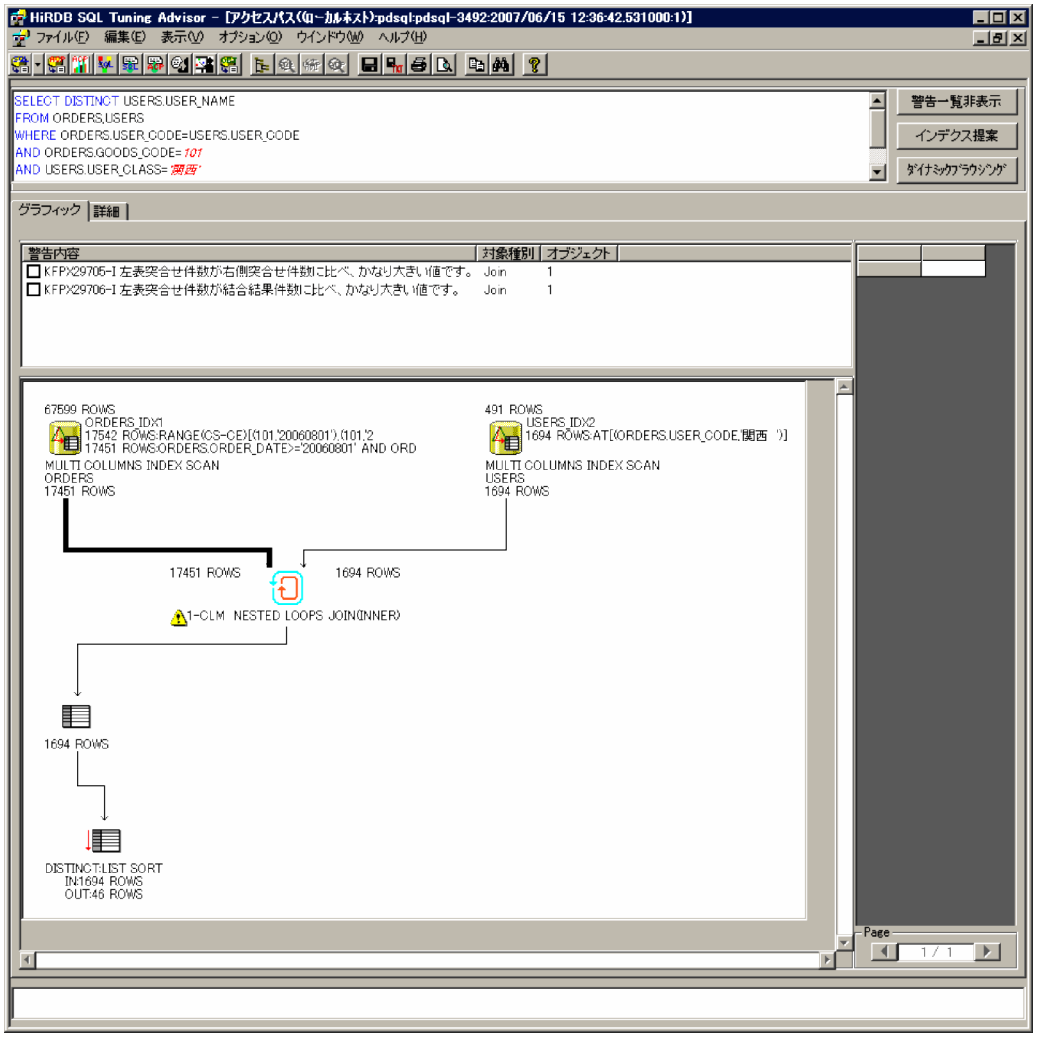
なお，HiRDB クライアントで HiRDB SQL Tuning Advisor を使用する場合は，HiRDB/Developer's Kit 又は HiRDB/Run Time が前提になることがあります。前提製品の詳細については，HiRDB SQL Tuning Advisor のリリースノートを参照してください。

HiRDB SQL Tuning Advisor の画面を次に示します。

SQL トレース解析情報（SQL 一覧）



アクセスパス解析情報（グラフィックイメージ図）



2.4.4 JP1/Performance Management - Agent Option for HiRDB

JP1/Performance Management - Agent Option for HiRDB は，JP1/Performance Management のエージェント製品で HiRDB のパフォーマンスデータを収集します。JP1/Performance Management はプラットフォームが混在する分散システムで，OS，サーバアプリケーション，データベースなどのパフォーマンスを一元管理する製品です。JP1/Performance Management - Agent Option for HiRDB を使用すると，JP1/Performance Management で HiRDB を監視できます。これによって，HiRDB のパフォーマンスを OS やほかのサーバアプリケーションと合わせて一元管理できます。JP1/Performance Management - Agent Option for HiRDB は HiRDB に関する次のパフォーマンスデータを収集します。

- グローバルバッファ
- サーバプロセス
- 作業表用ファイル用の HiRDB ファイルシステム領域
- HiRDB ユニット及びサーバ

- RD エリア
- 排他資源管理テーブルの使用率

JP1/Performance Management - Agent Option for HiRDB については，JP1 のバージョンに応じて，該当するマニュアルを参照してください。

- **JP1 Version 6，及び JP1 Version 7i の場合**
  「JP1/Performance Management - Agent for HiRDB」

2.4.5 JP1/NETM/Audit

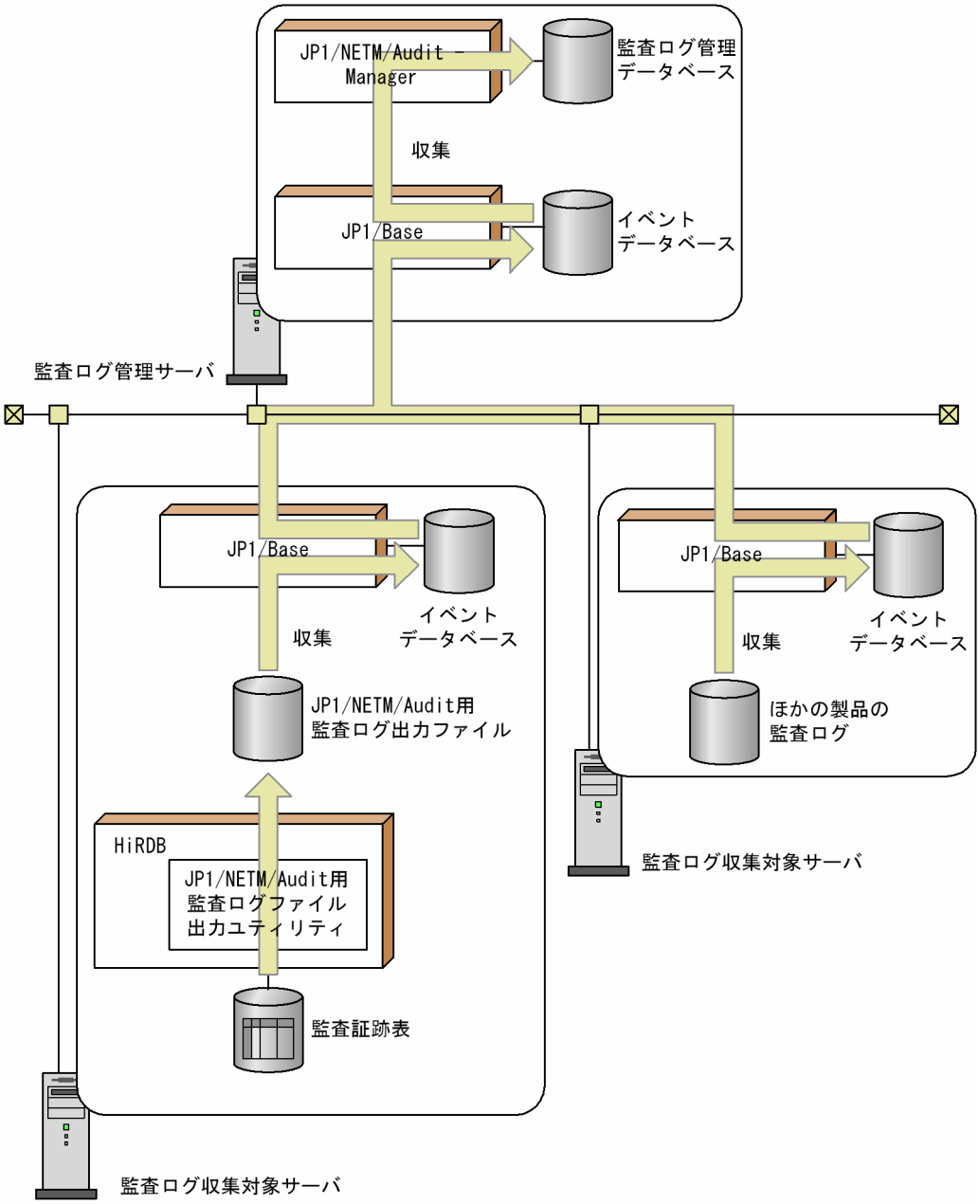
JP1/NETM/Audit を使用すると，システムの監査ログ※を収集・一元管理し，長期間にわたる保管を実現できます。また，GUI で監査ログを検索，集計でき，バックアップ履歴管理などもできます。これによって，企業の内部統制の評価や監査を支援します。HiRDB が出力した監査証跡表のデータを JP1/NETM/Audit 用に加工することで，JP1/NETM/Audit と連携して，HiRDB の監査証跡，及びほかの製品の監査ログを一元管理できます。HiRDB の監査証跡を JP1/NETM/Audit 用に加工するためには，**JP1/NETM/Audit 用監査ログ出力ユティリティ**（**pdaudput** コマンド）を使用します。

注※

監査ログとは，JP1/NETM/Audit がシステム全体から収集する監査証跡の総称です。

JP1/NETM/Audit との連携の概要を次の図に示します。

図 2–10　JP1/NETM/Audit との連携の概要



〔説明〕

監査ログ管理サーバ（JP1/NETM/Audit があるサーバマシン）は，**監査ログ収集対象サーバ**（HiRDB や監査ログを収集するほかの製品があるサーバマシン）のローカルディスクに出力した監査ログを自動収集し，一元管理します。

HiRDB がある監査ログ収集対象サーバでは，JP1/NETM/Audit 用監査ログ出力ユティリティ（pdaudput）が，監査証跡表から必要なデータを JP1/NETM/Audit 用に変換して，**JP1/NETM/Audit 用監査ログ出力ファイル**に出力します。このファイルを JP1/NETM/Audit の前提プログラムである JP1/Base が収集し，イベントデータベースに格納します。監査ログ管理サーバの JP1/Base は監査ログ収集対象サーバの JP1/Base から監査ログを収集し，JP1/NETM/Audit に渡します。

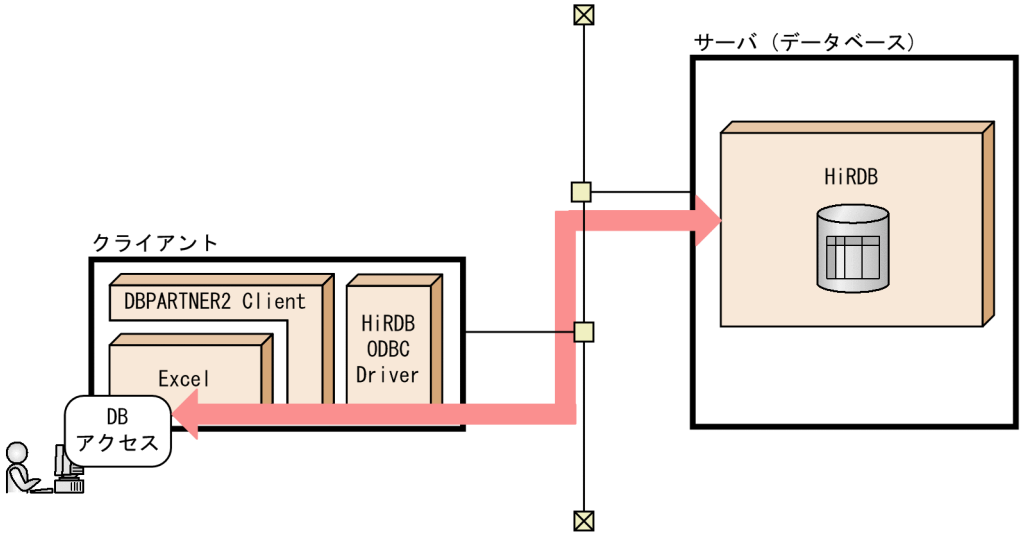
JP1/NETM/Audit との連携についての詳細は，マニュアル「HiRDB システム運用ガイド」を参照してください。また，JP1/NETM/Audit については，次に示すマニュアルを参照してください。

- **JP1 Version 9 の場合**
  「JP1/NETM/Audit 構築・運用ガイド」

2.5　PC 上でのデータベースアクセス製品との連携

DBPARTNER2 は，HiRDB のデータベースに格納されたデータを，SQL を意識しないでアクセスするための GUI 環境を提供するエンドユーザツールです。DBPARTNER2 からデータベースにアクセスする形態を次の図に示します。DBPARTNER2 については，マニュアル「DBPARTNER2 Client 操作ガイド」を参照してください。

図 2–11　DBPARTNER2 からデータベースにアクセスする形態



2.6　マルチメディア情報を扱う製品との連携

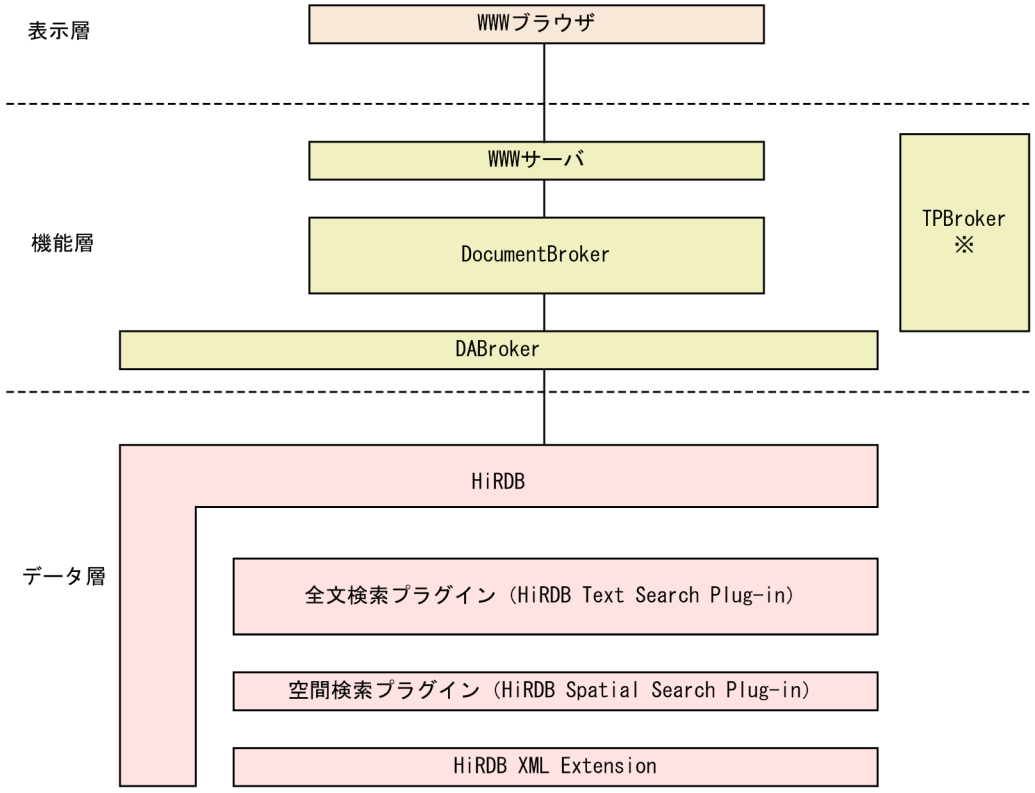
2.6.1　マルチメディア情報を扱う製品

次に示す製品と連携すると，文書や空間データなどのマルチメディア情報や分散オブジェクト環境に対応できるシステムを構築できます。

- 全文検索プラグイン（HiRDB Text Search Plug-in）
- 空間検索プラグイン（HiRDB Spatial Search Plug-in）
- HiRDB XML Extension
- DocumentBroker
- DABroker

HiRDB の関連製品のシステム構成を次の図に示します。

図 2–12　HiRDB の関連製品のシステム構成



注※　TPBrokerは，分散オブジェクト環境でのオブジェクト間の通信，トランザクション制御，運用支援機能などを提供する製品です。

(1)　全文検索プラグイン（HiRDB Text Search Plug-in）

HiRDB Text Search Plug-in を使用すると，構造を持った SGML/XML 文書又は構造を持たない文書を SQL で検索できます。HiRDB Text Search Plug-in は，日本語の検索方式には，文書中の n 文字（n-gram）の出現位置をインデクスとして登録し，検索対象の文書を高速に全文検索できる **n-gram インデクス方式**を採用しています。

(2)　空間検索プラグイン（HiRDB Spatial Search Plug-in）

HiRDB Spatial Search Plug-in を使用すると，地図情報などの空間データ（2 次元データ）を検索できます。

(3)　HiRDB XML Extension

HiRDB XML Extension を使用すると，XML 形式のデータを列データとして扱うことができます。XML 形式のデータを表形式に変換して管理するのではなく，一次情報のデータそのものを管理できるようになります。

XML 型の値に対しては，XML 型の値を処理する SQL（SQL/XML）を使用して，特定の条件に合致する XML 文書の絞り込みや，特定の部分構造の値の取り出しなどの操作を行うことができます。

HiRDB XML Extension では，述語による行の絞り込みに使用する，部分構造をキーとする B-tree インデクス「部分構造インデクス」と，全文検索用 n-gram インデクス「XML 型全文検索用インデクス」を提供しています。

(4)　DocumentBroker

DocumentBroker とは，企業や官公庁などの基幹業務で扱われる分散オブジェクト環境下にある大量の文書を効率良く管理するシステムを構築するためのアプリケーション開発及び実行の基盤となる製品です。DocumentBroker で扱う文書や文書ごとに設定する作成日などの属性を HiRDB のデータベースに格納すると，HiRDB の特長であるスケーラビリティを確保でき，高性能で信頼性のあるシステムを構築できます。さらに，全文検索プラグイン HiRDB Text Search Plug-in を使用することで，文書をより高速に検索できるようになります。DocumentBroker が提供する機能として，文書の履歴管理，複数文書のグループ化による管理などがあります。

(5)　DABroker

DABroker とは，大量データへの高速アクセスを実現するデータベースアクセスツールです。

分散オブジェクト環境でのデータベースアクセス

DABroker を使用すると，HiRDB のデータベースに文書，画像，音声などのマルチメディアデータを格納して，分散オブジェクト環境下でもデータベースの所在を意識することなくアクセスできます。

データベースとのレスポンス性能を向上

マルチスレッド環境で利用できるため，1 接続クライアント当たりのメモリ所要量が削減でき，接続クライアント数が増えた場合でも，HiRDB のサーバからの均一なレスポンスを実現できます。さらに，クライアントからのデータ転送とデータベースからのデータ読み込みを並行して動作できるため，少ないクライアント数の場合でもレスポンス性能を向上できます。

高度なデータベースアクセスを実現

複雑なデータベースアクセス処理を八つの関数に統合した Simple Interface 機能によって，絞り込み検索やランダム検索が実現できます。

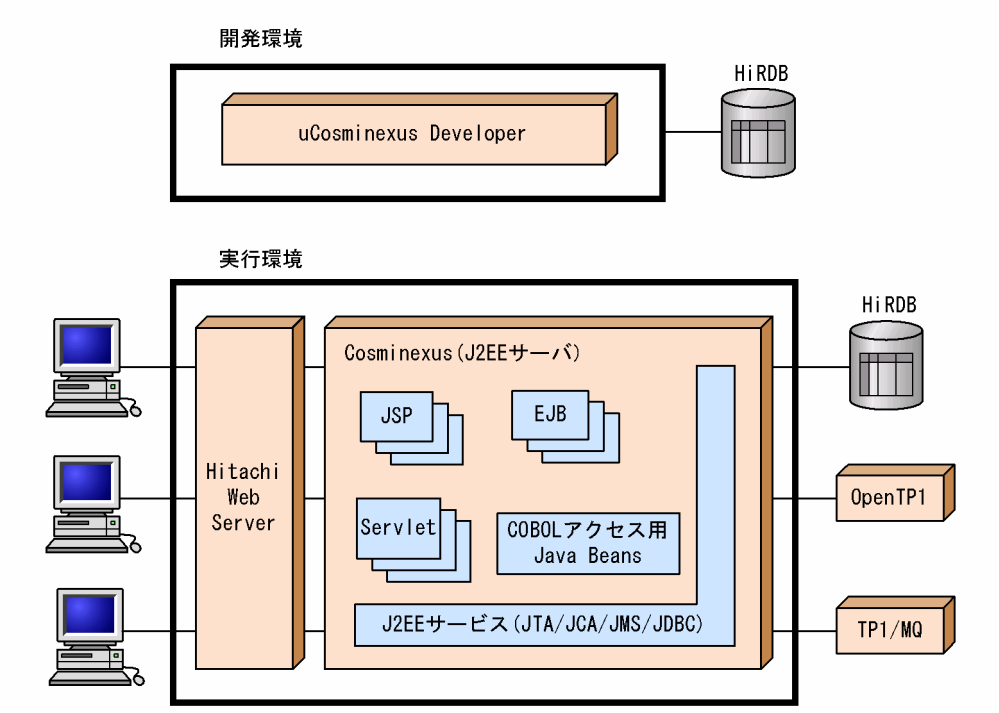
2.7　Cosminexus との連携

Cosminexus と連携すると次のことが実現できます。

- J2EE 標準インタフェースである JTA/JCA/JMS の利用などによって，既存の基幹システムを高信頼 Web システムに拡張できます。
- Java アプリケーションから COBOL アクセス用 JavaBean を介して，既存の COBOL 資産を活用できます。
- uCosminexus Developer の各種ツール群と連携し，アプリケーションの生産性を向上できます。

Cosminexus との連携を次の図に示します。

図 2–13　Cosminexus との連携



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

3 データベースの論理構造

この章では，データベースの論理構造（RD エリア，スキーマ，表，インデクス，及びオブジェクトリレーショナルデータベース）について説明します。

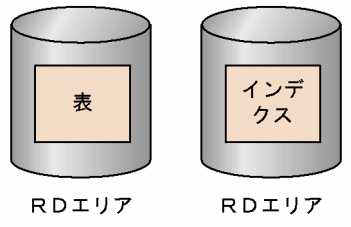
3.1　RD エリア

ここでは，RD エリアの種類と作成方法について説明します。

3.1.1　RD エリアの種類

RD エリアとは，表及びインデクスを格納する論理的なエリアのことです。RD エリアと表及びインデクスの関係を次の図に示します。

図 3–1　RD エリアと表及びインデクスの関係



RD エリアには次の表に示す種類があります。

表 3–1　RD エリアの種類

項番	RD エリアの種類	作成できる個数	必要／任意
1	マスタディレクトリ用 RD エリア	1	○
2	データディクショナリ用 RD エリア	1～41	○
3	データディレクトリ用 RD エリア	1	○
4	データディクショナリ LOB 用 RD エリア	2	△
5	ユーザ用 RD エリア	1～8,388,589	○
6	ユーザ LOB 用 RD エリア	1～8,388,325	△
7	レジストリ用 RD エリア	1	△
8	レジストリ LOB 用 RD エリア	1	△
9	リスト用 RD エリア	1～8,388,588	△

（凡例）

○：必要な RD エリアです。

△：任意で作成する RD エリアです。

注 1　項番 1～3 の RD エリアを総称して**システム用 RD エリア**といいます。

注 2　項番 4，6，8 の RD エリアを総称して **LOB 用 RD エリア**といいます。

(1)　マスタディレクトリ用 RD エリア

マスタディレクトリ用 RD エリアには，システムの内部情報を格納します。マスタディレクトリ用 RD エリアは必ず作成してください。

(2)　データディクショナリ用 RD エリア

データディクショナリ用 RD エリアには，ディクショナリ表及びディクショナリ表のインデクスを格納します。ディクショナリ表は，HiRDB の利用者が検索できます。ディクショナリ表については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

データディクショナリ用 RD エリアは必ず作成してください。

(3)　データディレクトリ用 RD エリア

データディレクトリ用 RD エリアには，システムの内部情報を格納します。データディレクトリ用 RD エリアは必ず作成してください。

(4)　データディクショナリ LOB 用 RD エリア

データディクショナリ LOB 用 RD エリアには，ストアドプロシジャ又はストアドファンクションの定義ソース及びオブジェクトを格納します。また，トリガを定義する場合，内部的に生成されるトリガの SQL オブジェクトを格納します。ストアドプロシジャ若しくはストアドファンクションを使用する場合，又はトリガを定義する場合に，データディクショナリ LOB 用 RD エリアを作成してください。

データディクショナリ LOB 用 RD エリアには，次に示す 2 種類があります。

- 定義ソースを格納する RD エリア
- SQL オブジェクトを格納する RD エリア

> **参考**
>
> トリガとは，参照制約動作で HiRDB が内部的に生成するトリガも含みます。そのため，参照制約を定義する場合もデータディクショナリ LOB 用 RD エリアを作成してください。

(5)　ユーザ用 RD エリア

ユーザ用 RD エリアには，表及びインデクスを格納します。ユーザ用 RD エリアは必ず作成してください。一つの表を一つ又は複数のユーザ用 RD エリアに格納できます。また，一つのユーザ用 RD エリアには，一つ又は複数の表を格納できます。なお，インデクスを格納する場合も同様です。ユーザ用 RD エリアには，次に示す 2 種類があります。

- **公用 RD エリア**：HiRDB に登録されているすべてのユーザが使用できる RD エリアです。
- **私用 RD エリア**：権限があるユーザだけが使用できる RD エリアです。

また，ユーザ用 RD エリアには，すべてのバックエンドサーバから参照できる**共用 RD エリア**，一時表及び一時インデクスを格納する**一時表用 RD エリア**があります。共用 RD エリア及び一時表用 RD エリアについては，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(6) ユーザ LOB 用 RD エリア

ユーザ LOB 用 RD エリアには，文書，画像，音声などの長大な可変長データを格納します。次に示すデータを取り扱う場合に，ユーザ LOB 用 RD エリアを作成してください。

- BLOB 型を指定した列（LOB 列）
- 抽象データ型内の，BLOB 型を指定した属性
- プラグインインデクス

一つの表に複数の「LOB 列」，「抽象データ型内の，BLOB 型を指定した属性」又は「プラグインインデクス」がある場合，一つの列，一つの属性，一つのプラグインインデクスごとに，それぞれを別々のユーザ LOB 用 RD エリアに格納します。ユーザ LOB 用 RD エリアには，次に示す 2 種類があります。

公用 RD エリア

HiRDB に登録されているすべてのユーザが使用できる RD エリアです。

私用 RD エリア

権限のあるユーザだけが使用できる RD エリアです。

(7) レジストリ用 RD エリア

レジストリ用 RD エリアには，レジストリ情報を格納します。プラグインがレジストリ機能を使用する場合にレジストリ用 RD エリアを作成してください。レジストリ機能を使用する場合は，レジストリ用 RD エリアとレジストリ LOB 用 RD エリアの両方を必ず作成する必要があります。全文検索プラグイン（HiRDB Text Search Plug-in）はレジストリ機能を使用しますが，プラグインの種類によってはレジストリ用 RD エリアを使用しないものもあります。

(8) レジストリ LOB 用 RD エリア

レジストリ LOB 用 RD エリアにはレジストリ情報に登録したキーのうち，キー長が 32,000 バイトを超えるものを格納します。プラグインがレジストリ機能を使用する場合にレジストリ LOB 用 RD エリアを作成してください。全文検索プラグイン（HiRDB Text Search Plug-in）はレジストリ機能を使用しますが，プラグインの種類によってはレジストリ LOB 用 RD エリアを使用しないものもあります。

(9) リスト用 RD エリア

リスト用 RD エリアには，ASSIGN LIST 文で作成するリストを格納します。絞込み検索をする場合に，リスト用 RD エリアを作成してください。

3.1.2　RD エリアの作成方法

次に示す方法で RD エリアを作成します。

- **HiRDB の初期導入時**
  次に示す環境設定支援ツールを使用した場合，ツール実行時に RD エリアが作成されます。
  - 簡易セットアップツール【Windows 版限定】
  - バッチファイル（SPsetup.bat）【Windows 版限定】

  コマンドで HiRDB の環境設定を行う場合は，次に示すユティリティで RD エリアを作成します。
  - データベース初期設定ユティリティ（pdinit）
- **RD エリアの追加時**
  次に示すユティリティで RD エリアを作成（追加）します。
  - データベース構成変更ユティリティ（pdmod）
  - レジストリ機能初期設定ユティリティ（pdreginit）

RD エリアの作成方法の詳細については，マニュアル「HiRDB システム導入・設計ガイド」又は「HiRDB システム運用ガイド」を参照してください。

ポイント

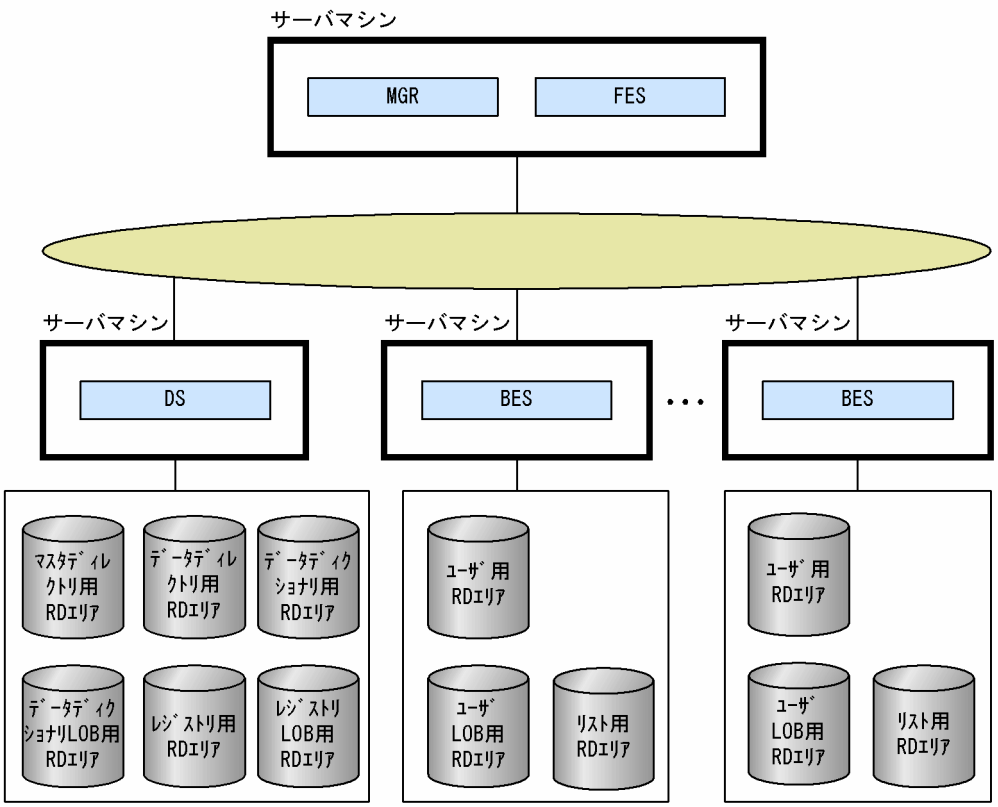
- pd_max_rdarea_no オペランドの値に注意してください。このオペランドには RD エリアの最大数を指定します。RD エリアの数がこのオペランドの値を超えている場合は HiRDB を正常開始できません。
- HiRDB/パラレルサーバの場合は RD エリアを作成するサーバを意識する必要があります。RD エリアを作成するサーバマシン（HiRDB/パラレルサーバの場合）を次の表に示します。また，RD エリアの構成例（HiRDB/パラレルサーバの場合）を次の図に示します。

表 3–2　RD エリアを作成するサーバマシン（HiRDB/パラレルサーバの場合）

RD エリアの種類	RD エリアを作成するサーバマシン
マスタディレクトリ用 RD エリア	ディクショナリサーバがあるサーバマシンに作成します。
データディクショナリ用 RD エリア	
データディレクトリ用 RD エリア	
データディクショナリ LOB 用 RD エリア	
レジストリ用 RD エリア	
レジストリ LOB 用 RD エリア	
ユーザ用 RD エリア	バックエンドサーバがあるサーバマシンに作成します。
ユーザ LOB 用 RD エリア	

RD エリアの種類	RD エリアを作成するサーバマシン
リスト用 RD エリア	

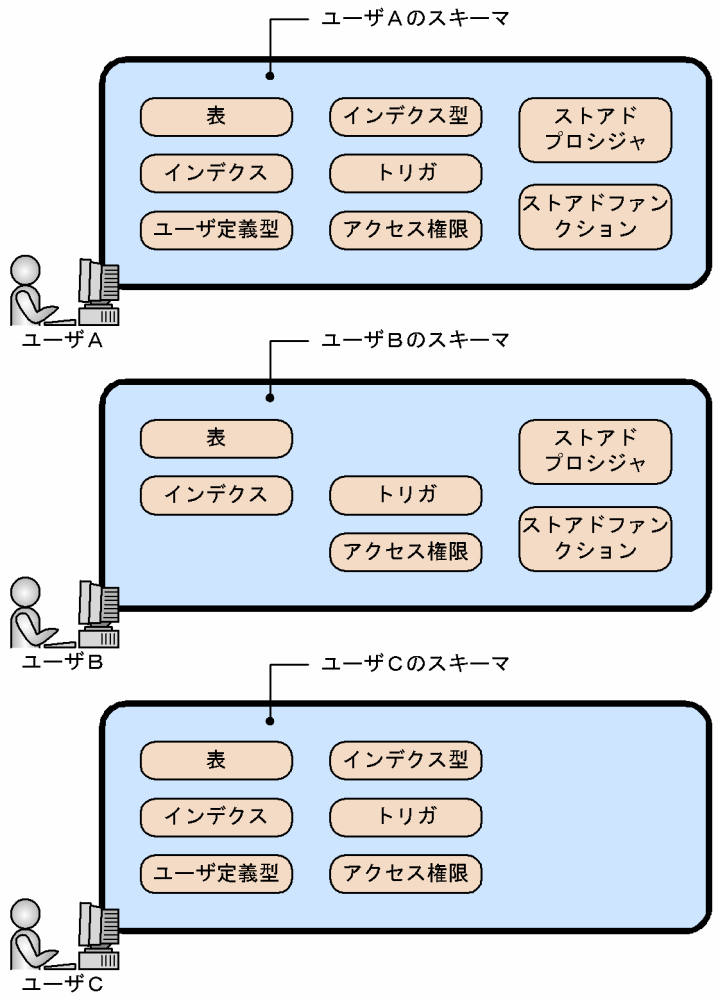
図 3–2　RD エリアの構成例（HiRDB/パラレルサーバの場合）



3.2　スキーマ

表，インデクス，抽象データ型（ユーザ定義型），インデクス型，ストアドプロシジャ，ストアドファンクション，トリガ，及びアクセス権限を包括した概念を**スキーマ**といいます。ユーザはこれらのリソースを定義する前にスキーマを定義しておく必要があります。スキーマは定義系 SQL の **CREATE SCHEMA** で定義します。1 ユーザには 1 スキーマを定義できます。スキーマの概念を次の図に示します。

図 3–3　スキーマの概念



3.3　表

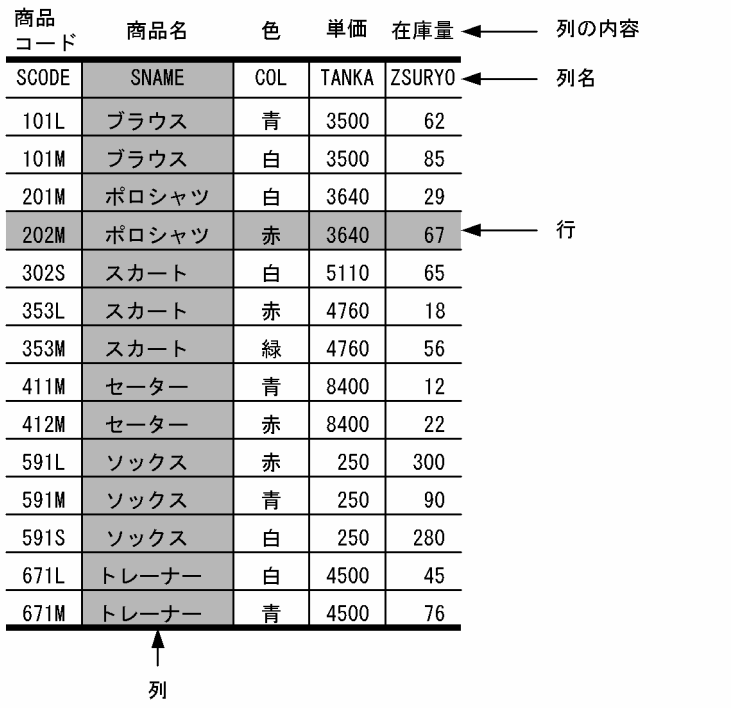
HiRDB のデータベースはリレーショナルデータベースであり，**行**と**列**から構成される**表**の構造になっています。

3.3.1　表の基本構造

(1)　行と列

表の横方向を**行**，縦方向を**列**といいます。一つの行は表を操作するときの単位であり，一つ以上の列から構成されます。列には**列名**を付けて，表を操作するときは列名で行の中の位置を識別します。表の構造を次の図に示します。

図 3–4　表の構造



〔説明〕

在庫表（表名：ZAIKO）の構造を例にしています。表をどのような列で構成するかは，定義系 SQL の **CREATE TABLE** で定義します。在庫表の定義例を次に示します。

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4),
     SNAME NCHAR(8),
     COL NCHAR(1),
     TANKA INTEGER,
     ZSURYO INTEGER);
```

(2)　データ型

「列」又は「抽象データ型を構成する属性」に対して，データ型を指定します。データ型には大きく分けて，既定義型とユーザ定義型があります。**既定義型**とは，HiRDB が提供するデータ型のことです。**ユーザ定義型**とは，ユーザが任意に定義できるデータ型のことです。データ型の種類を次の表に示します。

表 3–3　データ型の種類

分類		データ型	データ形式
既定義型	数データ	INTEGER	整数
		SMALLINT	整数
		LARGE DECIMAL	固定小数点数
		FLOAT	倍精度浮動小数点数
		SMALLFLT	単精度浮動小数点数
	文字データ	CHARACTER	固定長文字列
		VARCHAR	可変長文字列
	各国文字データ	NCHAR	固定長各国文字列
		NVARCHAR	可変長各国文字列
	混在文字データ	MCHAR	固定長混在文字列
		MVARCHAR	可変長混在文字列
	日付データ	DATE	日付
	時刻データ	TIME	時刻
	時刻印データ	TIMESTAMP	時刻印データの小数秒精度
	日間隔データ	INTERVAL YEAR TO DAY	日間隔
	時間隔データ	INTERVAL HOUR TO SECOND	時間隔
	長大データ	BLOB	バイナリデータ列
	バイナリデータ	BINARY	バイナリデータ列
	論理データ	BOOLEAN	論理値
ユーザ定義型	抽象データ型	－	－

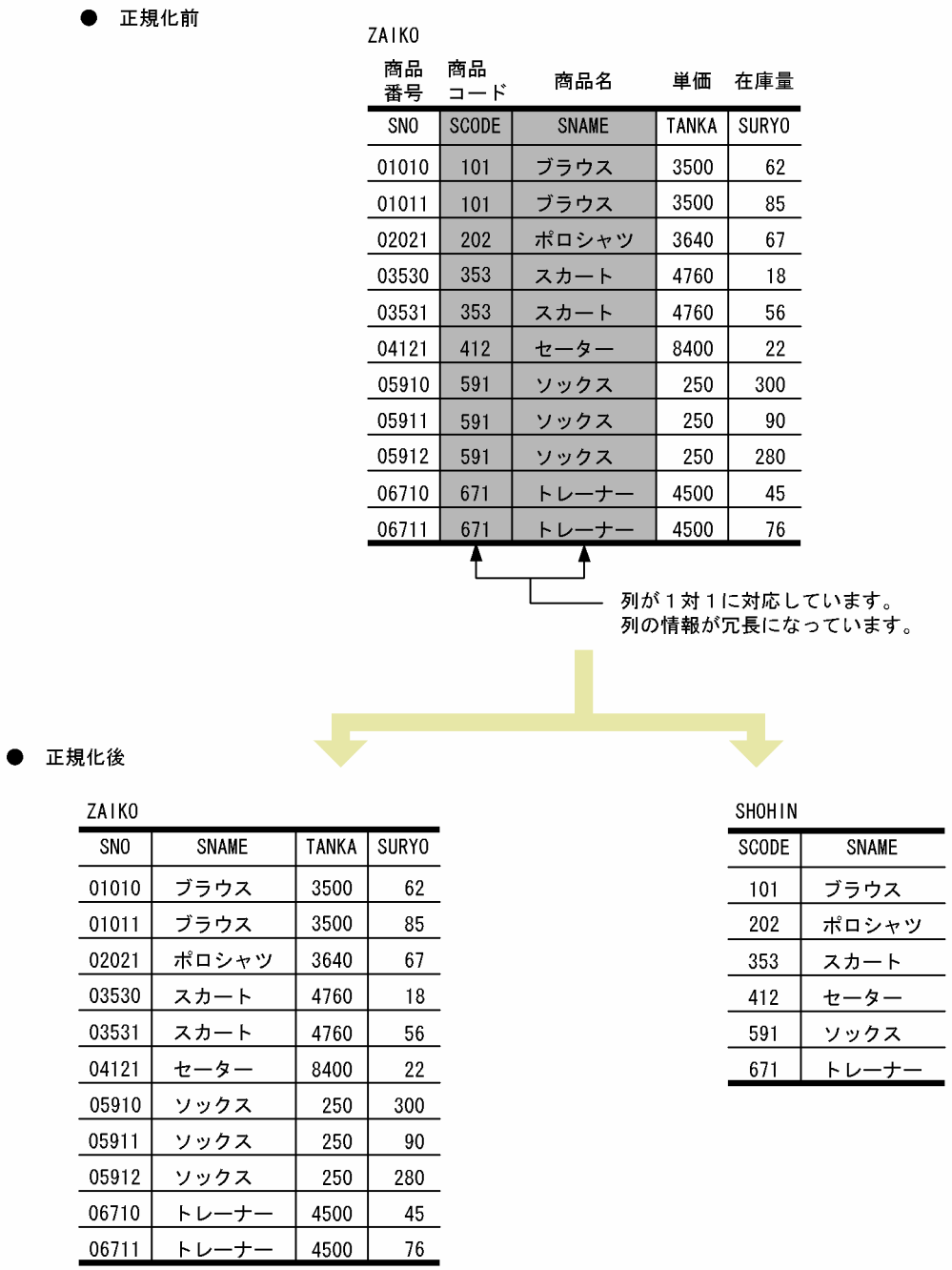
（凡例）－：該当しません。

データ型は，定義系 SQL の **CREATE TABLE** で表を作成するときに指定します。「抽象データ型を構成する属性」に対応させるデータ型は，定義系 SQL の **CREATE TYPE** で指定します。抽象データ型については，「オブジェクトリレーショナルデータベースへの拡張」を参照してください。

3.3.2　表の正規化

表中の重複するデータを別表に移動することを**表の正規化**といいます。表の格納効率又はアクセス処理の同時実行性を向上させるために，表の正規化をしてください。表の正規化を繰り返して複雑なデータベースを最適な形にしていきます。表の正規化を次の図に示します。表の正規化の詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

図 3–5　表の正規化



〔説明〕

正規化前の ZAIKO 表の SCODE の列と SNAME の列は 1：1 で対応し，それぞれの列の情報は冗長になっています。このような場合は，ZAIKO 表から，SCODE の列と SNAME の列で構成される SHOHIN 表を

ほかに作成します。このとき，SHOHIN 表では，SCODE の列と SNAME の列に重複した情報を持たないようにします。

3.3.3 FIX 属性

行長が固定の表に付ける属性を **FIX 属性**といいます。固定長でナル値がないデータの表に FIX 属性を指定すると，列の取り出し時間を短縮できます。さらに，列数が多い場合でもアクセス性能を向上できます。次に示す条件を満たす場合に，FIX 属性を指定してください。

- 列を追加しない場合
- ナル値を持つ列がない場合
- 可変長の列がない場合

表を FIX 属性にする場合は，定義系 SQL の **CREATE TABLE** で **FIX** オプションを指定します。FIX 属性の詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

3.3.4 主キー（プライマリキー）

表中の行を一意（ユニーク）に識別するためのキーとして**主キー**があります。主キーを定義した列には，**一意性制約**と**非ナル値制約**が適用されます。一意性制約とは，キー（列又は複数の列の組）中のデータの重複を許さない（キー中のデータが常に一意である）制約のことです。非ナル値制約とは，キー中の各列の値にナル値を許さない制約のことです。

表中に行を一意に識別できる列又は列の組（**候補キー**）が複数ある場合，その候補キーの中から主キーを選んでください。表中のキーの中で意味的に最も重要で，かつ一意性制約及び非ナル値制約を設定したいキーに主キーを定義します。

CREATE TABLE の **PRIMARY KEY** オプションで主キーを定義します。主キーの詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

3.3.5 クラスタキー

表中の行を昇順又は降順に格納できます。このためには**クラスタキー**を定義する必要があります。クラスタキーを定義して表中の行を昇順又は降順に格納すると，次に示す場合にデータの入出力時間を短縮できます。

- 範囲を指定して行の検索，更新，又は削除をする場合
- クラスタキー順に検索又は更新をする場合

適用基準

次に示す条件を満たす場合にクラスタキーを指定してください。

- キーの昇順又は降順にデータを蓄積する業務で，キー順にアクセスする業務が多い場合
- キーを変更しない場合
- 行長が固定の表の場合

CREATE TABLE の **CLUSTER KEY** オプションでクラスタキーを定義します。クラスタキーの詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

3.3.6　サプレスオプション

表中のデータの一部を省略して，実際のデータ長よりも短くして格納するオプションを**サプレスオプション**といいます。サプレスオプションを指定すると，データ格納時に，表中の DECIMAL データの有効けた（先頭の 0 の部分を除いたけた）部分及び格納データ長だけを格納します。サプレスオプションを指定すると，実際のデータ長よりも短くしてデータを格納できるため，ディスク所要量が削減でき，全件検索などの検索処理での入出力時間が短縮できます。

CREATE TABLE の **SUPPRESS** オプションでサプレスオプションを指定します。サプレスオプションの詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

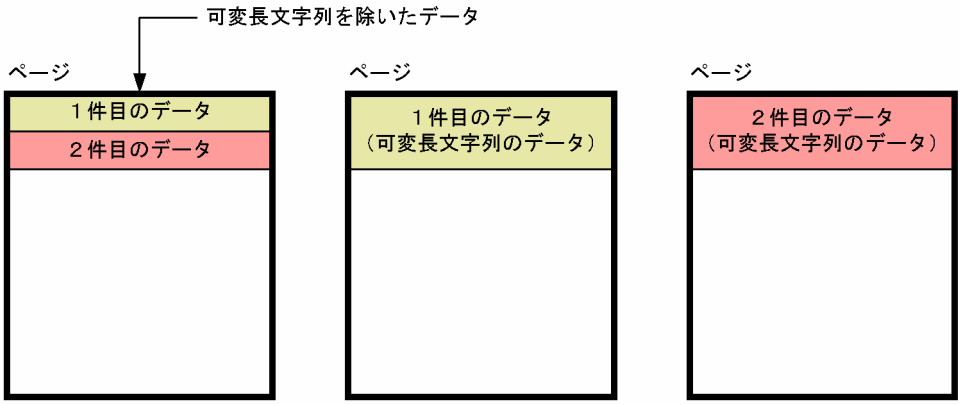
3.3.7　ノースプリットオプション

次に示すデータ型が表に定義されていて，次に示すデータ型の実際のデータ長が 256 バイト以上の場合，1 行のデータを複数のページに格納します。

- VARCHAR
- MVARCHAR
- NVARCHAR

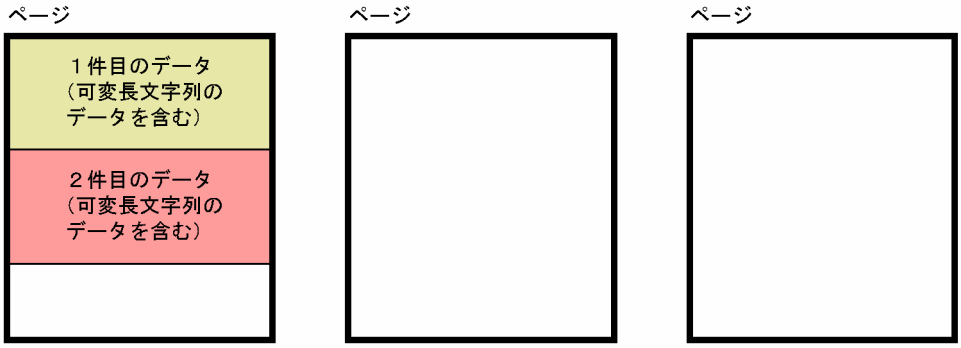
このときのデータ格納方式を次の図に示します。

図 3–6　可変長文字列の実際のデータ長が 256 バイト以上の場合のデータ格納方式



このように可変長文字列を除いたデータと可変長文字列のデータは，異なるページに格納されます。このため，データの格納効率が低下します。このような場合に，**ノースプリットオプション**を指定してデータの格納効率を向上してください。ノースプリットオプションを指定すると，可変長文字列の実際のデータ長が 256 バイト以上であっても，1 行を 1 ページに格納します。ノースプリットオプションを指定したときのデータ格納方式を次の図に示します。

図 3–7　ノースプリットオプションを指定したときのデータ格納方式



〔説明〕

1 行の全データを同じページに格納します。このため，データの格納効率がノースプリットオプションを指定しないときに比べて向上します。

CREATE TABLE 又は **CREATE TYPE** の **NO SPLIT** オプションで，ノースプリットオプションを指定します。ノースプリットオプションの詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

3.3.8　表の横分割

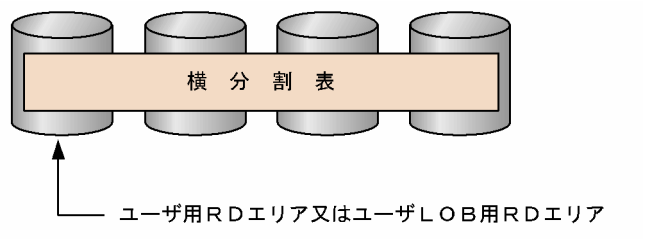
一つの表を複数のユーザ用 RD エリア又はユーザ LOB 用 RD エリアに分割して格納することを**表の横分割**といいます。また，横分割した表を**横分割表**といいます。表を横分割すると，ユーザ用 RD エリア又はユーザ LOB 用 RD エリア単位に，表へのデータの格納，表の再編成，バックアップの取得などの運用ができます。

例えば，UAP の種類（業務の種類）ごとに表を横分割して RD エリアに格納すると，バックアップの取得時にはバックアップ対象 RD エリアにアクセスする UAP だけを停止すればよく，運用の操作性が向上します。

また，HiRDB/パラレルサーバの場合には，表にアクセスする処理を複数のバックエンドサーバ下のユーザ用 RD エリア又はユーザ LOB 用 RD エリアにわたって並列化できるため，表に対するアクセスの高速化と負荷の分散ができます。

表の横分割を次の図に示します。

図 3–8　表の横分割



表を横分割する方法には，次に示す 2 種類があります。

- キーレンジ分割
- ハッシュ分割（フレキシブルハッシュ分割，FIX ハッシュ分割）

(1)　キーレンジ分割

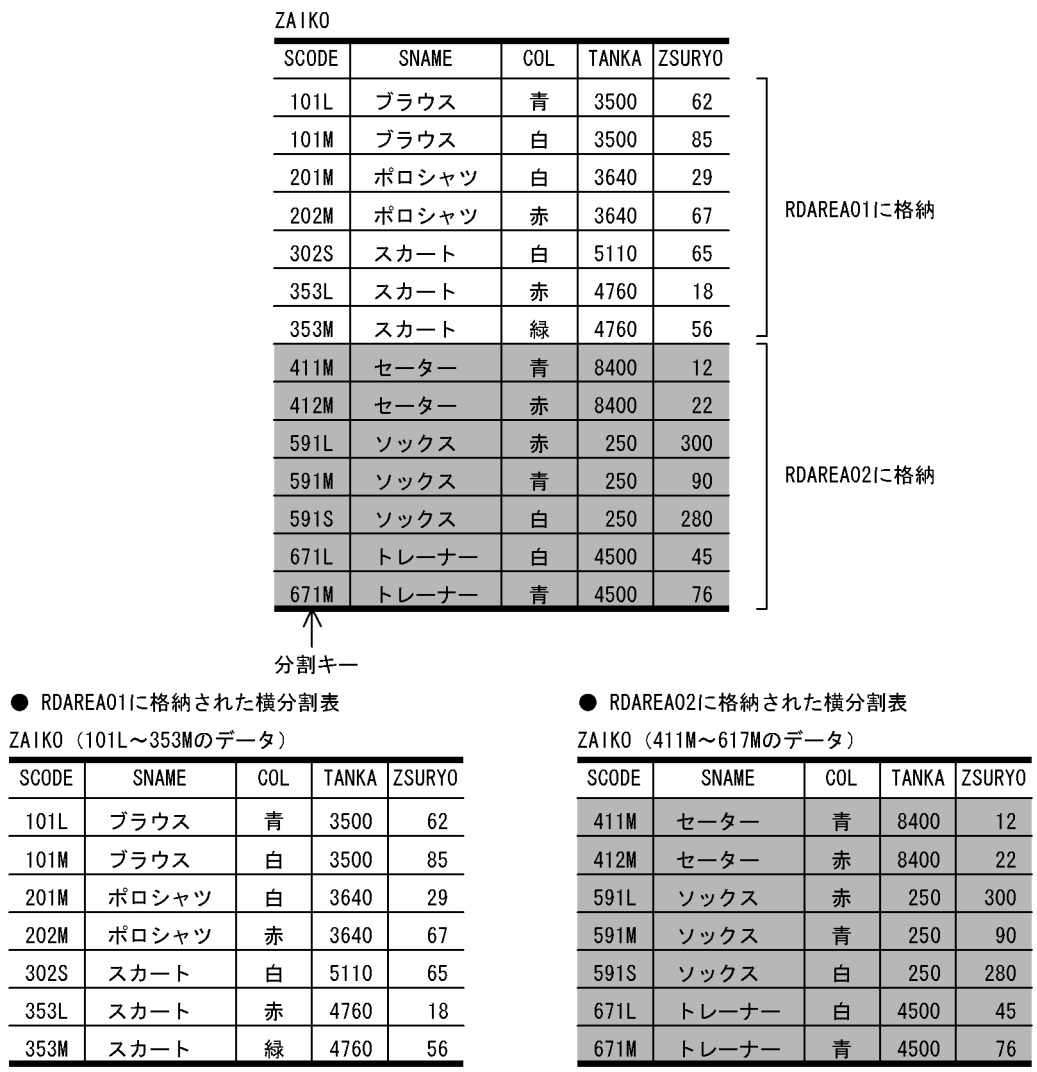
表を構成する列のうち，特定の列が持つ値の範囲を条件として表を横分割することを**キーレンジ分割**といいます。表を横分割するときの条件にした特定の列を**分割キー**といいます。キーレンジ分割は，表のデータがどの RD エリアに格納されているかどうかを意識したい場合に使用します。横分割の指定方法には，次に示す 2 種類があります。

- 格納条件指定
- 境界値指定

(a)　格納条件指定

比較演算子を使用して，それぞれの RD エリアへの格納条件を指定します。一つの RD エリアに対して，格納条件で指定された一つの範囲だけを指定できます。キーレンジ分割（格納条件指定）の例を次の図に示します。

図 3-9　キーレンジ分割（格納条件の指定）の例



〔説明〕

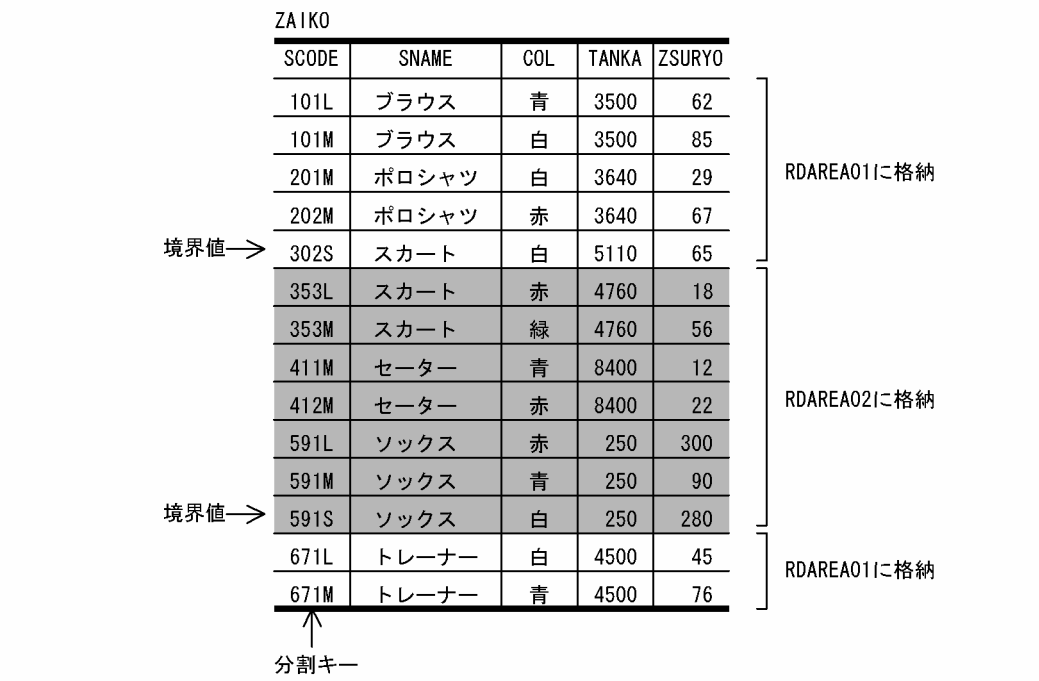
SCODE を分割キーとして在庫表を横分割します。格納 RD エリアは RDAREA01 と RDAREA02 とします。

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4) NOT NULL,SNAME NCHAR(8),
    COL NCHAR(1),TANKA INTEGER,ZSURYO INTEGER
    )IN ((RDAREA01)SCODE<='353M',(RDAREA02));
```

(b)　境界値指定

定数を使用して，それぞれの RD エリアに格納するデータの，境界となる値を昇順に指定します。一つの RD エリアに対して，境界値で区切られた複数の範囲を指定できます。キーレンジ分割（境界値指定）の例を次の図に示します。

図 3–10　キーレンジ分割（境界値指定）の例



● RDAREA01に格納された横分割表

ZAIKO

SCODE	SNAME	COL	TANKA	ZSURYO
101L	ブラウス	青	3500	62
101M	ブラウス	白	3500	85
201M	ポロシャツ	白	3640	29
202M	ポロシャツ	赤	3640	67
302S	スカート	白	5110	65
671L	トレーナー	白	4500	45
671M	トレーナー	青	4500	76

● RDAREA02に格納された横分割表

ZAIKO

SCODE	SNAME	COL	TANKA	ZSURYO
353L	スカート	赤	4760	18
353M	スカート	緑	4760	56
411M	セーター	青	8400	12
412M	セーター	赤	8400	22
591L	ソックス	赤	250	300
591M	ソックス	青	250	90
591S	ソックス	白	250	280

〔説明〕

SCODE を分割キーとして在庫表を横分割します。格納 RD エリアは RDAREA01 と RDAREA02 とします。

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4) NOT NULL,SNAME NCHAR(8),
    COL NCHAR(1),TANKA INTEGER,ZSURYO INTEGER
    )PARTITIONED BY SCODE
    IN ((RDAREA01)'302S',(RDAREA02)'591S',(RDAREA01));
```

(2)　ハッシュ分割

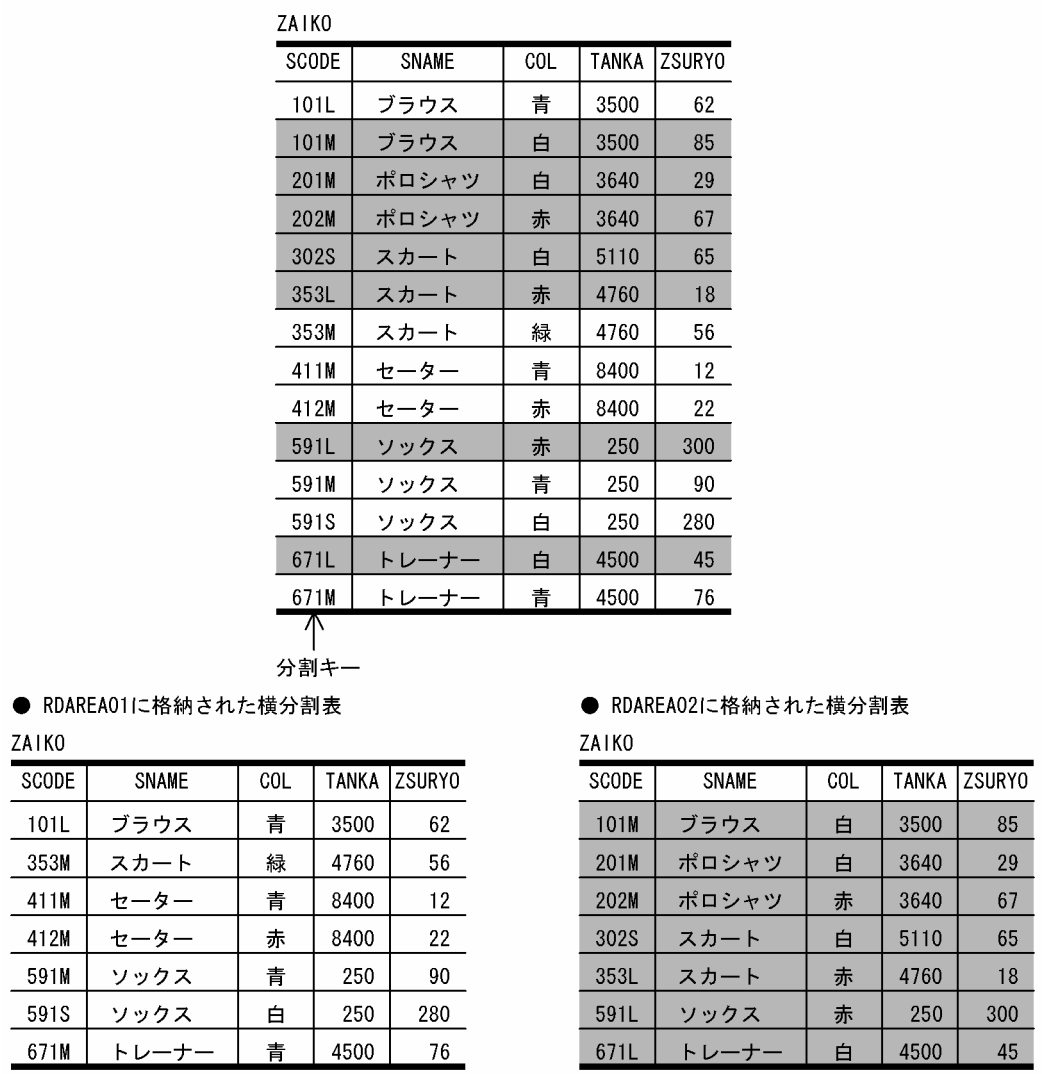
表を構成する列が持つ値をハッシュ関数を使用して，均等に RD エリアに格納し，表を横分割することを**ハッシュ分割**といいます。表を横分割するときに指定した特定の列を**分割キー**といいます。ハッシュ分割は，キーの範囲を意識しないで，表のデータを RD エリアに均等に格納したい場合に使用します。ハッシュ分割の種類を次の表に示します。

表 3–4　ハッシュ分割の種類

ハッシュ分割の種類	説明
フレキシブルハッシュ分割	表を分割して RD エリアに格納する場合，どの RD エリアに分割されるか定まりません。このため，検索処理では，該当する表があるすべてのバックエンドサーバが対象になります。
FIX ハッシュ分割	表がどの RD エリアに分割されたかを HiRDB が認識します。このため，検索処理では，該当するデータがあると予測されるバックエンドサーバだけが対象になります。

ハッシュ分割の例を次の図に示します。

図 3–11　ハッシュ分割の例



〔説明〕

SCODE を分割キーとして在庫表を横分割します。格納 RD エリアは RDAREA01 と RDAREA02 とします。なお，実際のデータの格納先 RD エリアはこの例と異なることがあります。

```
CREATE TABLE ZAIKO
    (SCODE CHAR(4) NOT NULL,SNAME NCHAR(8),
    COL NCHAR(1),TANKA INTEGER,ZSURYO INTEGER
```

```
    ) [FIX]※ HASH HASH6 BY SCODE
    IN (RDAREA01,RDAREA02);
```

注※　FIX ハッシュ分割の場合に指定します。

(3)　表の横分割の例

横分割表を格納する RD エリアは異なるディスク上に配置してください。同じディスク上に配置すると，ディスクに対するアクセスの競合が発生して性能が向上しません。

また，HiRDB/パラレルサーバの場合，**サーバ間横分割**及び**サーバ内横分割**という概念があります。複数のバックエンドサーバにわたって表が横分割される形態をサーバ間横分割といいます。それに対して，一つのバックエンドサーバ内で表が横分割される形態をサーバ内横分割といいます。**HiRDB/シングルサーバの場合は，常にサーバ内横分割になります。**

HiRDB/シングルサーバ及び HiRDB/パラレルサーバの表の横分割の例を次の図に示します。

図 3–12　表の横分割の例（HiRDB/シングルサーバの場合）

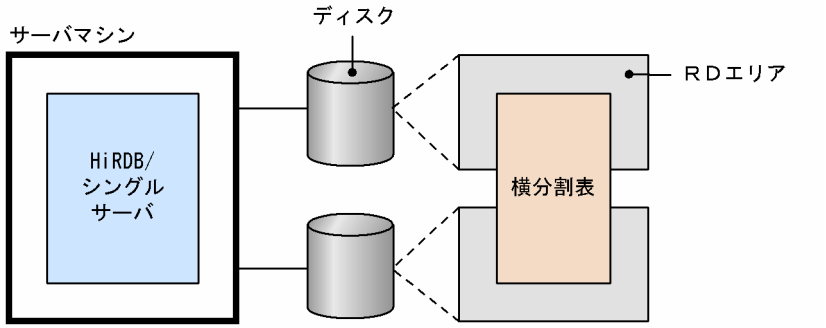
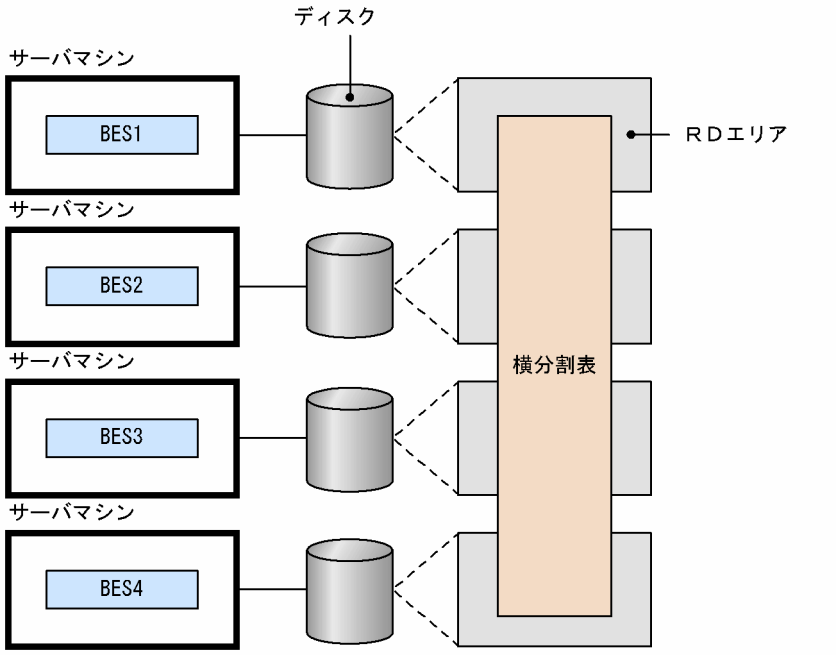


図 3–13　表の横分割の例（HiRDB/パラレルサーバの場合）



〔説明〕

BES1～BES4 に表がサーバ間横分割されています。

(4)　表の横分割の定義

表を横分割する場合には，定義系 SQL の **CREATE TABLE** に，次に示す要素を指定します。

- RD エリアへ表の割り当て
- 横分割の方法（格納条件指定，境界値指定又はハッシュ分割指定）

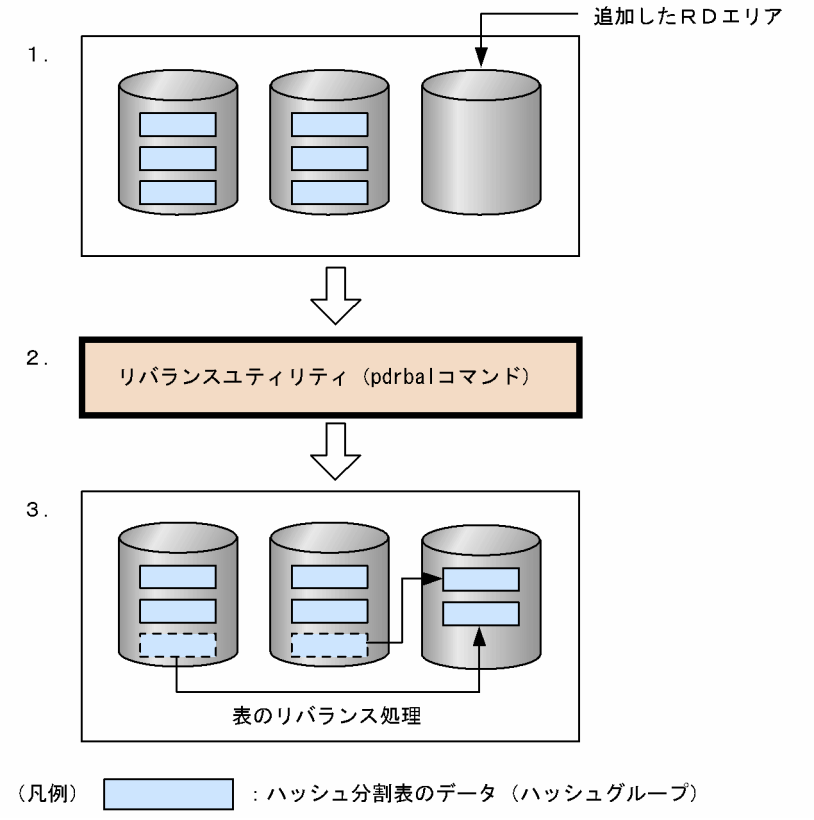
表をどのように横分割すれば処理性能が向上するかなど，表の横分割の設計方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(5)　ハッシュ分割表のリバランス機能

ハッシュ分割表のデータ量が増加したため RD エリアを追加すると（表の横分割数を増やすと），既存の RD エリアと新規追加した RD エリアとの間でデータ量の偏りが生じます。**ハッシュ分割表のリバランス機能**を使用すると，表の横分割数を増やすときにデータ量の偏りを修正できます。ハッシュ分割表のリバランス機能を次の図に示します。ハッシュ分割表のリバランス機能は，FIX ハッシュ及びフレキシブルハッシュのどちらにも適用できます。

ハッシュ分割表のリバランス機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

図 3–14　ハッシュ分割表のリバランス機能



〔説明〕

1. ハッシュ分割表のデータが一杯になったため，ハッシュ分割表を格納する RD エリアを追加しました（表の横分割数を増やしました）。追加した RD エリアにはデータが配置されず，データ量の偏りが生じます。
2. データ量の偏りを修正するために**リバランスユティリティ**（pdrbal コマンド）を実行します。
3. リバランスユティリティを実行すると，ハッシュグループ単位にデータが移動して再配置されます。これを**表のリバランス**といいます。ハッシュ分割表のリバランス機能を使用すると，分割キーをハッシュした結果を基にして HiRDB がデータを 1,024 のグループ（これを**ハッシュグループ**といいます）に分けます。このグループごとに RD エリアのセグメントを割り当ててデータを格納します。データの再配置もこのハッシュグループ単位に実行されます。

適用基準

- データの増加が見込まれていて，将来 RD エリアを追加する可能性がある場合※
- データ容量が大きいため，表を再作成するのが難しい場合

注※

データが格納されている FIX ハッシュ分割表には RD エリアを追加できませんが，ハッシュ分割表のリバランス機能を使用すると RD エリアを追加できるようになります。

運用方法

ハッシュ分割表のリバランス機能の運用手順の概略を次に示します。

1. ハッシュ分割表を定義するときにハッシュ関数 A～F を使用して，その表を**リバランス表**として定義します。
2. 表の横分割数を増やすため，表格納 RD エリアを追加します。
3. リバランスユティリティを実行して表のリバランスを行います。

3.3.9　表のマトリクス分割

表の二つの列を分割キーとして，分割方法の指定を組み合わせて分割することを**マトリクス分割**といいます。一つ目の分割キーとなる列を**第 1 次元分割列**，二つ目の分割キーとなる列を**第 2 次元分割列**といいます。マトリクス分割は，第 1 次元分割列で境界値指定のキーレンジ分割をし，分割されたデータをさらに第 2 次元分割列で分割します。第 2 次元分割列に指定できる分割方法を次に示します。

- 境界値指定のキーレンジ分割
- フレキシブルハッシュ分割
- FIX ハッシュ分割

マトリクス分割によって分割された表を**マトリクス分割表**といいます。マトリクス分割表に対応させて，インデクスもマトリクス分割できます。なお，表をマトリクス分割するためには，**HiRDB Advanced High Availability** が必要です。

(1)　表のマトリクス分割の効果

複数の列を分割キーとして分割することで得られる効果を次に示します。

- SQL 処理の高速化
  SQL の処理を並列に実行したり，複数のキーによる検索で検索範囲を絞り込んで高速に処理したりできます。
- 運用時間の短縮
  より細かな分割ができるため，1RD エリアの大きさを小さくして，再編成，バックアップの取得，障害発生時の回復作業などに必要な時間を短縮できます。

(2)　適用基準

次の場合，境界値指定のキーレンジ分割の組み合わせをお勧めします。

- 第 1 次元分割列による分割では，各境界値に該当するデータ量が膨大となる
- 表にアクセスする UAP で指定できる探索条件に指定する列が複数あり，複数の列でアクセスする RD エリアを限定したい場合，又は一つの SQL 文中で n 番目に指定した列だけでアクセスする RD エリアを限定したい場合。

次の場合，境界値指定のキーレンジ分割とハッシュ分割の組み合わせをお勧めします。

- 第 1 次元分割列による分割では，各境界値に該当するデータ量が膨大となる
- 第 1 次元分割列で分割された範囲のデータを，均等に細分化して格納したい

(3)　定義方法

定義系 SQL の **CREATE TABLE** の **PARTITIONED BY MULTIDIM** オペランドで次の指定をします。

- RD エリアへの表の割り当て
- マトリクス分割の方法（分割キー，分割方法）

(4)　マトリクス分割の例

(a)　境界値指定のキーレンジ分割の組み合わせの場合

顧客表の登録日及び店番号に境界値を指定し，登録日，店番号によって，それぞれの顧客データを次のようにユーザ用 RD エリア（USR01～USR06）に格納するように表をマトリクス分割します。格納するのに必要なユーザ用 RD エリア数は，（境界値数＋ 1）×（境界値数＋ 1）なので，この例の場合，3×2 ＝ 6 です。

登録日	店番号	
	100 以下	101 以上
2000 年以前	USR01	USR02
2001 年	USR03	USR04
2002 年以降	USR05	USR06

マトリクス分割する表を定義する SQL 文を次に示します。

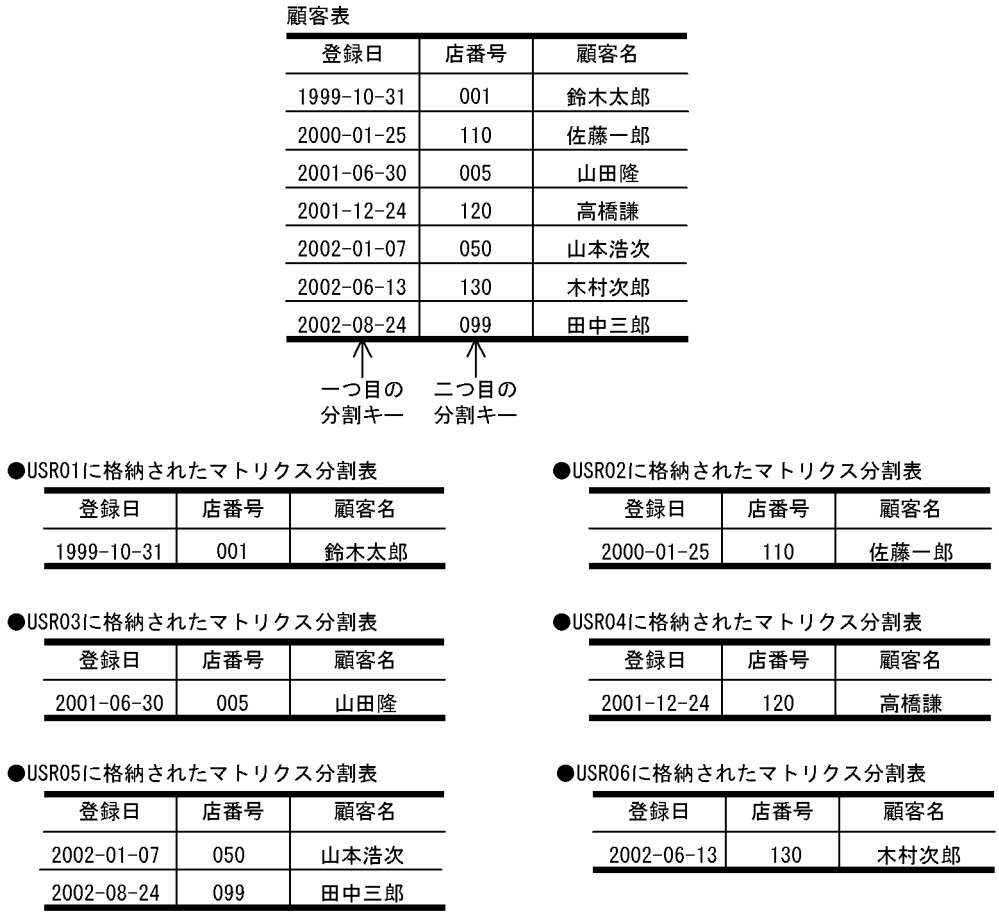
```
CREATE FIX TABLE 顧客表
  (登録日 DATE, 店番号 INT, 顧客名 NCHAR(10))
  PARTITIONED BY MULTIDIM(
  登録日 (('2000-12-31'),('2001-12-31')), …1.
  店番号 ((100))                          …2.
  )IN ((USR01,USR02),(USR03,USR04),(USR05,USR06))
```

〔説明〕

1. 第 1 次元分割列名（一つ目の分割キーとなる列名）と，その境界値リストを指定します。
2. 第 2 次元分割列名（二つ目の分割キーとなる列名）と，その境界値リストを指定します。

マトリクス分割の例を次の図に示します。

図 3-15　マトリクス分割の例（境界値指定のキーレンジ分割の組み合わせ）



(b)　境界値指定のキーレンジ分割とハッシュ分割の組み合わせの場合

第 2 次元分割列で FIX ハッシュ分割する場合の例について説明します。

顧客表の登録日に境界値を指定し，店番号と地域コードをハッシュ関数で 3 分割すると，それぞれの顧客データを次のようにユーザ用 RD エリア（USR01～USR09）に格納するように表をマトリクス分割します。格納するのに必要なユーザ用 RD エリア数は，（境界値数＋ 1）×（ハッシュ関数で分割するユーザ任意の数）なので，この例の場合，3×3 ＝ 9 です。

登録日	店番号と地域コード（ハッシュ関数で 3 分割）		
2002 年以前	USR01	USR02	USR03
2003 年	USR04	USR05	USR06
2004 年以降	USR07	USR08	USR09

マトリクス分割する表を定義する SQL 文を次に示します。

```
CREATE FIX TABLE 顧客表
  (登録日 DATE, 店番号 INT, 地域コード INT, 顧客名 NCHAR(10))
  PARTITIONED BY MULTIDIM
  (登録日 (('2002-12-31'),('2003-12-31')), …1.
  FIX HASH HASH6 BY 店番号, 地域コード      …2.
```

```
  )IN ((USR01,USR02,USR03),
       (USR04,USR05,USR06),
       (USR07,USR08,USR09))
```

〔説明〕

1. 第 1 次元分割列名（一つ目の分割キーとなる列名）と，その境界値リストを指定します。
2. 第 2 次元列分割名（二つ目の分割キーとなる列名）と，ハッシュ関数名を指定します。

マトリクス分割の例を次の図に示します。

図 3–16　マトリクス分割の例（境界値指定のキーレンジ分割とハッシュ分割の組み合わせ）

顧客表

登録日	店番号	地域コード	顧客名
2002-01-31	001	01	鈴木太郎
2002-05-25	110	03	佐藤一郎
2002-06-30	005	01	山田隆
2003-01-24	120	03	高橋謙
2003-03-07	050	01	山本浩次
2003-04-13	130	03	木村次郎
2003-08-24	099	02	田中三郎
2003-11-15	010	01	山口美佐
2003-12-24	020	01	大田五郎
2004-01-27	080	02	斎藤和美
2004-03-24	170	04	木下美智子
2004-07-24	015	01	大川靖男

一つ目の分割キー　二つ目の分割キー　二つ目の分割キー

●USR01に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2002-01-31	001	01	鈴木太郎

●USR02に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2002-05-25	110	03	佐藤一郎

●USR03に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2002-06-30	005	01	山田隆

●USR04に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2003-01-24	120	03	高橋謙
2003-08-24	099	02	田中三郎

●USR05に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2003-03-07	050	01	山本浩次
2003-11-15	010	01	山口美佐

●USR06に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2003-04-13	130	03	木村次郎
2003-12-24	020	01	大田五郎

●USR07に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2004-01-27	080	02	斎藤和美

●USR08に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2004-03-24	170	04	木下美智子

●USR09に格納されたマトリクス分割表

登録日	店番号	地域コード	顧客名
2004-07-24	015	01	大川靖男

3.3.10　表の分割格納条件の変更

キーレンジ分割※で横分割した表の分割格納条件を，ALTER TABLE で変更できます。表の分割格納条件を変更することで，古いデータを格納していた RD エリアを再利用でき，作業時間を短くできます。ALTER TABLE を使用すると，分割格納条件を変更する表を削除し，再作成する必要がありません。

注※

次に示す分割方法の場合に，表の分割格納条件を ALTER TABLE で変更できます。

- 境界値指定
- 格納条件指定（格納条件の比較演算子に＝だけを使用している場合）
- マトリクス分割（境界値指定のキーレンジ分割の次元だけ）

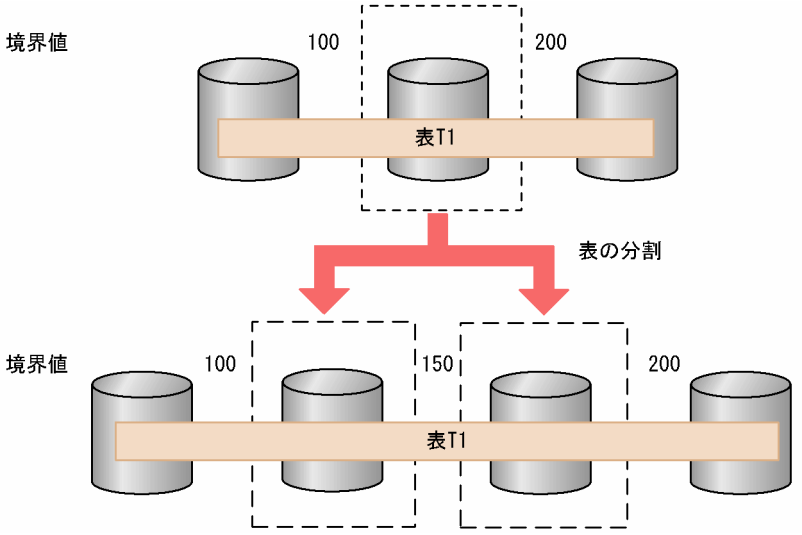
なお，表の分割格納条件を変更するには，**HiRDB Advanced High Availability** が必要です。

表の分割格納条件の変更には，次の二つの機能があります。

(1)　分割機能

表の分割格納条件を変更し，一つの RD エリアに格納していたデータを複数の RD エリアに分割して格納します。分割機能の例（境界値指定の場合）を次の図に示します。

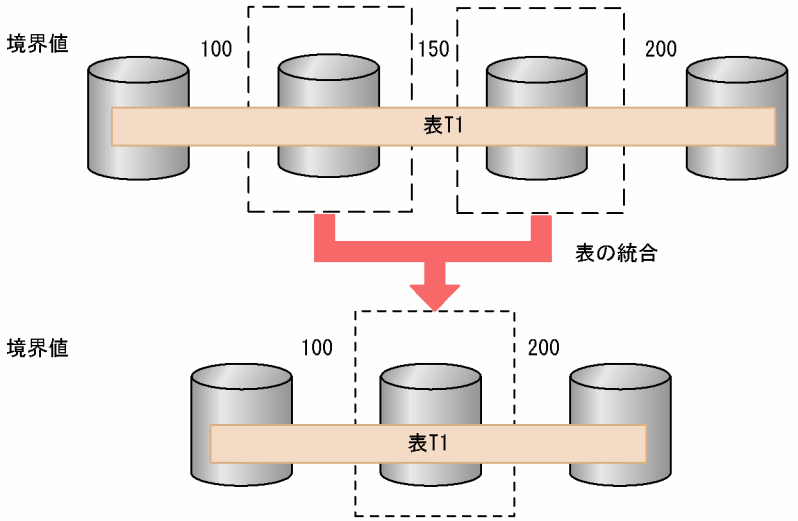
図 3–17　分割機能の例（境界値指定の場合）



(2)　統合機能

表の分割格納条件を変更し，複数の RD エリアに格納していたデータを一つの RD エリアに統合して格納します。統合機能の例（境界値指定の場合）を次の図に示します。

図 3–18　統合機能の例（境界値指定の場合）



3.3.11　改竄防止表

重要なデータを人為的ミスや不正な改竄から守るため，表の所有者も含め，すべてのユーザに対して表データの更新を禁止した改竄防止表を定義できるようにしました。改竄防止表に対する操作の実行可否を次の表に示します。

表 3–5　改竄防止表に対する操作の実行可否

操作	改竄防止表	
	行削除禁止期間指定あり	行削除禁止期間指定なし
挿入（INSERT）	○	○
検索（SELECT）	○	○
列単位の更新（UPDATE）	○※1	○※1
行単位の更新（UPDATE）	×	×
削除（DELETE）	○※2	×
全行削除（PURGE TABLE）	×	×
上記以外の操作系 SQL	○	○

（凡例）

○：実行できます。

×：実行できません。

注※1

更新可能列だけ更新できます。

注※2

行削除禁止期間を経過しているデータだけ削除できます。行削除禁止期間を指定しないと，表データを削除できません。

(1) 指定方法

定義系 SQL の **CREATE TABLE** で **INSERT ONLY** オプション（改竄防止オプション）を指定します。また，**ALTER TABLE** の **INSERT ONLY** オプションを指定して，既存の表を改竄防止表に定義変更することもできます。

表定義時又は定義変更時に次の列を定義できます。

- **更新可能列**

  更新可能列を定義すると，列単位に，次の方法でデータを更新できます。

  - 常に更新できる（UPDATE を指定）
  - ナル値から非ナル値へ一度だけ更新できる（UPDATE ONLY FROM NULL を指定）

  更新可能列を定義できるのは，次の時点です。

  - CREATE TABLE 実行時
  - ALTER TABLE CHANGE INSERT ONLY 実行前
  - ALTER TABLE ADD 列名，又は ALTER TABLE CHANGE 列名※ 実行時

  注※

  ALTER TABLE CHANGE 列名は，改竄防止表に対しては実行できません。

- **挿入履歴保持列**

  挿入履歴保持列を定義すると，**行削除禁止期間**を指定できます。行削除禁止期間を省略すると，表データは一切削除できません。また，表にデータがあると DROP TABLE が実行できないため，行削除禁止期間を省略した場合，表及び表データは共に削除できません。そのため，データの保存期間が決まっている場合，又は決められる場合は行削除禁止期間を指定してください。

(2) 制限事項

改竄防止表及び改竄防止表格納 RD エリアに対する制限事項を次に示します。詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

- すべての列に更新可能列属性を指定し，かつ改竄防止オプションを指定することはできません。
- 既存の列を更新可能列に変更したり，更新可能列を通常の列に変更したりすることはできません。
- 更新可能列は，改竄防止機能を適用する前に定義しておく必要があります。
- 既存の表を改竄防止表に変更する場合，データが存在する表に対して改竄防止機能は適用できません。まず，既存の表のデータをアンロードして，改竄防止表に変更してからデータをロードする必要があります。

- データがある改竄防止表は削除できません。行削除禁止期間を指定していない改竄防止表はデータが削除できないため，表も削除できません。
- 改竄防止表格納 RD エリアに対して，データベース構成変更ユティリティ（pdmod）による RD エリアの再初期化（initialize rdarea）はできません。
- 改竄防止表格納 RD エリアにはインナレプリカ機能（UNIX 版限定）を使用できません。また，インナレプリカ機能を使用している RD エリアに格納されている表に，改竄防止機能は適用できません。
- 改竄防止表に対して，レプリケーション機能（HiRDB Dataextractor 及び HiRDB Datareplicator）を使用してデータの複写，及び更新結果を反映しないでください。使用すると，反映元と反映先でデータの内容が不一致になったり，エラーになったりすることがあります。

3.3.12　採番業務で使用する表

伝票番号や書類の番号などを管理しながら，その番号を利用してほかの処理をする業務のことを**採番業務**といいます。採番業務で使用する表とは，採番の番号だけを管理する表のことをいいます。この場合，複数の利用者が同時に同じ表を使用することが多いため，更新処理での排他の影響を少なくし，更新管理用時期を早くする必要があります。

採番業務で使用する表には排他の影響を少なくするため，トランザクションのコミットを待たないで，表への更新処理が完了した時点でその行への排他を解除し，それ以降はロールバックされなくなるという機能を提供しています。

採番業務で使用する表を定義する場合は，**CREATE TABLE** で **WITHOUT ROLLBACK** オプションを指定します。採番業務で使用する表の利用方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

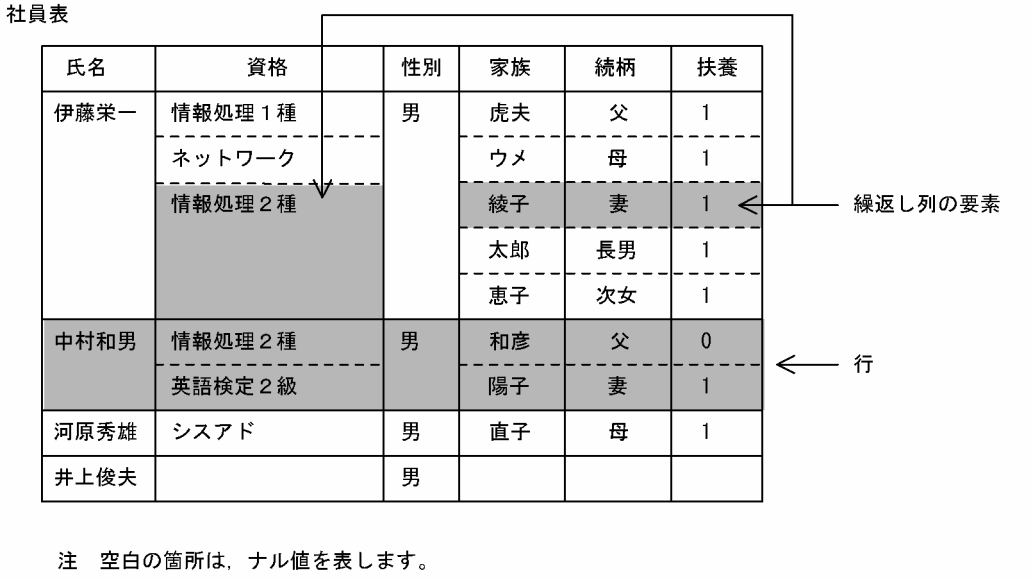
3.3.13　繰返し列

繰返し列とは，複数の要素から構成される列のことです。**要素**とは，繰返し列中で繰り返されている各項目のことです。繰返し列は **CREATE TABLE** で定義し，要素数も同時に定義します。ただし，要素数は後から **ALTER TABLE** で増やせます。繰返し列を含む表を定義する利点を次に示します。

- 複数の表で重複している情報が多い場合に重複情報がなくなります。これによって，ディスク容量を削減できます。
- 複数の表の結合が不要になります。
- 関連データ（繰り返しデータ）が近くに格納されるため，別の表にするよりアクセス性能が優れています。

繰返し列を含む表の例を次の図に示します。なお，繰返し列を含む表については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

図 3–19　繰返し列を含む表の例



注　空白の箇所は，ナル値を表します。

(1)　繰返し列の定義例

図「繰返し列を含む表の例」の繰返し列を含む表を定義する SQL 文（CREATE TABLE）を次に示します。

```
CREATE TABLE 社員表
   (氏名 NVARCHAR(10),
    資格 NVARCHAR(20) ARRAY[10],
    性別 NCHAR(1),
    家族 NVARCHAR(5) ARRAY[10],
    続柄 NVARCHAR(5) ARRAY[10],
    扶養 SMALLINT ARRAY[10]);
```

(2)　繰返し列に対する操作

繰返し列を含む表に対して，HiRDB では次に示す操作ができます。

構造化繰返し述語を使用した検索

繰返し列がある表の，添字が同じ要素で対応している複数の繰返し列に対して条件を指定して検索する場合，構造化繰返し述語を使用します。

（例 1）

```
SELECT 氏名 FROM 社員表 WHERE ARRAY(続柄,扶養）[ANY]
       （続柄='父' AND 扶養=1）
```

繰返し列の更新

更新する場合，次に示す三つの方法があります。

- 既存の要素を更新する（UPDATE 文の SET 句）
- 新たに要素を追加する（UPDATE 文の ADD 句）

- 既存の要素を削除する（UPDATE 文の DELETE 句）

繰返し列がある表を更新する場合，繰返し列名 [{添字 | *}]で更新する繰返し列の要素を指定します。

（例 2）既存の要素を更新する場合

```
UPDATE 社員表 SET 資格[2]=N'英語検定1級'
       WHERE 氏名=N'中村和男'
```

（例 3）新たに要素を追加する場合

```
UPDATE 社員表 ADD 資格[*]=ARRAY[N'データベース']
       WHERE 氏名=N'中村和男'
```

（例 4）既存の要素を削除する場合

```
UPDATE 社員表 DELETE 資格[2] WHERE 氏名=N'中村和男'
```

そのほかの繰返し列に対する操作

SQL 文中で繰返し列を指定する場合は，**繰返し列名[添字]**で指定します。

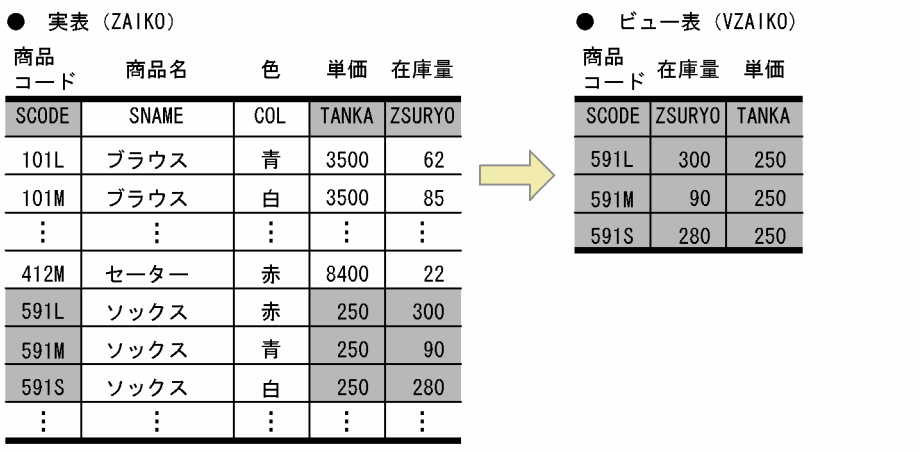
繰返し列を含む表の操作については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

3.3.14　ビュー表

実際にデータベースに格納している表（**実表**）から特定の行や列を選択して，新たに定義した仮想の表を作成できます。この仮想の表を**ビュー表**といいます。特定の列だけを公開するビュー表を作成すれば，データの機密保護に効果的です。また，操作する列が少なくなるため，表の操作性が向上します。

ビュー表を作成するには，定義系 SQL の **CREATE VIEW** を実行します。ビュー表の例を次の図に示します。

図 3–20　ビュー表の例



〔説明〕

実表（ZAIKO）から，商品名（SNAME）がソックスの行である，商品コード（SCODE），在庫量（ZSURYO）及び単価（TANKA）の列で構成されるビュー表（VZAIKO）を作成しています。なお，列の並びは，商品コード，在庫量，単価の順とします。

```
CREATE VIEW VZAIKO
    AS SELECT SCODE,ZSURYO,TANKA
    FROM ZAIKO
    WHERE SNAME = N'ソックス';
```

3.3.15　共用表

HiRDB/パラレルサーバの場合，複数の表を結合するとき，それぞれの表が配置されたバックエンドサーバから表データを読み込み，別のバックエンドサーバで突き合わせ処理をします。そのため，複数のサーバを接続し，データを転送する処理が発生します。このとき，結合処理のための検索範囲のデータが一つのバックエンドサーバにあれば，そのデータを共用表として作成することで一つのバックエンドサーバで結合処理が完結します。**共用表**とは，**共用 RD エリア**に格納された表で，すべてのバックエンドサーバから参照できる表のことです。また，共用表に定義するインデクスを，**共用インデクス**といいます。共用表については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

共用表及び共用インデクスは HiRDB/シングルサーバにも定義できます。これによって，HiRDB/パラレルサーバと SQL 及び UAP の互換性を保てます。共用表及び共用インデクスは HiRDB/パラレルサーバで有効なので，通常は HiRDB/パラレルサーバで使用します。ここでは，HiRDB/パラレルサーバで共用表を使用する場合について説明します。HiRDB/シングルサーバで共用表を使用する場合については，「HiRDB/シングルサーバで共用表を使用する場合の注意事項」を参照してください。

(1)　共用表の効果

一つのバックエンドサーバで結合処理が完結するため，バックエンドサーバ間の接続やデータ転送によるオーバヘッドが削減できます。また，トランザクションごとに使用するバックエンドサーバ数を少なくできるため，多重実行時などに並列処理の効率が上がります。

(2)　適用基準

更新処理が少なく，結合処理など複数のトランザクションから参照されるような表は，共用表として作成することをお勧めします。

(3)　共用表を更新する場合の注意

共用表を更新する場合，共用 RD エリアを配置している全バックエンドサーバに排他が掛かります。インデクスキー値を変更しない UPDATE 文，PURGE TABLE 文の実行以外は LOCK TABLE 文で IN EXCLUSIVE MODE を指定し，全バックエンドサーバに排他を掛けなければ実行できません。そのため，排他が掛かっている RD エリアにアクセスする業務があるとデッドロック，又はサーバ間のグローバルデッドロックが発生する可能性があります。

なお，ローカルバッファを使用して共用表を更新する場合は，LOCK 文を発行して更新してください。LOCK 文を発行しない更新をしていて，サーバプロセスが異常終了すると，アボートコード Phb3008 が出力されます（ユニットが異常終了することがあります）。

(4) 定義方法

定義系 SQL の **CREATE TABLE** で **SHARE** を指定（CREATE SHARE FIX TABLE と指定）します。なお，共用表は次の条件を満たす必要があります。

- 共用表は非分割の FIX 表である
- 共用表，及び共用インデクスを格納する RD エリアは共用 RD エリア（pdfmkfs コマンドの-k オプションに SDB を指定）である
- WITHOUT ROLLBACK オプションが指定されていない
- 参照制約が定義された参照表でない

(5) 制限事項

共用表を使用する場合の制限事項を次に示します。

- IN EXCLUSIVE MODE 指定の LOCK TABLE 文実行中は共用表を検索できません。
- 共用表に対しては，ASSIGN LIST 文でリストを作成できません。
- レプリケーションの反映先に共用表は指定できません。

(6) HiRDB/シングルサーバで共用表を使用する場合の注意事項

- HiRDB/シングルサーバでは共用 RD エリアを定義できないため，共用表及び共用インデクスは，通常のユーザ用 RD エリアに格納してください。このとき，共用表及び共用インデクスを格納するユーザ用 RD エリアと，共用表ではない表及び共用インデクスではないインデクスを格納するユーザ用 RD エリアは別にしてください。
- HiRDB/シングルサーバから HiRDB/パラレルサーバに移行するとき，共用表及び共用インデクスが定義されている状態でデータベース構成変更ユティリティ（pdmod）を使用しないでください。移行手順については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

HiRDB/パラレルサーバの場合との相違の詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

3.3.16 圧縮表

HiRDB が表にデータを格納するとき，データを圧縮して格納できます。これを**データ圧縮機能**といいます。データの圧縮は列単位に指定でき，圧縮を指定した列を**圧縮列**といい，圧縮列がある表を**圧縮表**といいます。圧縮表については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

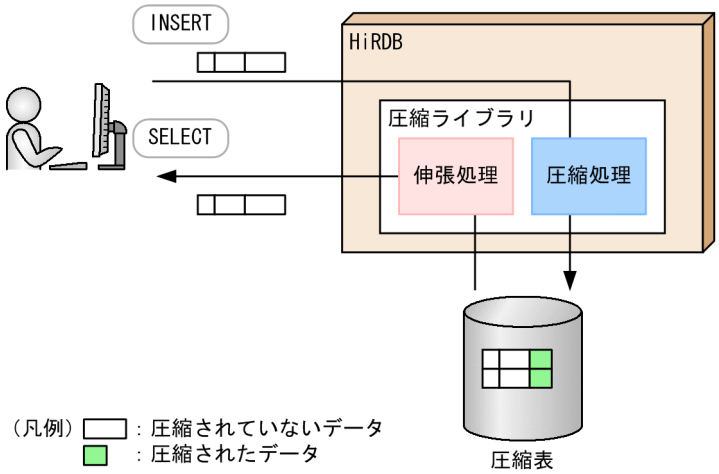
HiRDB がデータを圧縮することで，次のメリットがあります。

- データベースの容量を削減できる。

- UAP 側でデータ圧縮処理を実装する必要がない。

データ圧縮の概要を次の図に示します。

図 3–21　データ圧縮の概要



〔説明〕

HiRDB がデータの圧縮及び伸張処理を実行するため，ユーザはデータの圧縮及び伸張を指示する必要はありません。

(1)　適用基準

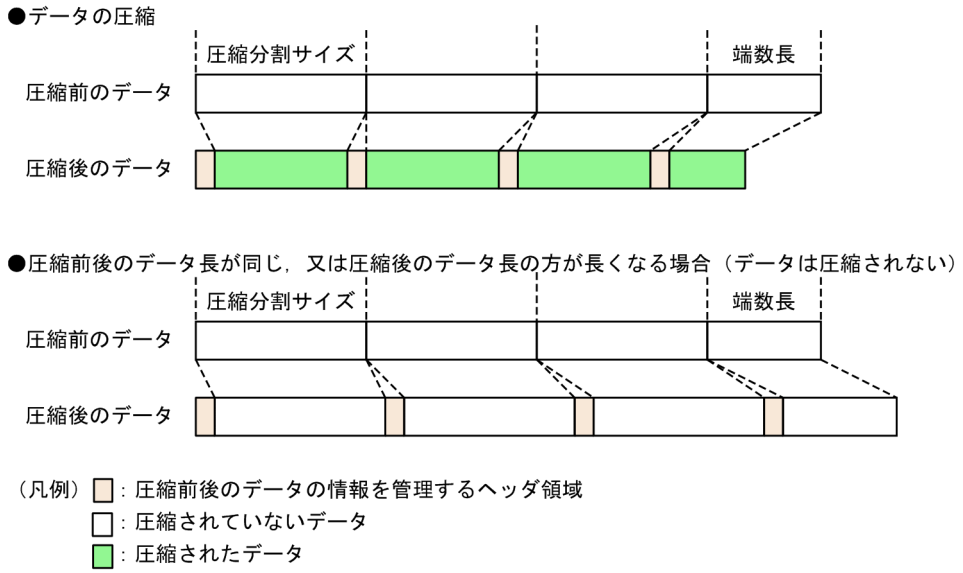
画像，音声など，容量が大きい可変長バイナリデータを含む表を圧縮表にすることをお勧めします。ただし，圧縮表の場合，圧縮処理や伸張処理によるオーバヘッドが掛かります。このため，性能よりも格納効率を重視するシステムで使用してください。

(2)　データ圧縮の仕組み

データ圧縮時に使用する圧縮ライブラリは zlib です。HiRDB は，zlib を使用して表定義時に指定した**圧縮分割サイズ**（省略値：MIN（32,000 バイト，圧縮列の定義長））ごとに圧縮します。このとき，圧縮前後のデータの情報を管理するヘッダ領域（8 バイト）を圧縮分割サイズごとに追加します（zlib が圧縮データに付与するヘッダ領域とは別に追加します）。

ただし，圧縮前後のデータ長が同じ，又は圧縮後のデータ長の方が長くなる場合，HiRDB はデータを圧縮しないで格納します。このため，ヘッダ領域の付与によって，圧縮後のデータサイズが圧縮前よりも大きくなることがあります。圧縮前後のデータを次の図に示します。

図 3–22　圧縮前後のデータ



(3)　圧縮表の定義方法

定義系 SQL の **CREATE TABLE** の列定義で圧縮指定（**COMPRESSED**）をします。必要に応じて，圧縮分割サイズも指定します。ただし，圧縮指定には次の条件があります。

- 圧縮指定は列単位に指定します（表単位に指定することはできません）。
- 圧縮指定できるのは，次のデータ型の列だけです
  - 定義長が 256 バイト以上の BINARY 型
  - 抽象データ型（XML 型）※

注※

抽象データ型（XML 型）の列のデータを圧縮するためには，バージョン 09-03 以降の HiRDB XML Extension が必要です。

(4)　圧縮表使用時の留意事項

- 圧縮列のデータを SQL やユティリティで操作する場合，圧縮処理や伸張処理のオーバヘッドが掛かります。
- 圧縮対象となるデータの実態によって異なりますが，圧縮分割サイズが大きいほどデータの圧縮効率は上がります。ただし，圧縮分割サイズを大きくすると，圧縮列に対するデータの格納及び抽出が発生する SQL（SUBSTR 関数や POSITION 関数を使用したり，後方削除更新をしている SQL）を実行する場合に SQL 実行時に確保するプロセス固有領域が増加します。
- 共有モードで圧縮表のリバランスを実行する場合，データの伸張及び圧縮を行うため，圧縮列がない場合に比べてリバランス処理に掛かる時間が長くなることがあります。実行時間を短縮したい場合は，占有モードで実行してください。

3.3.17　一時表

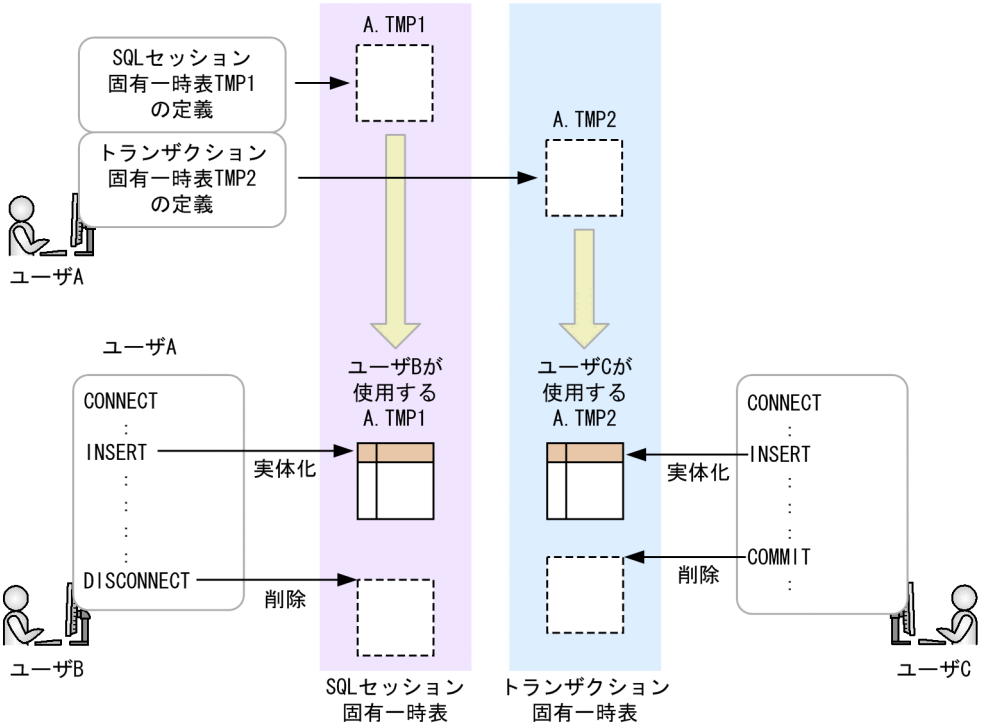
一時表は，トランザクション又は SQL セッションの期間中だけ存在する実表です。トランザクションの期間中だけ存在する一時表を**トランザクション固有一時表**，SQL セッションの期間中だけ存在する一時表を**SQL セッション固有一時表**といいます。

一時表は，表定義時点では作成されません。一時表に対して，最初に INSERT 文が実行されたときに表が作成されます。このことを，一時表の**実体化**といいます。

また，一時表は一つの表定義に対して接続（CONNECT 文実行）ごとに専用の表が作成されるため，複数ユーザが同時に使用しても，ほかのユーザのデータ操作（参照，挿入，更新，又は削除）の影響を受けません。一時表及び一時表に定義したインデクス（**一時インデクス**）は，**一時表用 RD エリア**に格納され，トランザクションの決着時又は SQL セッションの終了時に，自動的に削除されます。

一時表の概要を次の図に示します。

図 3–23　一時表の概要



一時表の効果

- トランザクション又は SQL セッションで複雑な処理を行う場合，中間処理結果を一時的に保持し，さらに加工して最終的な結果を得るなど，作業用の表として使用できます。
- データ件数が多い表の一部のデータに対してトランザクション又は SQL セッション内で頻繁にアクセスする場合，該当するデータを一時表に格納することで入出力回数を削減でき，性能向上できます。
- 一時表は，トランザクションの決着時又は SQL セッションの終了時に自動的に削除されるため，UAP による後処理が不要になり，UAP 作成の負担が軽減できます。

一時表の適用基準

データ件数が多い表の一部にだけアクセスが頻繁にあるトランザクションや，

中間処理結果を一時的に保存する必要があるような複雑な処理をするバッチ業務などに一時表の使用をお勧めします。

一時表については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(1)　一時表のデータ有効期間

実体化された一時表のデータ有効期間（実体が存在する期間）は，その一時表がトランザクション固有一時表か，SQL セッション固有一時表かによって異なります。一時表のデータ有効期間の開始及び終了タイミングを次の表に示します。

表 3–6　一時表のデータ有効期間の開始と終了

一時表の種類	開始となるタイミング	終了となるタイミング
トランザクション固有一時表	トランザクション中で，一時表に対して最初に INSERT 文が実行されたとき	トランザクションが決着したとき
SQL セッション固有一時表	SQL セッション中で，一時表に対して最初に INSERT 文が実行されたとき	• SQL セッションが終了したとき • 一時表を実体化したバックエンドサーバが終了したとき • 一時表を実体化したバックエンドサーバがあるユニットが終了したとき • 一時表を実体化したバックエンドサーバ，又はそのバックエンドサーバがあるユニットに対して系切り替えが発生したとき

注意事項

一時表のデータ有効期間外に，一時表に対して検索，更新，及び削除を実行しても，データがない表に対して SQL を実行したときと同じ結果になります。

(2)　一時表及び一時インデクスの定義方法

(a)　一時表を定義する場合

定義系 SQL の **CREATE TABLE** で **GLOBAL TEMPORARY** を指定します。トランザクション固有一時表を定義する場合は **ON COMMIT DELETE ROWS** を，SQL セッション固有一時表を定義する場合は **ON COMMIT PRESERVE ROWS** を指定します。なお，一時表の場合，指定できない，又は指定しても無視されるオペランドがあります。詳細については，マニュアル「HiRDB SQL リファレンス」の CREATE TABLE を参照してください。

(b)　一時インデクスを定義する場合

基本的に，通常のインデクスの定義と同じです。一時インデクスも一時表と同様に，指定できない，又は指定しても無視されるオペランドがあります。詳細については，マニュアル「HiRDB SQL リファレンス」の CREATE INDEX を参照してください。

(c)　データを格納する一時表用 RD エリアを指定する場合

クライアント環境定義 **PDTMPTBLRDAREA** に使用する一時表用 RD エリア名を指定します。複数の RD エリアを指定した場合や，この環境定義の指定を省略した場合，次の規則に従って HiRDB がデータを格納する一時表用 RD エリアを決定します。

- **PDTMPTBLRDAREA に複数の RD エリアを指定している場合**
  指定した RD エリアに特定 SQL セッション占有属性と SQL セッション間共有属性の一時表用 RD エリアが混在しているときは，特定 SQL セッション占有属性の一時表用 RD エリアを優先して使用します。
- **PDTMPTBLRDAREA の指定を省略している場合**
  SQL セッション間共有属性の一時表用 RD エリアを使用します。

格納先 RD エリアの決定規則については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(3)　一時表使用時の制限事項

(a)　運用コマンド又はユティリティ

次に示す運用コマンド又はユティリティは，一時表に対して実行できません。詳細は，マニュアル「HiRDB コマンドリファレンス」を参照してください。

- pdorbegin コマンド【UNIX 版限定】
- 最適化情報収集ユティリティ（pdgetcst）
- データベース作成ユティリティ（pdload）
- グローバルバッファ常駐化ユティリティ（pdpgbfon）
- 空きページ解放ユティリティ（pdreclaim）
- データベース再編成ユティリティ（pdrorg）

(b)　SQL

次に示す SQL は，一時表又は一時インデクスを指定できないなどの制限があります。詳細は，マニュアル「HiRDB SQL リファレンス」を参照してください。

- 定義系 SQL
  ALTER INDEX
  ALTER TABLE

CREATE INDEX
CREATE TABLE
CREATE TRIGGER
GRANT
REVOKE

- 操作系 SQL
  ALLOCATE CURSOR 文
  ASSIGN LIST 文
  DECLARE CURSOR 文
  動的 SELECT 文
  SELECT 文（表参照，問合せ式 形式 2）
  DELETE 文
  UPDATE 文
- 制御系 SQL
  COMMIT 文
  DISCONNECT 文
  ROLLBACK 文
  LOCK TABLE 文
  SET SESSION AUTHORIZATION 文

3.4　インデクス

表を検索するためのキーとして列に付けた索引のことを**インデクス**といいます。インデクスには，HiRDB が提供する **B-tree インデクス**及びプラグインから提供される**プラグインインデクス**があります。ここでは，B-tree インデクスについて説明します。プラグインインデクスについては，「プラグイン使用時に HiRDB で設定する項目」を参照してください。

インデクスを定義するときには，表のどの列にインデクスを定義すれば検索性能が向上するかを検討する必要があります。

3.4.1　インデクスの基本構造

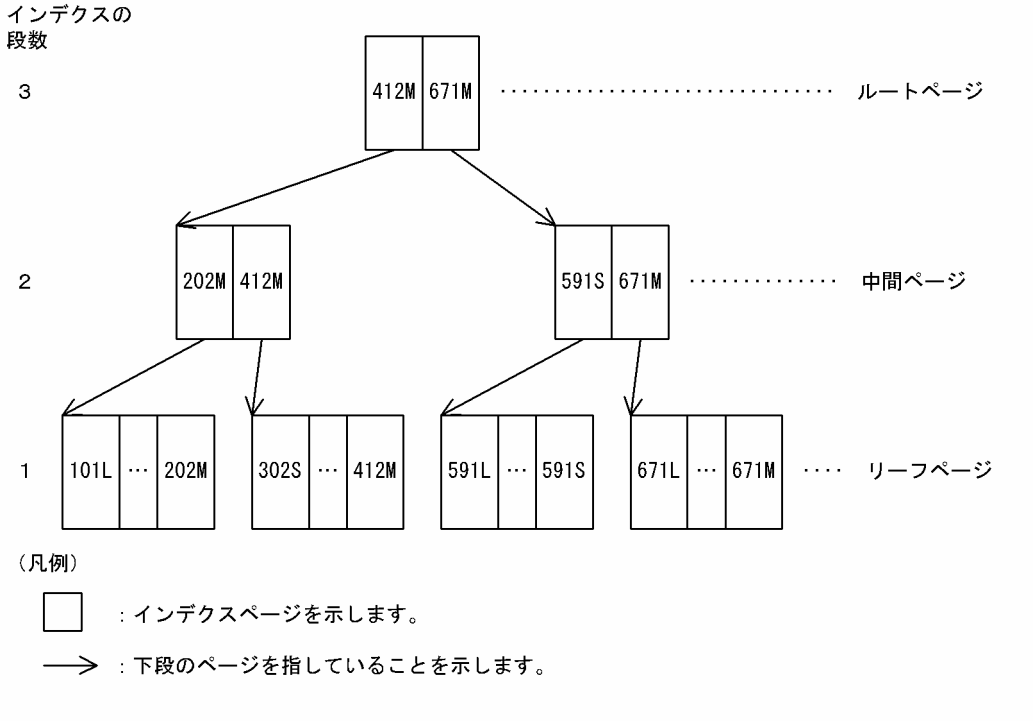
インデクスは，キーとキー値から構成されます。列の内容を示した列名のことを**キー**といいます。また，列の値のことを**キー値**といいます。表を検索するときのキーとなる列にインデクスを作成しておくと，表の検索性能が向上します。インデクスを作成する方がよい列を次に示します。

- **データを絞り込むための条件に使用する列**
- **表の結合処理の条件として使用する列**
- **データのソート又はグループ分けの条件として使用する列**
- **参照制約を定義した構成列（外部キー）**

また，HiRDB から提供されるインデクスは，**B-tree 構造**になっています。B-tree 構造中の最上位のインデクスページを**ルートページ**，最下位のインデクスページを**リーフページ**，中間のインデクスページを**中間ページ**といいます。ルートページと中間ページは下段のページを指しています。リーフページはキー値と行識別子を持っています。

インデクスの B-tree 構造を次の図に示します。

図 3–24　インデクスの B-tree 構造



〔説明〕

段数 3 で「商品コード」列にインデクスを作成した場合の，「SCODE」をキーとし，「101L～671M」をキー値とする B-tree 構造のインデクスです。

(1)　インデクスの効果

インデクスは B-tree 構造であり，キー値を昇順又は降順に格納しているため，条件を満たすキー値を効率良く検索できます。また，そのキー値に対応する行識別子を持っているため，条件を満たす行を効率良く取り出せます。

(2)　単一列インデクスと複数列インデクス

インデクスには，単一列インデクスと複数列インデクスがあります。表の一つの列で作成した一つの列のインデクスを**単一列インデクス**といいます。単一列インデクスは，一つの列をキーにして検索する場合に指定します。表の複数の列で作成した一つのインデクスを**複数列インデクス**といいます。複数列インデクスは，次に示す場合に指定してください。

- 検索するデータを複数の条件を満たすデータだけに絞り込む場合
- 探索条件でデータを絞り込んだ後，グループ分けやソートなどをする場合
- 一つの表に作成した複数の複数列インデクスが，それぞれある列で重複している場合

(3)　コストベースの最適化

ある表にインデクスが複数作成されている場合，HiRDB は表の検索で指定された探索条件を基にして，最もアクセスコストの少なく，最適と判断したインデクスを優先して選択します。このように HiRDB が最適と判断したインデクスを優先して選択する処理を**コストベースの最適化**といいます。HiRDB が判断するアクセスコストには，次に示すものがあります。

- 指定された探索条件によるヒット率
- SQL 処理に掛かる入出力処理の回数
- SQL 処理に掛かる CPU 負荷

HiRDB はコストベースの最適化をするため，表の検索性能が向上します。特に，探索条件を指定した SQL を実行する場合でも，表の検索性能を低下させません。このため，インデクスのアクセスの優先順位を意識しないで UAP を作成できます。ただし，HiRDB に最適なインデクスを使用させるには，探索条件が指定された列にインデクスが定義されていることが前提になります。

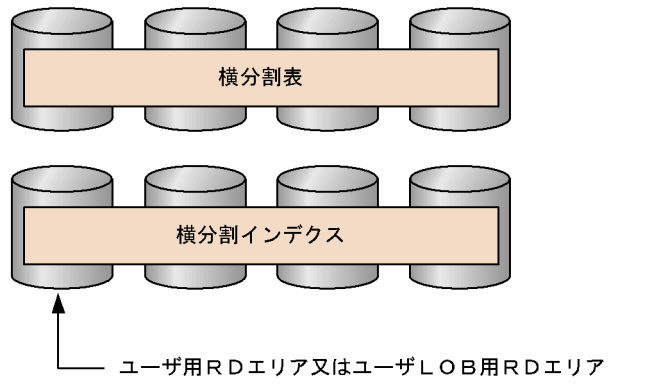
(4)　インデクスの定義

どの表のどの列をインデクスとするかは，定義系 SQL の **CREATE INDEX** で定義します。どのような列にインデクスを定義すれば検索性能が向上するか，などのインデクスの設計については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

3.4.2　インデクスの横分割

表を横分割した場合，横分割した表に対応させてインデクスも複数のユーザ用 RD エリアにわたって横分割することを**インデクスの横分割**といいます。また，横分割したインデクスを**横分割インデクス**といいます。インデクスを横分割すると，インデクスの一括作成又は再作成をするときに，ユーザ用 RD エリア又はユーザ LOB 用 RD エリアごとに独立した運用ができます。インデクスの横分割を次の図に示します。

図 3–25　インデクスの横分割



〔説明〕

横分割インデクスをどの RD エリアに格納するかは，定義系 SQL の **CREATE INDEX** で指定します。

(1)　インデクスの横分割の例（HiRDB/シングルサーバの場合）

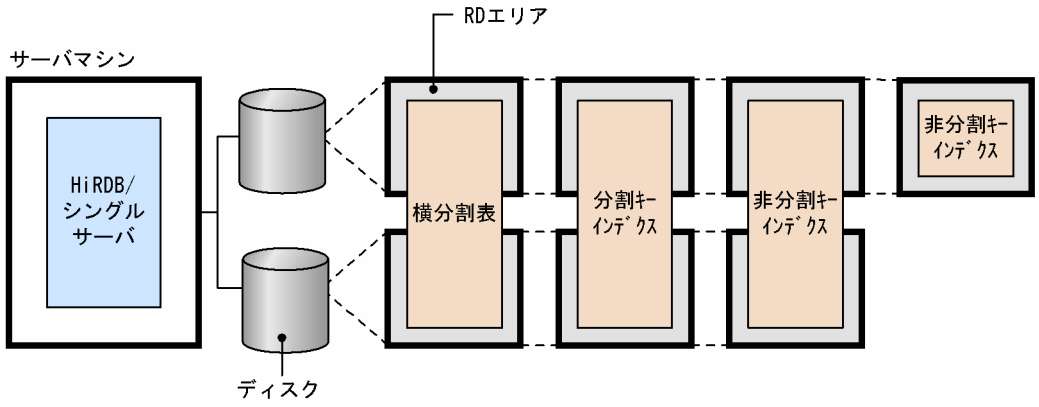
表を横分割した場合，定義するインデクスが分割キーインデクスか，非分割キーインデクスかを意識する必要があります。分割キーインデクス及び非分割キーインデクスについては，「分割キーインデクスと非分割キーインデクス」を参照してください。インデクスの種類によるインデクスの横分割指針を次の表に示します。

表 3–7　インデクスの横分割指針（HiRDB/シングルサーバの場合）

インデクスの種類	分割指針
分割キーインデクスの場合	横分割した表に対応させてインデクスも横分割してください。
非分割キーインデクスの場合	インデクスを横分割しないことをお勧めします。インデクスを横分割すると，インデクスを使用した検索性能が悪くなることがあります。 ただし，表のデータが非常に多い場合は，インデクスの横分割を検討してください。インデクスを横分割すると，表格納 RD エリアとインデクス格納 RD エリアが 1 対 1 で管理できるため，ユティリティの操作性が向上します。例えば，インデクスを横分割しない場合に RD エリア単位のデータロード，又は RD エリア単位の再編成をしたときは，データロード又は再編成後にインデクスを一括作成する必要があります。インデクスを横分割すれば，RD エリア単位のデータロード，又は RD エリア単位の再編成後にインデクスを一括作成する必要はありません。

インデクスの横分割の例（HiRDB/シングルサーバの場合）を次の図に示します。

図 3–26　インデクスの横分割の例（HiRDB/シングルサーバの場合）



〔説明〕

- ディスクのアクセス競合を避けるために，分割した表及びインデクスを格納する RD エリアを異なるディスク上に配置してください。
- 分割キーインデクスは横分割してください。
- 性能を重視する場合は，非分割キーインデクスを横分割しないでください。
- 操作性を重視する場合は，非分割キーインデクスを横分割してください。

(2)　インデクスの横分割の例（HiRDB/パラレルサーバの場合）

HiRDB/パラレルサーバの場合，表をサーバ内横分割するか，又はサーバ間横分割するかによってインデクスの横分割指針が変わります。

(a)　表をサーバ内横分割する場合

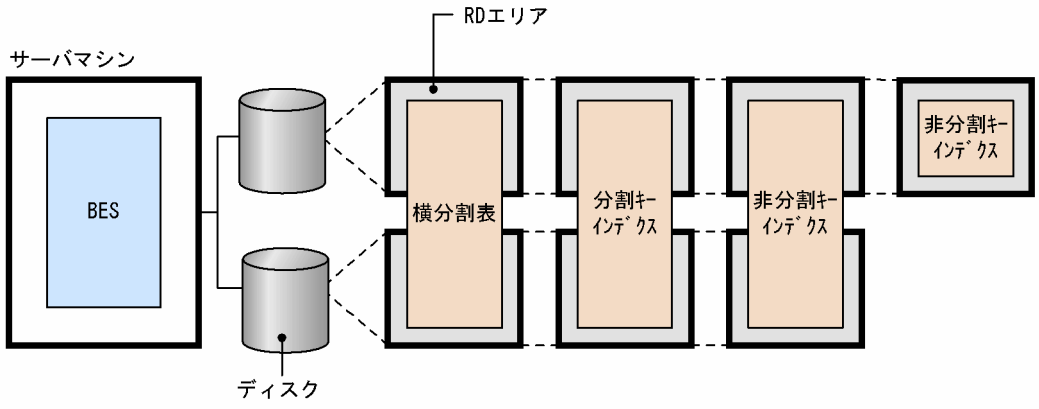
定義するインデクスが分割キーインデクスか，非分割キーインデクスかを意識する必要があります。分割キーインデクス及び非分割キーインデクスについては，「分割キーインデクスと非分割キーインデクス」を参照してください。インデクスの種類によるインデクスの横分割指針を次の表に示します。

表 3–8　インデクスの横分割指針（HiRDB/パラレルサーバの場合）

インデクスの種類	分割指針
分割キーインデクスの場合	横分割した表に対応させてインデクスも横分割してください。
非分割キーインデクスの場合	インデクスを横分割しないことをお勧めします。インデクスを横分割すると，インデクスを使用した検索性能が悪くなることがあります。 ただし，表のデータが非常に多い場合は，インデクスの横分割を検討してください。インデクスを横分割すると，表格納 RD エリアとインデクス格納 RD エリアが 1 対 1 で管理できるため，ユティリティの操作性が向上します。例えば，インデクスを横分割しない場合に RD エリア単位のデータロード，又は RD エリア単位の再編成をしたときは，データロード又は再編成後にインデクスを一括作成する必要があります。インデクスを横分割すれば，RD エリア単位のデータロード，又は RD エリア単位の再編成後にインデクスを一括作成する必要はありません。

インデクスの横分割の例（サーバ内横分割の場合）を次の図に示します。

図 3–27　インデクスの横分割の例（サーバ内横分割の場合）



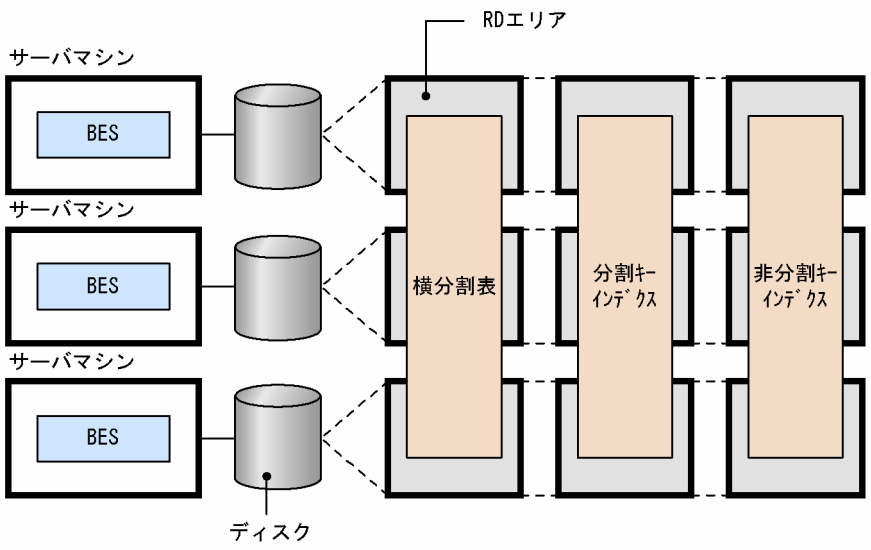
〔説明〕

- ディスクのアクセス競合を避けるために，分割した表及びインデクスを格納する RD エリアを異なるディスク上に配置してください。
- 分割キーインデクスは横分割してください。
- 性能を重視する場合は，非分割キーインデクスを横分割しないでください。
- 操作性を重視する場合は，非分割キーインデクスを横分割してください。

(b)　表をサーバ間横分割する場合

横分割した表に対応させてインデクスも横分割してください。定義するインデクスが分割キーインデクスか，非分割キーインデクスかを意識する必要はありません。インデクスの横分割の例（サーバ間横分割の場合）を次の図に示します。

図 3–28　インデクスの横分割の例（サーバ間横分割の場合）



〔説明〕

- ディスクのアクセス競合を避けるために，分割した表及びインデクスを格納する RD エリアを異なるディスク上に配置してください。
- 分割キーインデクス及び非分割キーインデクスを横分割してください。

(3)　分割キーインデクスと非分割キーインデクス

インデクスがある一定の条件を満たすと，そのインデクスは**分割キーインデクス**になります。条件を満たさないインデクスは**非分割キーインデクス**になります。ここでは，その条件について説明します。この条件は，表が単一列分割か複数列分割かによって異なります。表の分割条件に一つの列だけを使用している場合を**単一列分割**といい，表の分割条件に複数の列を使用している場合を**複数列分割**といいます。

(a)　単一列分割の場合

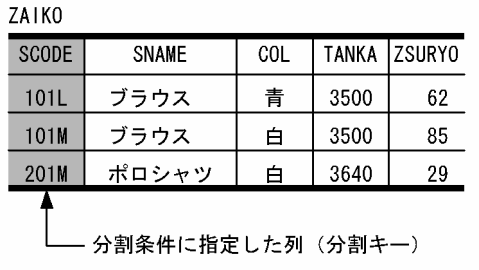
次に示すどちらかの条件を満たす場合，そのインデクスは分割キーインデクスになります。

〈条件〉

- 表を横分割するときに格納条件を指定した列（分割キー）に定義した単一列インデクス
- 表を横分割するときに格納条件を指定した列（分割キー）を第 1 構成列とした複数列インデクス

ZAIKO 表を例にして，インデクスが分割キーインデクスになる場合を次の図に示します。

図 3–29　分割キーインデクスになる場合（単一列分割の場合）



〔説明〕

```
CREATE INDEX A12 ON ZAIKO (SCODE ASC)                  1
CREATE INDEX A12 ON ZAIKO (SCODE ASC,TANKA DESC)       2
CREATE INDEX A12 ON ZAIKO (TANKA DESC,SCODE ASC)       3
```

1. 分割キーである SCODE 列をインデクスとした場合，そのインデクスは**分割キーインデクス**になります。そのほかの列をインデクスとした場合，そのインデクスは非分割キーインデクスになります。
2. 分割キーである SCODE 列を複数列インデクスの第 1 構成列にすると，その複数列インデクスは**分割キーインデクス**になります。
3. 分割キーである SCODE 列を第 1 構成列以外に指定すると，その複数列インデクスは**非分割キーインデクス**になります。

(b)　複数列分割の場合

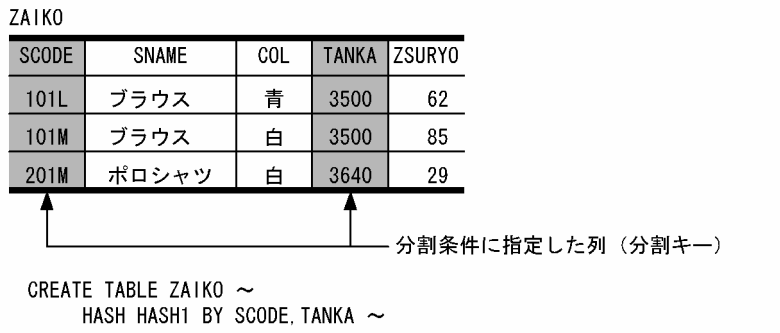
次に示す条件を満たす場合，そのインデクスは分割キーインデクスになります。

〈条件〉

- 分割キーを先頭とし，分割に指定した列を先頭から同順にすべて含んで，複数の列に作成したインデクスです。

ZAIKO 表を例にして，インデクスが分割キーインデクスになる場合を次の図に示します。

図 3–30　分割キーインデクスになる場合（複数列分割の場合）



〔説明〕

```
CREATE INDEX A12 ON ZAIKO (SCODE ASC,TANKA DESC)              1
CREATE INDEX A12 ON ZAIKO (SCODE ASC,TANKA DESC,ZSURYO ASC)   2
```

```
CREATE INDEX A12 ON ZAIKO (TANKA DESC,SCODE ASC)              3
CREATE INDEX A12 ON ZAIKO (SCODE ASC,ZSURYO DESC,TANKA ASC)   4
```

1. すべての分割キー（SCODE 及び TANKA 列）を指定し，かつ分割キーの指定順序が表定義時と同じため，この複数列インデクスは**分割キーインデクス**になります。
2. すべての分割キー（SCODE 及び TANKA 列）を指定し，かつ分割キーの指定順序が表定義時と同じため，この複数列インデクスは**分割キーインデクス**になります。
3. すべての分割キー（SCODE 及び TANKA 列）を指定しているが，分割キーの指定順序が表定義時と異なるため，この複数列インデクスは**非分割キーインデクス**になります。
4. すべての分割キー（SCODE 及び TANKA 列）を指定しているが，分割キーの指定順序が表定義時と異なるため，この複数列インデクスは**非分割キーインデクス**になります。

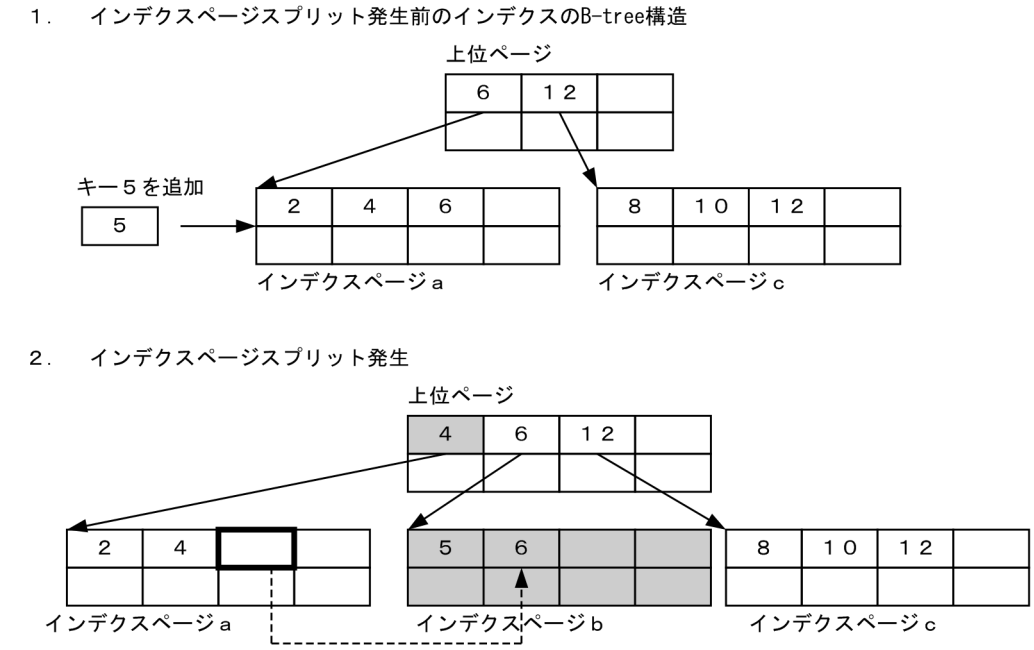
3.4.3　インデクスページスプリット

空き領域がないインデクスページにキーが追加されると，HiRDB はインデクスページの空き領域を確保します。空き領域の確保方法はインデクスページのインデクス情報を分割して，後の半分を新しいページに移します。この処理を**インデクスページスプリット**といいます。インデクスページスプリットには，インデクス情報を均等に分割する方法と不均等に分割する方法（アンバランスインデクススプリット）があります。

(1)　均等に分割するインデクスページスプリット

均等に分割するインデクスページスプリットの例を次の図に示します。

図 3–31　均等に分割するインデクスページスプリットの例



〔説明〕

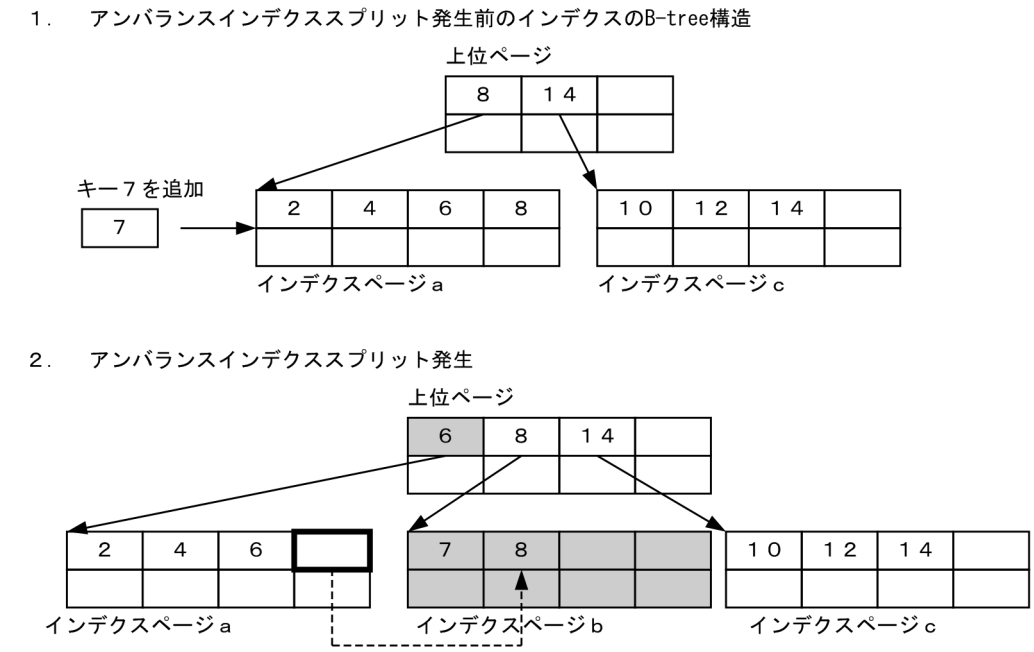
インデクスページ a にキー 5 が追加されたため，インデクスページスプリットが発生し，キー 5 とその後の 6 が新しいインデクスページ b に移り，均等に分割されています。

(2)　アンバランスインデクススプリット

連続したキー値を追加すると，インデクスページのデータ格納効率が下がります。このような場合，インデクスページのインデクス情報を均等に分割しないで，不均等に分割できます。これを**アンバランスインデクススプリット**といいます。

アンバランスインデクススプリットの場合，インデクスページのインデクス情報を分割する比率をキー値の追加位置で決定します。追加位置がインデクスページから見て前半部分であれば，以降，前半部分にキーが追加されると予測します。このため，追加するキー値より一つ大きいキー値を分割位置として，前半部分を左側ページに格納します。追加位置が後半部分であれば，以降，後半部分にキーが追加されると予測します。このため，追加位置を分割位置として，後半部分を右側ページに格納します。インデクスページの後半部分にキーを追加した場合のアンバランスインデクススプリットの例を次の図に示します。

図 3–32　アンバランスインデクススプリットの例



〔説明〕

インデクスページ a にキー 7 が追加され，追加位置が後半であるため，キー 7 とそれより後のキー 8 が，右側のインデクスページ b に移っています。

適用基準

次に示すような場合にアンバランスインデクススプリットをすると，データの格納効率を向上できます。また，インデクスページスプリットの発生回数の削減ができます。

- キーの重複度に偏りがないインデクス
- キー長がほぼ均一であるインデクス
- 連続した中間キー値を頻繁に追加するインデクス

アンバランスインデクススプリットの指定方法

アンバランスインデクススプリットをするには，定義系 SQL の **CREATE INDEX** 又は **CREATE TABLE** で **UNBALANCED SPLIT** オプションを指定します。アンバランスインデクススプリットについては，マニュアル「HiRDB システム運用ガイド」を参照してください。

3.4.4　除外キー値

インデクスを定義した列のデータは，ナル値であってもすべてインデクスのキー値としてインデクス中に取り込まれます。インデクス中にナル値のキー値があっても意味がないため，むだなデータとなってしまいます。このような場合，すべての構成列の値がナル値で，キー値の重複が多いインデクスに対してナル値を**除外キー値**として指定できます。インデクスに除外キー値を設定すると，次に示す効果が期待できます。

期待できる効果

1. インデクスには，ナル値のキーを作成しないため，インデクスの容量を削減できます。
2. 行の挿入，削除及び更新時のインデクスメンテナンスのオーバヘッド（CPU 時間，入出力回数，排他制御要求回数及びデッドロック発生頻度）とログ量を削減できます。
3. ナル値を除外キー値とするインデクスの構成列に対する探索条件が，「IS NULL」だけの検索ではインデクスを使用しません。これによって，次に示す場合に検索性能が向上します。

   ・ナル値の重複が多い状態でインデクスを使用し，データページをランダムにアクセスしたため，同じページに入出力処理が発生していた場合

除外キー値の指定方法

定義系 SQL の **CREATE INDEX** で **EXCEPT VALUES** オプションを指定します。インデクスの除外キー値の詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

3.4.5　データを格納している表にインデクスを定義する場合

データが大量にある表に対してインデクスを定義する場合，インデクスの実体の作成（CREATE INDEX の実行）に時間が掛かります。その間，ほかの定義系 SQL は実行できません。

CREATE INDEX に **EMPTY** オプションを指定すると，インデクスの実体を作成しないで，定義上のインデクスを作成します。これを**未完状態のインデクス**といいます。インデクスの実体を作成しないため，CREATE　INDEX の実行は即時終了し，ほかの定義系 SQL を実行できるようになります。

インデクスの実体が未作成であるため，未完状態のインデクスを使った検索や未完状態のインデクスを定義している表の列の更新はできません（SQL エラーとなります）。インデクスの実体は，データベース再編成ユティリティ（pdrorg）のインデクス再作成機能（-k ixrc）を使用して作成します。インデクスの実体を作成すると，実体を作成したインデクスの未完状態は解除されます。また，PURGE TABLE 文で表を全件削除すると，その表のすべてのインデクスの未完状態が解除されます。

EMPTY オプションの運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

3.4.6　インデクスキー値無排他

インデクスキー値に排他を掛けないで，表のデータだけに排他を掛けて表にアクセスすることを**インデクスキー値無排他**といいます。

インデクスキー値無排他を適用すると，次に示す問題を回避できます。

- データ更新とインデクス検索との間のデッドロック
- 同一キーを持つデータに対するアクセスが必要もなく待たされる
- 異なるキーを持つデータに対するアクセスが必要もなく待たされる

インデクスキー値無排他を使用すると，インデクスを利用した検索処理ではインデクスキー値に排他を掛けません。また，表に対する更新処理（行挿入，行削除及び列値更新）の場合にも，更新対象列に定義されているインデクスのインデクスキー値に対して排他は掛けません。

(1)　適用基準

どの業務でも，インデクスキー値無排他を使用することをお勧めします。ただし，ユニークインデクスの動作，残存エントリ及びインデクスログ量を考慮した上で決めてください。ユニークインデクスの一意性制約保証処理の動作，残存エントリについては，「注意事項」を参照してください。インデクスキー値無排他を使用するときのインデクスログ量については，マニュアル「HiRDB システム導入・設計ガイド」のインデクスログ量と排他資源数の見積もりの説明を参照してください。

(2)　指定方法

インデクスキー値無排他を使用する場合は，システム共通定義の **pd_indexlock_mode** オペランドに NONE を指定します。

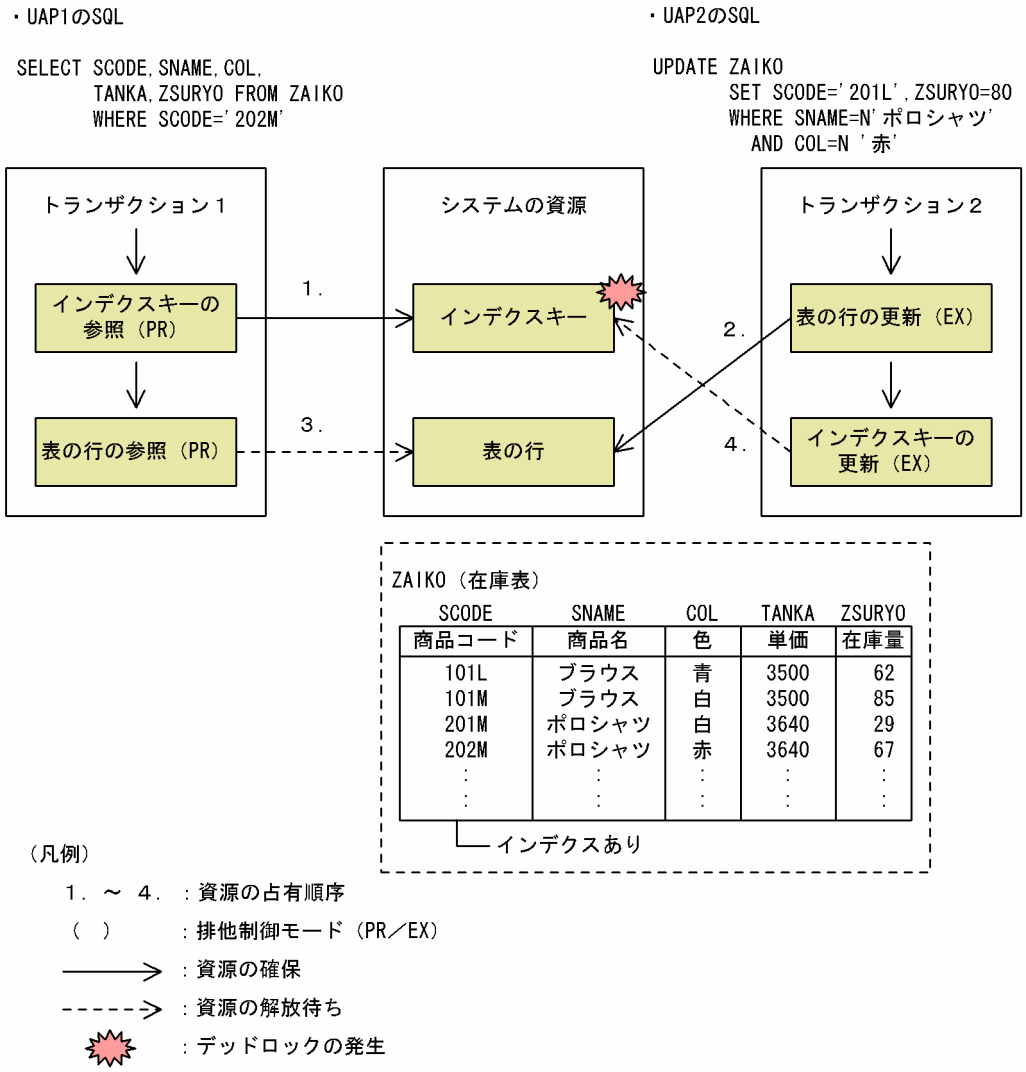
【UNIX 版の場合の留意事項】

システム定義の pd_inner_replica_control オペランドの指定値が 1 より大きいとき，システム定義の pd_indexlock_mode オペランドの指定に関係なく，pd_indexlock_mode オペランドには NONE が仮定されます。

(3)　インデクスキー値無排他を使用した場合のデッドロック回避の例

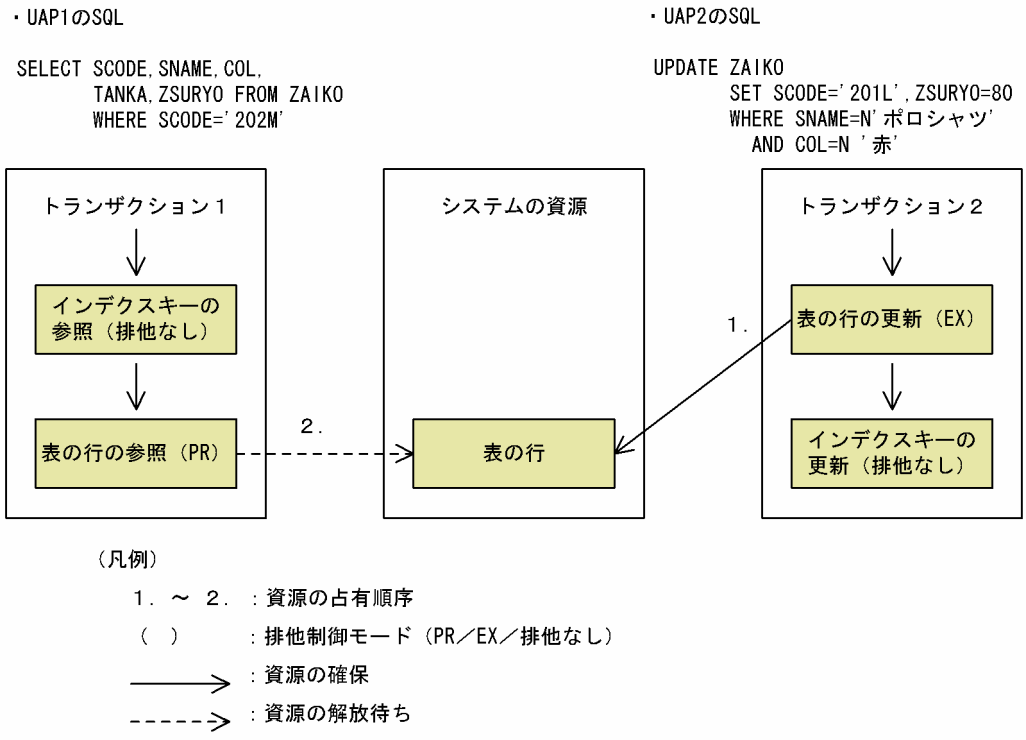
インデクスキー値無排他を使用した場合のデッドロック回避の例について説明します。ここで説明するデッドロックの例を次の図に示します。排他制御モードについては，「排他制御モード」を参照してください。

図 3-33　デッドロックの例（インデクスキー値無排他を使用しない場合）



インデクスキー値無排他を指定しておくと，図「デッドロックの例（インデクスキー値無排他を使用しない場合）」のようなデッドロックを回避できます。インデクスキー値無排他を使用した場合のデッドロック回避の例を次の図に示します。

図 3–34　インデクスキー値無排他を使用した場合のデッドロック回避の例



(4)　注意事項

(a)　ユニークインデクスの一意性制約保証処理の動作

一意性制約定義が指定されている表では，インデクスキー値無排他の場合とそうでない場合とで，行の追加及び更新時に行われる一意性制約の保証処理の動作が異なります。一意性制約の保証処理とは，行データの挿入，又は列値更新のときに，インデクス（ユニークインデクス）を使用して追加しようとしているキーを持つデータが既に表にあるかをチェックするとともに，排他制御で追加キーの一意性を保証する処理のことをいいます。一意性制約の保証処理では，同一キーを持つインデクスキーエントリが見付かった場合，エラーとなります。そのインデクスキーを操作している相手トランザクションが未決着状態で，ロールバックする可能性があったとしても，排他制御でのチェックをしないでエラーとなります。

一意性制約を指定した表データの挿入，又は更新処理をする場合に，待つことよりも処理を続行することを優先したいときには，インデクスキー値無排他を適用してください。また，待ってでも挿入，又は更新処理を試みる方を優先したい場合には，インデクスキー値無排他を適用しないでください。

(b)　ユニークインデクスの残存エントリ

インデクスキー値無排他を適用する場合，ユニークインデクスで排他待ち，及びデッドロックが発生することがあります。インデクスキー値無排他でのユニークインデクスでは，一意性制約保証のために DELETE 文，又は UPDATE 文実行前のインデクスキーをインデクス上から削除しないで残すようにしています。この残っているインデクスキーのことを残存エントリといいます。この残存エントリは，トランザクション決着後の適当なタイミングで削除されますが，残存エントリと同一のキーに対する INSERT 文，又は UPDATE

文を実行した場合，タイミングによっては予想外に待たされたり，デッドロックが発生したりすることがあります。

これらを回避するためには，一意性制約の列を更新しないように UAP を作成する必要があります。

(5)　インデクスキー値無排他を適用しても回避できないデッドロック

UAP のアクセス順序によって，インデクスキーとインデクスキーとでデッドロックが発生することがあります。これを回避するためには，一意性制約の列を更新しないように UAP を作成しなければなりません。

3.5　オブジェクトリレーショナルデータベースへの拡張

リレーショナルデータモデルにオブジェクト指向の概念を取り込んで拡張した，**オブジェクトリレーショナルデータベース管理システム**を構築できます。

HiRDB では，マルチメディアデータなどの複雑な構造を持つデータと，そのデータに対する操作を一体化してオブジェクトとして扱え，データベースで管理できます。これによって，従来のリレーショナルデータベースと同様の SQL 文を使用してマルチメディアデータを操作できます。例えば，次に示すマルチメディアデータを管理及び操作できます。

- **SGML/XML 構造化文書データ**
  構造指定による全文検索，検索ヒット位置のハイライト表示などの操作ができます。
- **空間データ**
  地図情報などの空間データ（2 次元データ）を検索できます。

SGML/XML 構造化文書データ，空間データの操作機能はプラグインによって提供されています。プラグインの利用方法については，「プラグイン」を参照してください。

3.5.1　抽象データ型

ユーザ定義型である**抽象データ型**とルーチンを使用すると，複雑な構造を持つデータとその操作を独自に定義／利用できます。抽象データ型として列を定義すると，オブジェクト指向に基づく概念化・モデル化ができるようになります。また，オブジェクト指向に基づくソフトウェア開発手法を適用して，データベースの設計，UAP の開発及びメンテナンスの負担を軽減できます。

HiRDB では，独自の抽象データ型やその構造を定義系 SQL を使用して定義できます。抽象データ型は，数値型や文字型などの HiRDB の既定義型と同様に，表を構成する列のデータ型として扱えます。抽象データ型の値に対する操作も，定義系 SQL を使用してルーチンとして定義できます。UAP では，ルーチンを使用して抽象データ型に対する複雑な操作を SQL で記述できます。

抽象データ型とルーチン及びそれらについての特徴的な概念について，例を使用して説明します。

(1)　抽象データ型の定義

抽象データ型を使用して，従業員についての情報をデータベースで管理及び操作する例を示します。

従業員についての情報は，氏名，性別，入社年月日，役職，基本給などの情報から構成され，顔写真のような画像情報も従業員についての情報を構成する情報とします。また，従業員についての情報の操作として，勤続年数を算出するものとします。

この情報をデータベースで取り扱う場合の，データモデル化した抽象的な概念としての「従業員」は，その概念に共通した特性を示す属性「氏名」，「性別」，「入社年月日」，「役職」，「顔写真」，「基本給」から構

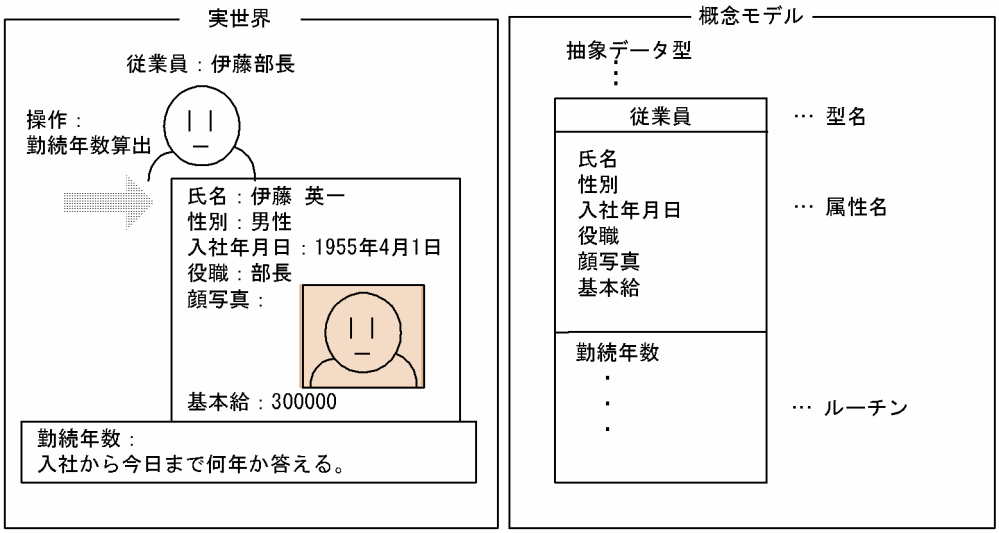
成されると考えられます。また，「従業員」に対する操作「勤続年数」算出などについても，「従業員」の特性を示す操作であると考えられます。

「従業員」は，これらの特性（属性と操作）を一つのまとまりとみなします。

HiRDB では，データベースで取り扱う実世界にある対象を抽象化した概念でとらえ，その概念を抽象データ型を使ってデータ型の一つにできます。

実世界の情報と抽象データ型による概念モデルを次の図に示します。

図 3-35　実世界の情報と抽象データ型による概念モデル



抽象データ型は，次に示す定義系 SQL の CREATE TYPE によってデータベースに定義できます。

```
CREATE TYPE t_従業員 (
      氏名         CHAR(16),
      性別         CHAR(1),
      入社年月日    DATE,
      役職         CHAR(10),
      顔写真       BLOB(64K),
      基本給       INTEGER,
      …
      FUNCTION  勤続年数 ( p t_従業員 )
         RETURNS INTEGER
       BEGIN
        DECLARE working_years INTERVAL YEAR TO DAY;
        SET working_years = CURRENT_DATE - p..入社年月日;
        RETURN YEAR(working_years);
       END,
      …
)
```

このように抽象データ型を定義すると，更にユーザがその抽象データ型の属性及び操作を指定して，新たなデータ型を定義できます。なお，操作はルーチンに定義できます。

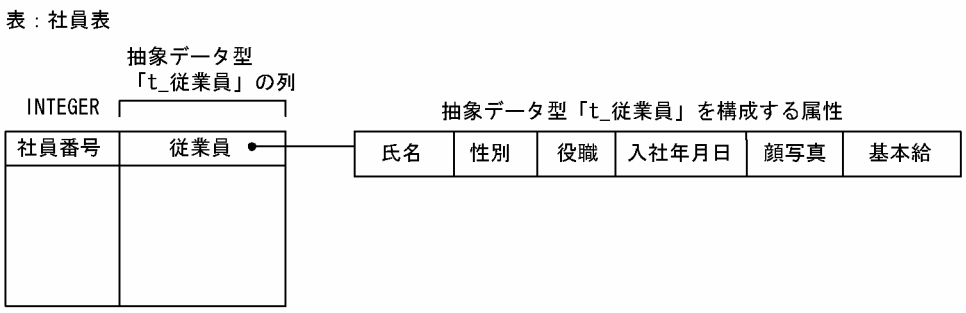
(2)　データ型としての抽象データ型

抽象データ型は，数値型や文字型などの HiRDB の既定義型と同様に扱えます。例えば，次に示す定義系 SQL によって，抽象データ型 **t_従業員** を列のデータ型として，表 **社員表** を定義できます。

```
CREATE TABLE 社員表 (
     社員番号       INTEGER,
     従業員          t_従業員   ALLOCATE (顔写真 IN (lobarea) )
     )
```

抽象データ型の属性の中で，BLOB 型の属性がある場合には，そのデータを格納するユーザ LOB 用 RD エリアを **CREATE TABLE** の **ALLOCATE** で指定します。上記の例では，**t_従業員**の属性である**顔写真**が BLOB 型であるため，ALLOCATE を使用してユーザ LOB 用 RD エリア lobarea に格納しています。抽象データ型を定義した社員表を次の図に示します。

図 3–36　抽象データ型を定義した社員表



(3)　カプセル化

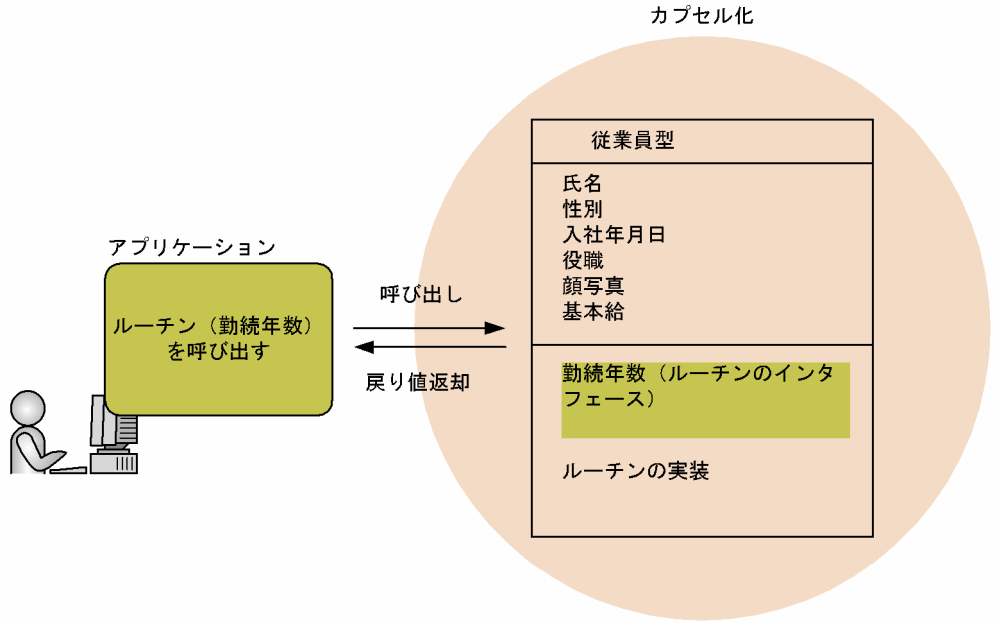
抽象データ型を使用した場合，アプリケーションでは，その抽象データ型に宣言されたルーチンを使用することで，個々の属性の詳細な構成やルーチンの実装を知らなくても，抽象データ型の値を扱えるようになります。例えば，次に示す操作系 SQL で，t_従業員型の値を操作できます。

```
SELECT 社員番号, 従業員..氏名, 勤続年数(従業員)
FROM  社員表
```

このように，抽象データ型によって，値の内部情報を意識させないようにして，外部的なインタフェースだけで値を扱えるようにすることを**カプセル化**といいます。

カプセル化の概要を次の図に示します。

図 3–37　カプセル化の概要



(4)　抽象データ型の値

(a)　値の生成

抽象データ型と同じ名前で識別される引数のない関数を実行して，その抽象データ型の値を生成できます。例えば，t_従業員型 の場合には，関数 t_従業員() で，t_従業員型 の値を生成できます。このように，抽象データ型の値を生成する関数を**コンストラクタ関数**といいます。

```
BEGIN                        … SQL手続き文の始まり
  DECLARE p t_従業員;        … t_従業員型の変数宣言
  SET p = t_従業員();        … t_従業員型の値を生成し，変数に代入
  SET p..氏名 = 'イトウエイイチ' … コンポネント指定による属性値の設定
  RETURN p;                  … 関数の戻り値の返却（t_従業員型の値を返す）
END                          … SQL手続き文の終わり
```

CREATE TYPE で抽象データ型をデータベースに定義すると，t_従業員()のような，データ型名が同じで引数のない関数が自動的に定義されます。このような関数を特に**デフォルトコンストラクタ関数**といいます。

(b)　ユーザ定義のコンストラクタ関数

コンストラクタ関数をユーザが定義することもできます。**CREATE TYPE** のルーチン宣言で，定義する抽象データ型と同じ名前でその抽象データ型を戻り値の型とする関数を定義すると，コンストラクタ関数を定義できます。

```
CREATE TYPE t_従業員 (
      氏名        CHAR(16),
      性別        CHAR(1),
      入社年月日  DATE,
      役職        CHAR(10),
      顔写真      BLOB(64K),
```

```
        基本給      INTEGER,

  FUNCTION t_従業員(
        p_氏名      CHAR(16),
        p_性別      CHAR(1),
        p_入社年月日   DATE,
        p_役職        CHAR(10),
        p_顔写真      BLOB(64K),
        p_基本給      INTEGER)
        RETURNS         t_従業員
  BEGIN
        DECLARE    d_従業員 t_従業員;
          SET      d_従業員 = t_従業員();
          SET      d_従業員..氏名       = p_氏名;
          SET      d_従業員..性別       = p_性別;
          SET      d_従業員..入社年月日 = p_入社年月日;
          SET      d_従業員..役職       = p_役職;
          SET      d_従業員..顔写真     = p_顔写真;
          SET      d_従業員..基本給     = p_基本給;

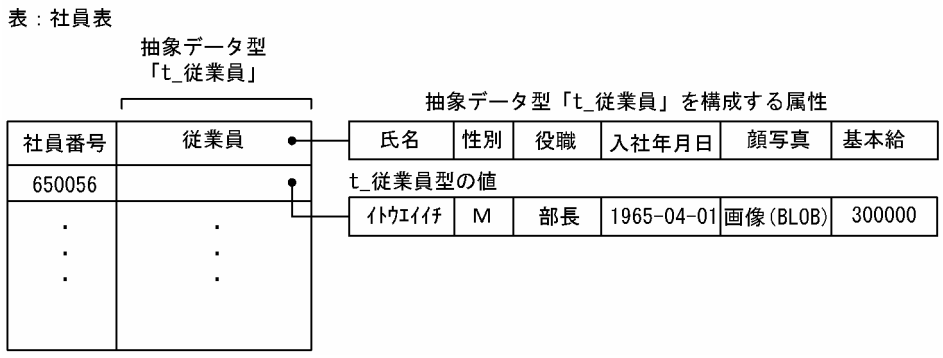
        RETURN d_従業員;
  END,
  ...
  )
```

例えば，ユーザ定義のコンストラクタ関数 t_従業員() を使用して，次に示す操作系 SQL で値を生成し，データベースに格納できます。

```
INSERT INTO 社員表
  VALUES ( 650056,
        t_従業員(:name AS CHAR(16), :sex AS CHAR(1), :yrs AS DATE,
              :post AS CHAR(10), :picture AS BLOB(64K) ,:salary AS INTEGER)
          )
```

コンストラクタ関数 **t_従業員()** によって値を生成し，列値として挿入した表 **社員表**を次の図に示します。

図 3-38　コンストラクタ関数によって値を生成した表 社員表



(5)　抽象データ型のナル値

HiRDB の既定義型と同様に，抽象データ型にもナル値を適用できます。例えば，前述の**社員表**について次に示す操作系 SQL を実行した場合，列 **従業員**の値はナル値になります。

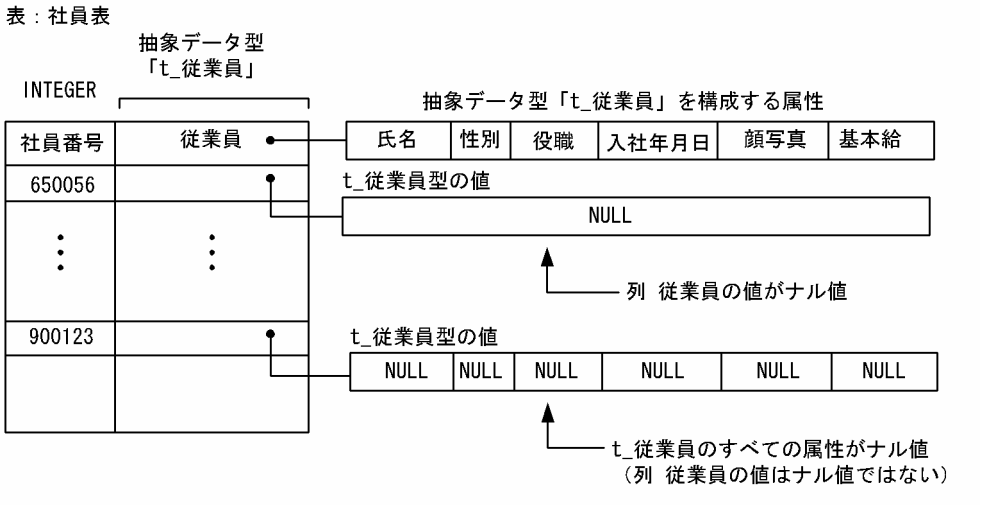
```
INSERT INTO 社員表(社員番号) VALUES ( 650056 )
```

一方，次に示す操作系 SQL を実行すると，**t_従業員型**の全属性の値がナル値になります。なお，すべての属性の値がナル値であるような抽象データ型の値は，ナル値ではない値とみなされます。

```
INSERT INTO 社員表 (900123, t_従業員())
```

抽象データ型を定義した社員表でのナル値の扱いを次の図に示します。

図 3–39　抽象データ型を定義した社員表でのナル値の扱い



例えば，次に示す操作系 SQL を実行すると，列 従業員の値がナル値ではない従業員の社員番号が検索されるため，t_従業員型のすべての属性がナル値である従業員の社員番号は検索されます。

```
SELECT 社員番号 FROM 社員表
   WHERE 従業員 IS NOT NULL
```

検索結果

```
900123
```

(6)　抽象データ型の値の操作

従業員に対して，勤続年数を算出するという操作を考えます。

HiRDB では，**CREATE TYPE** のルーチン宣言で，抽象データ型の値に対する操作を定義できます。例えば，「勤続年数」算出や「報酬率」算出の操作を次に示す定義系 SQL で定義できます。

```
CREATE TYPE t_従業員 (
      氏名        CHAR(16),
      性別        CHAR(1),
      入社年月日  DATE,
      役職        CHAR(10),
      顔写真      BLOB(64K),
```

```
        基本給       INTEGER,
      …
        FUNCTION  勤続年数 ( p t_従業員 )
           RETURNS INTEGER
         BEGIN
          DECLARE working_years INTERVAL YEAR TO DAY;
           SET working_years = CURRENT_DATE - p..入社年月日;
          RETURN YEAR(working_years);
         END,
       …
)
```

このように，抽象データ型で定義したルーチンをその抽象データ型の値に対して使用できます。例えば，**勤続年数**が 10 年以上の**社員**の**社員番号**と**氏名**を検索する SQL は，次のように記述できます。

```
SELECT 社員番号, 従業員..氏名, 勤続年数(従業員)
 FROM  社員表
 WHERE 勤続年数(従業員) >= 10
```

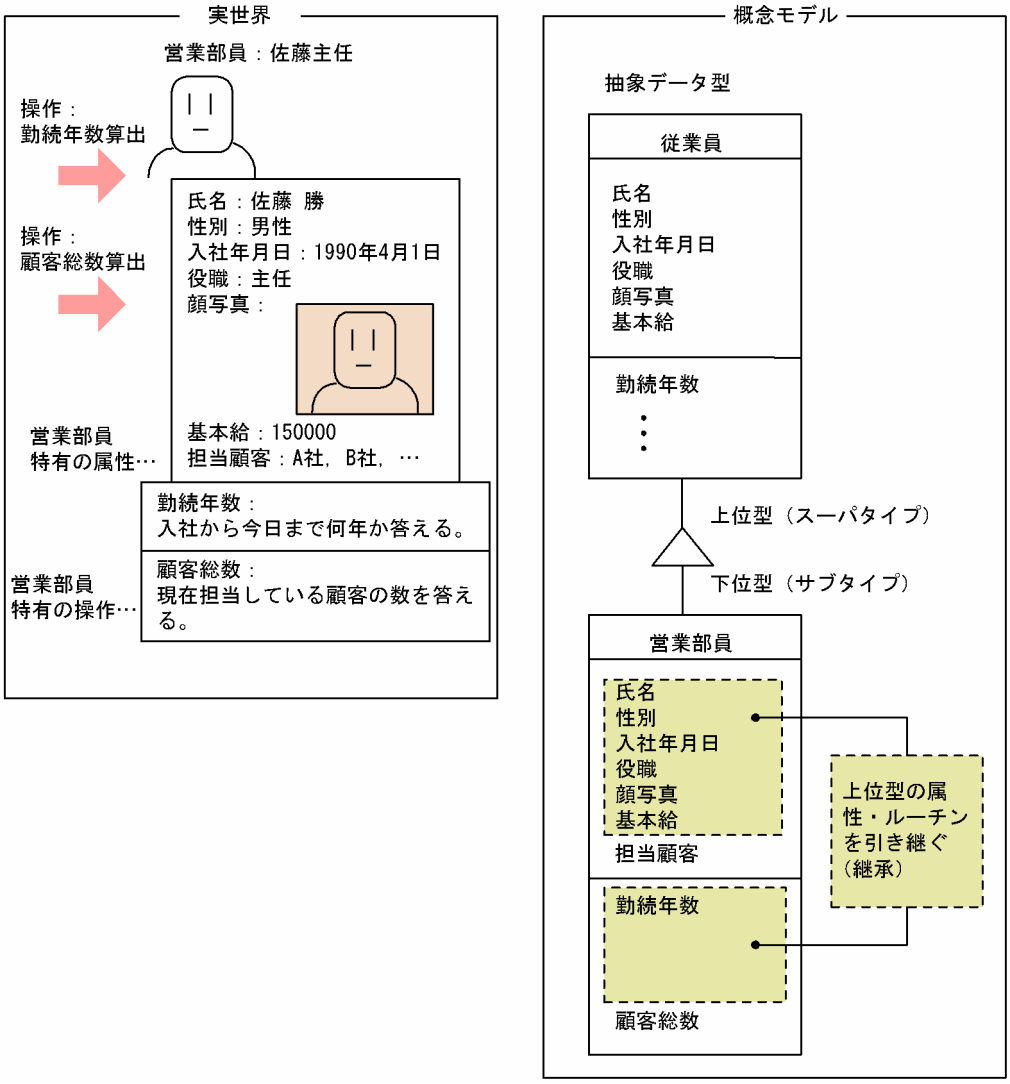
3.5.2　サブタイプと継承

(1)　サブタイプ（subtyping）

従業員のうち，特に営業部員の情報について管理することを考えます。「営業部員」は，「従業員」という抽象的な概念に対して，より具体的で特殊な（特化した）概念であると考えられます。営業部員についての情報は，従業員が持つ情報に加えて，営業活動についての情報があると考えられます。そこで，「営業部員」は，「従業員」であることに加えて，属性「担当顧客」や操作「顧客総数」などを持つことにします。

営業部員について，実世界の情報と抽象データ型による概念モデルを次の図に示します。

図 3–40　実世界の情報と抽象データ型による概念モデル（営業部員の場合）



HiRDB では，あるデータ型を基にその型を特化した抽象データ型を**サブタイプ**として定義できます。

例えば，定義系 SQL 文 **CREATE TYPE** のサブタイプ句を使用して，t_従業員を基に，t_営業部員を次に示すように定義できます。

```
CREATE TYPE t_営業部員 UNDER t_従業員(
    担当顧客  VARCHAR(3000),

    FUNCTION 顧客総数 ( …    )
    RETURNS INTEGER
      …
    )
```

なお，t_従業員のような，サブタイプに対する上位の抽象データ型を**スーパタイプ**といいます。

(2) 代替可能性（substitutability）

「営業部員」は「従業員」でもあります。HiRDB では，サブタイプで特化した（下位の）抽象データ型の値をその上位の抽象データ型の値としても扱えます。例えば，**社員表**では次に示す SQL で，**t_営業部員**の値を **t_従業員**の値として列の値にできます。

注意事項

t_営業部員の値も t_従業員の値と同様に扱えるようにするために，この SQL を実行する前に，ALTER ROUTINE を実行し，SQL オブジェクトを再作成しておく必要があります。

```
INSERT INTO 社員表 VALUES（ 51, t_営業部員(:name AS CHAR(16),…））
```

このように，下位の抽象データ型の値が上位の抽象データ型の値としてみなされることを**代替可能性**（substitutability）といいます。

(3) 継承（inheritance）

「営業部員」は「従業員」でもあるので，「従業員」と同様に「氏名」や「性別」の属性を持ち，「勤続年数」算出の操作が適用できると考えられます。

図「実世界の情報と抽象データ型による概念モデル（営業部員の場合）」のように，HiRDB ではサブタイプで特化した（下位の）抽象データ型にその上位の抽象データ型に定義された属性とルーチンが引き継がれます。

このように，下位の抽象データ型が，上位の抽象データ型の属性及びルーチンを引き継ぐことを**継承**（inheritance）といいます。

例えば，継承によって **t_営業部員**の値に対して属性「氏名」と操作「勤続年数」が利用できるようになるため，**t_営業部員**の値を列値に挿入した**社員表**について，次に示す SQL のように，先に示した SQL を変更することなく実行できます。

注意事項

この SQL を実行する前に，ALTER ROUTINE を実行し，SQL オブジェクトを再作成しておく必要があります。

```
SELECT 社員番号, 従業員..氏名, 勤続年数(従業員)
  FROM  社員表
  WHERE 勤続年数(従業員）>= 10
```

サブタイプと継承で，次に示す効果が期待できます。

- 「営業部員」が「従業員」であるという代替可能性の概念を簡潔に表現できます。
- 属性及びルーチンの定義を共有し，既存の定義を基に新しい定義を追加できます。そのため，データベース及びアプリケーション開発のオーバヘッドを削減でき，拡張性のあるシステムを構築できます。

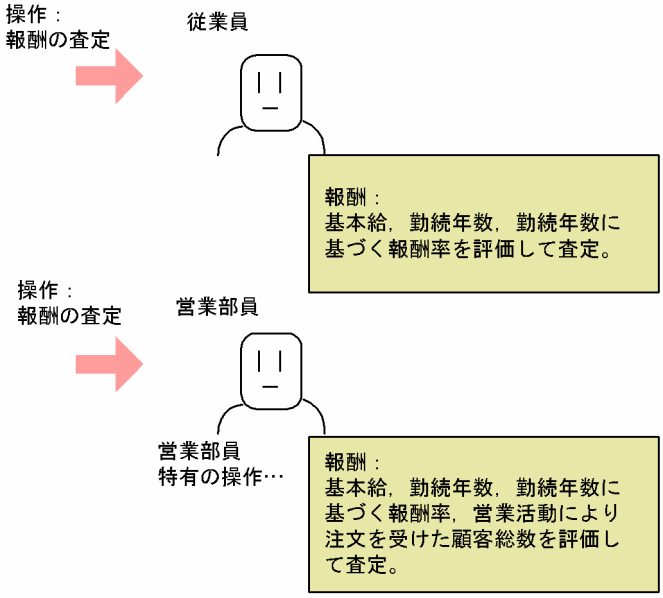
(4)　多重定義（override）

「従業員」に対して「報酬」を査定する操作を考えます。報酬の査定は，例えば，「基本給」，「勤続年数」，「勤続年数に基づく報酬率」によって設定された情報を基に，作業状況を評価して査定することが考えられます。

一方で，継承によって「従業員」に定義される操作「報酬」は自動的に「営業部員」に引き継がれます。しかし，特に営業部員については，一般的な従業員の査定方法とは異なり，営業活動で注文を受けた「顧客総数」などを基に，営業部員特有の査定をすることが考えられます。

実世界での従業員，営業部員についての操作を次の図に示します。

図 3–41　実世界での従業員，営業部員についての操作



例えば，「従業員報酬」や「営業部員報酬」のような，それぞれの抽象データ型ごとに異なる名称のルーチンを定義するとします。その場合，代替可能性で「従業員」の値と「営業部員」の値を統一的に扱えるのにもかかわらず，それぞれの値の型によって呼び出すルーチンの名称を変更しなければならなくなります。そのため，アプリケーションでは次に示す操作系 SQL を実行できなくなります。

```
×　SELECT　社員番号，従業員..氏名，従業員報酬(従業員)
　　FROM　　社員表
```

…「営業部員」に対して，営業部員特有の報酬査定を実行できません。

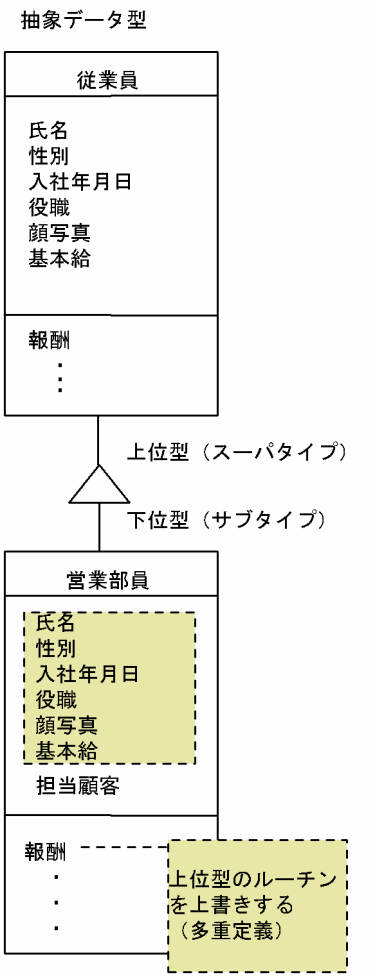
```
×　SELECT　社員番号，従業員..氏名，営業部員報酬(従業員)
　　FROM　　社員表
```

…「営業部員」でない「従業員」に対しても，営業部員特有の報酬査定を実行してしまいます。

HiRDB では，上位の抽象データ型で定義されたルーチンと同じ名前のルーチンを下位の抽象データ型を定義するときに上書きして定義できます。このように上書きして定義することを**多重定義**（override）といいます。

多重定義を次の図に示します。

図 3–42　多重定義



また，多重定義されたルーチンの実行では，その引数となる値の型に応じて，HiRDB が自動的に適切な定義に従って実行します。多重定義によって，呼び出すルーチンの名称を値の型によって変更する必要がなくなります。そのため，次に示す操作系 SQL を実行できます。

注意事項

この SQL を実行する前に，ALTER ROUTINE を実行し，SQL オブジェクトを再作成しておく必要があります。

```
SELECT    社員番号，従業員..氏名，報酬(従業員)
    FROM      社員表
```

…多重定義したルーチン「報酬」によって，実行時にそれぞれの引数の値に応じたルーチンが実行されます。

この SQL の実行では，**t_従業員型**の値に対しては，**t_従業員型**に定義されたルーチン「報酬」が実行され，**t_営業部員型**の値に対しては，**t_営業部員型**に定義されたルーチン「報酬」が実行されます。

なお，上記の SQL のように，アプリケーションではルーチンが多重定義されているかどうかを意識しないでルーチンを呼び出せます。また，多重定義によってルーチンを追加した場合でも，アプリケーションを変更しないで SQL を実行できるようになります。

3.5.3 隠蔽

「従業員」についての情報では，個人的な内部情報（例えば，操作「報酬」で査定処理するための情報や，属性「入社年月日」の値など）は，直接アプリケーションからはアクセスできないようにすることが好ましいと考えられます。その理由を次に示します。

- 情報秘匿のため，アプリケーションから直接参照させたくない場合
  （例）
  属性「基本給」の値は，「報酬」などのルーチンの内部処理で参照するが，直接外部には公開させたくない場合
- アプリケーションから直接参照しても意味がない場合
  （例）
  作業実績を評価してコード化した値を属性値として保持しているときに，そのような値をアプリケーションから参照しても用途がない場合
- 内部情報がアプリケーションによって直接変更されることを防ぎたい場合
  （例）
  属性「入社年月日」の値がアプリケーションによって来年の日付に変更されるのを防ぎたい場合

(1) 隠蔽レベル

HiRDB では，抽象データ型の属性及びルーチンの宣言で隠蔽レベルを指定して，それらへのアクセスを制御できます。

隠蔽レベルには次に示す三つのレベルがあります。

- **PRIVATE**
  その抽象データ型の定義内だけで，属性の値へのアクセス及びルーチンを使用できます。
  例えば，t_従業員型の属性「入社年月日」について PRIVATE を指定することが考えられます。
  PRIVATE を指定するときには，サブタイプの定義内やアプリケーションから，その属性へのアクセス及びルーチンを使用できないということを意識して，指定する必要があります。
- **PROTECTED**
  その抽象データ型の定義内及びその抽象データ型のサブタイプの定義内だけで，属性の値へのアクセス及びルーチンを使用できます。

例えば，t_従業員型のルーチン「報酬率」について PROTECTED を指定することが考えられます。これによって，t_従業員型のサブタイプ t_営業部員からは，属性「入社年月日」を直接参照することなく，「報酬率」の処理内容（入社年度から勤続年数を算出し，その値を基に報酬率を算出していることなど）を知らなくても，ルーチン「報酬率」を実行でき，報酬率を算出できます。

- **PUBLIC**

  PRIVATE，PROTECTED のような限定はなく，その抽象データ型やサブタイプ以外の抽象データ型の定義内やアプリケーションからも，属性の値へのアクセス及びルーチンを使用できます。

  例えば，ルーチン「勤続年数」のようにアクセスに制限が不要な場合に，PUBLIC を指定することが考えられます。

隠蔽レベルと抽象データ型の値の属性へのアクセス及びルーチンの使用可否を次の表に示します。

表 3–9　隠蔽レベルと抽象データ型の値の属性へのアクセス及びルーチンの使用可否

隠蔽レベル	アクセス元			
	その抽象データ型の定義内	サブタイプの抽象データ型の定義内	左記以外の抽象データ型の定義内	アプリケーション
PUBLIC	○	○	○	○
PROTECTED	○	○	×	×
PRIVATE	○	×	×	×

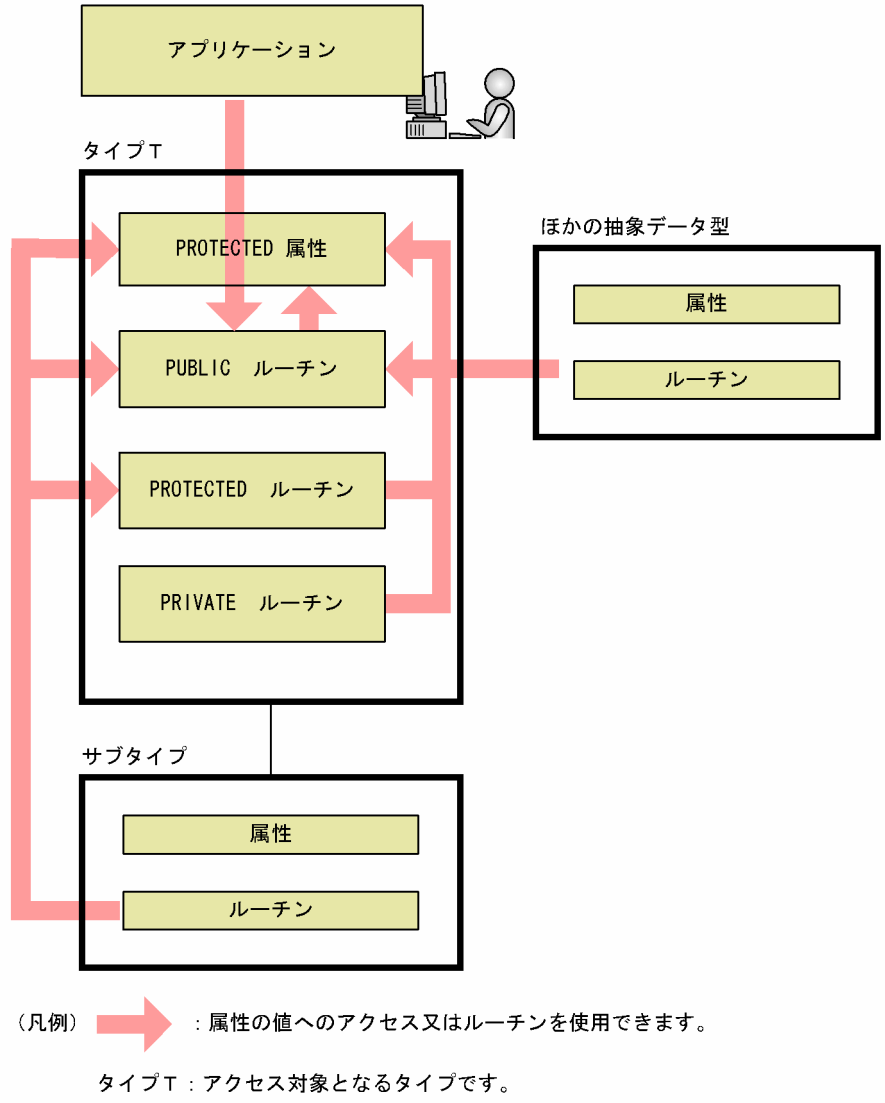
（凡例）

○：抽象データ型の属性の値へのアクセス及びルーチンを使用できます。

×：抽象データ型の属性の値へのアクセス及びルーチンを使用できません。SQL エラーとなります。

ある抽象データ型 T に対する，隠蔽レベルとアクセス可否の関係を次の図に示します。

図 3–43　隠蔽レベルとアクセス可否の関係



抽象データ型を定義した表の作成，設計方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。抽象データ型を定義した表のデータ操作方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

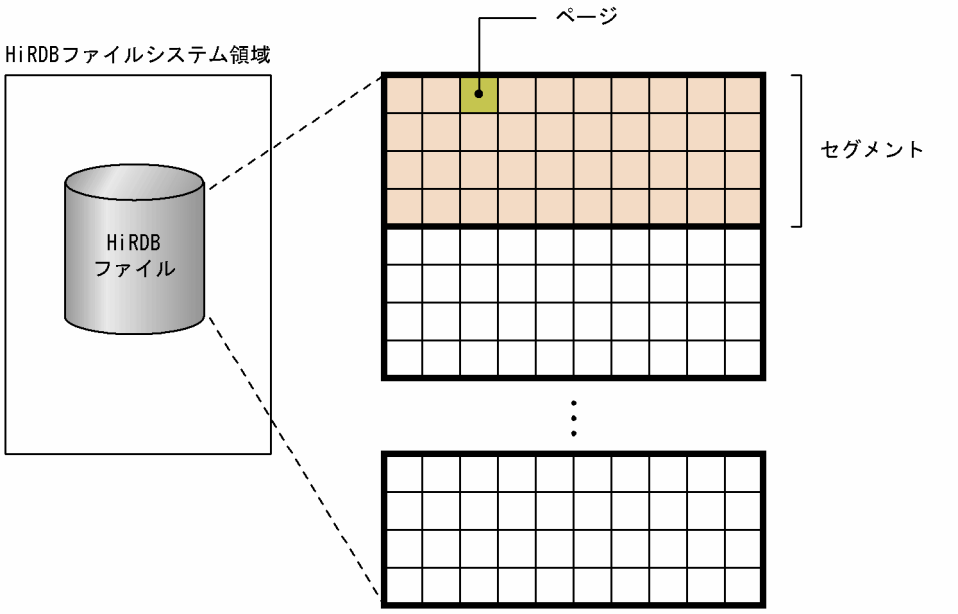
4 データベースの物理構造

この章では，データベースの物理構造（セグメント及びページ）について説明します。

4.1　データベースの物理構造の仕組み

データベースの物理構造を次の図に示します。

図 4–1　データベースの物理構造



〔説明〕

- **HiRDB ファイルシステム領域**
  HiRDB ファイルを作成する領域です。
- **HiRDB ファイル**
  表及びインデクスのデータを格納する HiRDB 専用のファイルです。
- **セグメント**
  表及びインデクスのデータを格納する最小単位です。1 セグメントには，一つの表又は一つのインデクスのデータだけが格納されます。連続した複数ページから構成されています。
- **ページ**
  データベースの入出力処理の最小単位です。ページ長を大きくすると，連続した行を同一ページ内に格納できるため，データを連続して処理する場合に入出力回数を削減できます。ページには次の表に示す種類があります。

  表 4–1　ページの種類

ページの種類	説明
データページ	表の行データを格納するページです。
インデクスページ	インデクスのキー値を格納するページです。
ディレクトリページ	RD エリアの状態の管理情報を格納するページです。

データベースの物理構造の指定方法

データベースの物理構造は RD エリアを定義するときに指定します。具体的には，次に示す制御文で指定します。

- データベース初期設定ユティリティ（pdinit）の **create rdarea 文**
- データベース構成変更ユティリティ（pdmod）の **create rdarea 文**

create rdarea 文の指定例を次に示します。

（例）

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

〔説明〕

1. RD エリアの名称（USRRD01）を指定します。
2. ページ長に 4096 バイトを指定します。
3. セグメントサイズに 20 ページを指定します。
4. RD エリアを構成する HiRDB ファイルシステム領域名と HiRDB ファイル名を指定します。

   C:¥rdarea01：HiRDB ファイルシステム領域名

   file01：HiRDB ファイル名
5. セグメント数を指定します。

4.2　セグメントの設計

4.2.1　セグメントの状態

セグメントには次の表に示す状態があります。

表 4–2　セグメントの状態

セグメントの状態	説明
使用中セグメント※	表又はインデクスのデータを格納しているセグメントです。 データが満杯でセグメント内にデータを追加できないセグメントを**満杯セグメント**，それ以外の使用中セグメントを**空きありセグメント**といいます。 空きありセグメントのうち，データの削除でセグメント内の全ページが空きページ（使用中空きページ又は未使用ページ）のセグメントを**使用中空きセグメント**といいます。
未使用セグメント	使用されたことがないセグメントです。このセグメントは RD エリア内のすべての表（又はインデクス）が使用できます。
空きセグメント	データを格納していないセグメントです。使用中空きセグメントと未使用セグメントは空きセグメントになります。

注※

使用中セグメントを使用できるのは，このセグメントにデータを格納した表又はインデクスだけです。ほかの表又はインデクスはこのセグメントを使用できません。

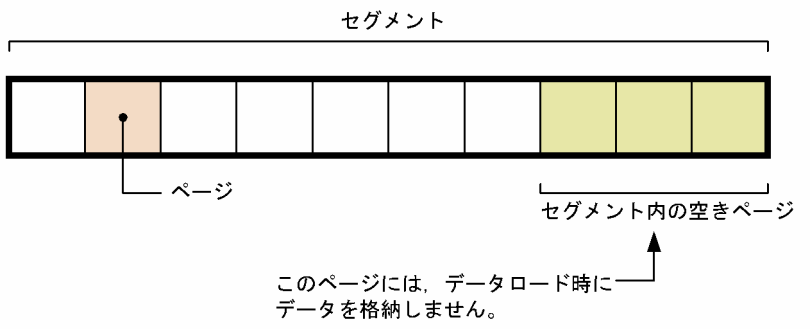
4.2.2　セグメントの設計方針

セグメントサイズはデータの入出力時間又はディスク所要量に影響を与えるため，綿密に設計する必要があります。通常は 1 セグメント当たり 10〜20 ページ程度にすることをお勧めします。セグメントの設計については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

4.2.3　セグメント内の空きページ比率

表を定義するときに**セグメント内の空きページ比率**を設定できます。セグメント内の空きページ比率を次の図に示します。

図 4–2　セグメント内の空きページ比率



〔説明〕

- セグメント内の空きページ比率を 30%としています。したがって，10 ページ中 3 ページが空きページになります。
- セグメント内の空きページ比率は，**CREATE TABLE** の **PCTFREE** オプションで指定します。空きページ比率は 0～50%の範囲で指定でき，省略値は 10%となります。
- ここで指定した空きページには，表へのデータロード（リロード又は表の再編成も含む）時にデータを格納しません。

セグメント内の空きページ比率を小さくすれば，データの格納効率が向上します。

セグメント内の空きページ比率を大きくすれば，性能が向上することがあります。例えば，クラスタキーを定義した表にデータを追加する場合，セグメント内の空きページ比率を設定しておくと，クラスタキーの値に近い場所のページにデータが格納されるため，データの入出力回数を削減できます。

セグメント内の空きページ比率の設計については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

4.2.4　セグメントの確保と解放

表を定義したときにはセグメントを確保しません。表にデータを格納するときに必要に応じてセグメントを確保します。一度確保したセグメント（一度使用したセグメント）はそのセグメントを解放しないかぎり，ほかの表又はインデクスが使用できません。このため，データの追加と削除を繰り返した場合，データ量が増えていないのに RD エリアが容量不足になることがあります。これを防ぐには次に示す操作を定期的に行ってセグメントを解放してください。

- データベース再編成ユティリティ（pdrorg コマンド）による表の再編成又はインデクスの再編成
- 空きページ解放ユティリティ（pdreclaim コマンド）による使用中空きセグメントの解放

表の再編成，インデクスの再編成，使用中空きセグメントの解放については，マニュアル「HiRDB システム運用ガイド」を参照してください。なお，これらの操作以外にも次に示す操作をした場合はセグメントを解放します。

- PURGE TABLE 文の実行
- RD エリアの再初期化
- 表の定義の削除
- インデクスの定義の削除
- データロードを作成モード（-d オプション指定）で実行

4.2.5　空き領域の再利用

空き領域の再利用機能を使用すると，データの削除でできる空き領域を有効活用できます。

(1)　データ格納時のサーチ方式

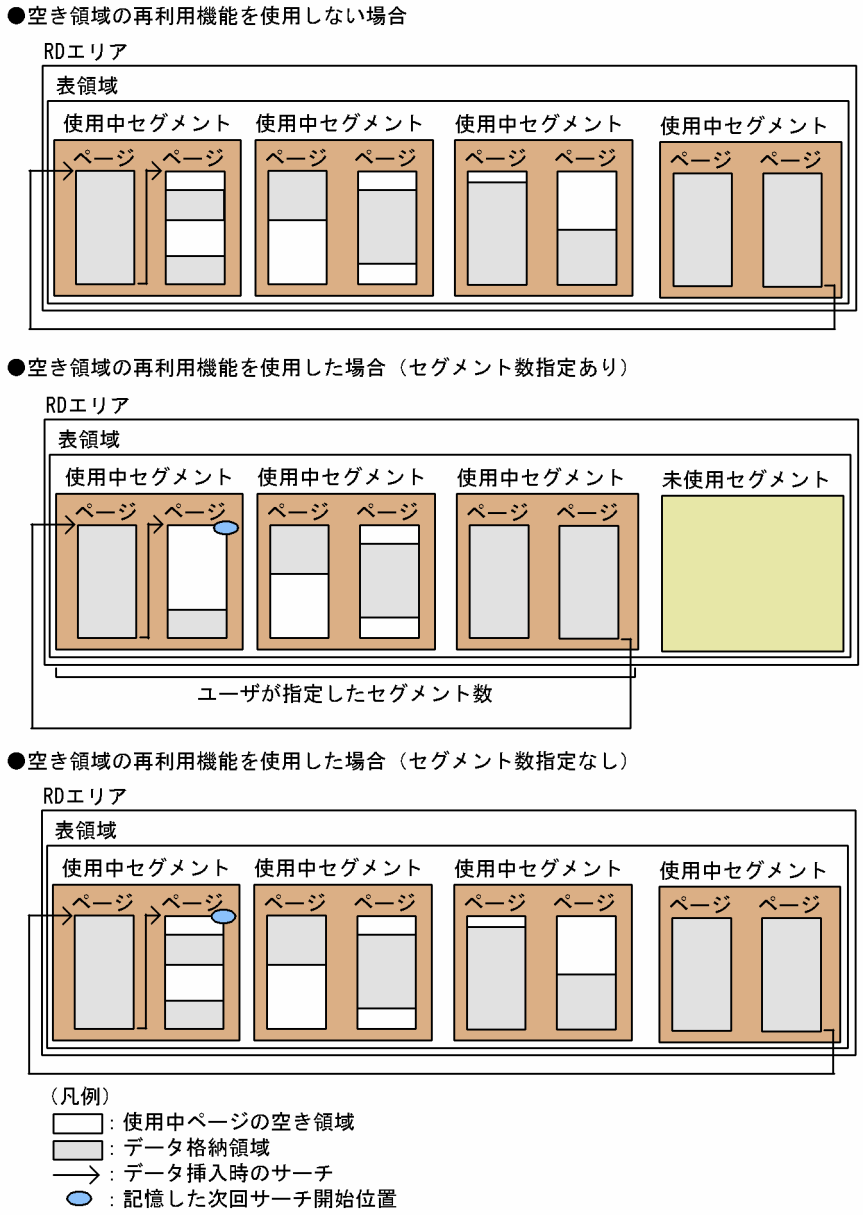
表にデータを格納するとき，格納領域をサーチする方式には次の二つのページサーチモードがあります。

- **新規ページ追加モード**
  使用中セグメントの最終ページが満杯になると，新規に未使用セグメントを確保します。RD エリア中に未使用ページがなくなると，使用中ページの空き領域を使用中セグメントの先頭からサーチして空き領域にデータを格納します。
- **空きページ再利用モード**
  使用中セグメントの最終ページが満杯になると，未使用セグメントを確保する前に使用中セグメント内の使用中ページの空き領域をサーチします。また，次回サーチ開始位置を記憶し，次に空き領域をサーチするときそこからサーチを開始します。

(2)　空き領域の再利用機能とは

空き領域の再利用機能とは，表の使用中セグメントがユーザの指定したセグメント数に達し，そのセグメントが満杯になるとページサーチモードを空きページ再利用モードに切り替えて，使用中ページの空き領域を使用する機能です。指定した数のすべてのセグメントに空き領域がなくなると，新規ページ追加モードに切り替わり，新規に未使用セグメントを確保します。なお，セグメント数を指定しないと RD エリア中に未使用ページがなくなるまで空き領域を再利用しません。空き領域の再利用機能を使用しない場合，毎回使用中セグメントの先頭から空き領域をサーチします。この機能を使用している場合，空きページ再利用モードに切り替わった後，次回サーチ位置を記憶してそれ以降をサーチするのでこの機能を使用しない場合より効率良くサーチできます。空き領域の再利用機能の概要を次の図に示します。

図 4–3　空き領域の再利用機能の概要



〔説明〕

- 空き領域の再利用機能を使用しない場合

  RD エリア中に未使用ページがなくなると，その後データが挿入されるたびに使用中セグメントの先頭から使用中ページの空き領域をサーチして空き領域にデータを格納します。

- 空き領域の再利用機能を使用した場合（セグメント数指定あり）

  指定したセグメント数に達した後で表にデータを挿入しようとすると，未使用セグメントを確保しないで，使用中ページの空き領域を使用中セグメントの先頭からサーチしてそこにデータを格納します。そこで次回サーチ開始位置を記憶しておき，次に空き領域をサーチするときはそこからサーチを開始します。

- 空き領域の再利用機能を使用した場合（セグメント数指定なし）

RD エリア中に未使用ページがなくなってからデータを挿入しようとすると，使用中ページの空き領域を使用中セグメントの先頭からサーチしてそこにデータを格納します。そこで次回サーチ開始位置を記憶しておき，次に空き領域をサーチするときはそこからサーチを開始します。

空き領域の再利用機能の詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(3) 適用基準

削除と挿入を繰り返すため，データ量に対してセグメントが大量に消費され，頻繁に再編成しなければならない業務で，再編成の回数をできるだけ減らしたい場合に空き領域の再利用機能を使用してください。

(4) 環境設定

空き領域の再利用機能を使用するための環境設定について次に示します。詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

1. **pd_assurance_table_no** オペランドに空き領域の再利用機能を使用する表数を指定します。
2. 定義系 SQL の CREATE TABLE の **SEGMENT REUSE** オプションでセグメント数を指定します。作成済みの表に対しては ALTER TABLE の SEGMENT REUSE オプションで指定します。

(5) 注意事項

次の場合，空き領域の再利用機能は動作しません。

- ハッシュ分割表のリバランス機能でデータを格納するとき
- データロードやデータベース再編成ユティリティ（pdrorg）で表にデータを格納するとき
- ユーザ LOB 用 RD エリアのとき

4.3　ページの設計

ここでは，ページの状態，及びページの設計方針について説明します。

4.3.1　ページの状態

ページには次の表に示す状態があります。

表 4–3　ページの状態

ページの状態	説明
未使用ページ	まだ割り当てられていないページです。
使用中空きページ	データの削除※によって，データが格納されていないページです。
使用中ページ	データが格納されていて，データを追加できる空き領域があるページです。 空き領域の再利用機能を使用している表の場合は，データの削除※によってページ内の空き領域が使用できないために，データを追加できなかったページを含みます。
使用中満杯ページ	データが格納されていて，データを追加できる空き領域がないページです。 空き領域の再利用機能を使用していない表及びインデクスの場合は，データの削除※によってページ内の空き領域が使用できないために，データを追加できなかったページを含みます。

注※

データの削除を実行したトランザクションが COMMIT するまで，データの削除によって発生した空き領域は使用できません。

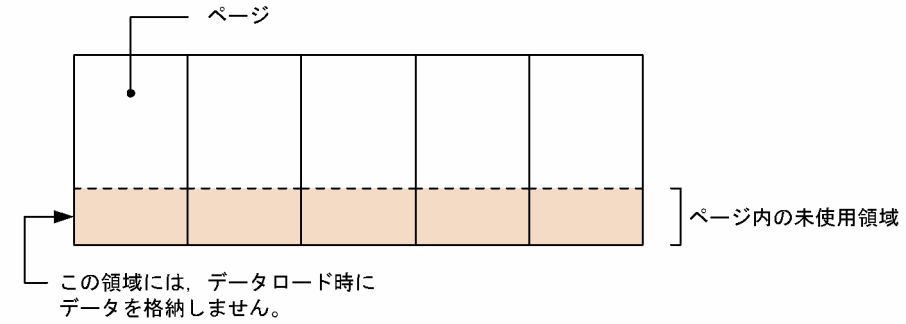
4.3.2　ページの設計方針

ページ長はデータの入出力時間に影響を与えるため，綿密に設計する必要があります。ページの設計については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

4.3.3　ページ内の未使用領域の比率

表又はインデクスを定義するときに**ページ内の未使用領域の比率**を設定できます。ページ内の未使用領域の比率を次の図に示します。

図 4–4　ページ内の未使用領域の比率



〔説明〕

- ページ内の未使用領域の比率を 30%としています。
- ページ内の未使用領域の比率は，**CREATE TABLE** 又は **CREATE INDEX** の **PCTFREE** オプションで指定します。未使用領域比率は 0～99%の範囲で指定でき，省略値は 30%となります。
- 未使用領域には，表へのデータロード（リロード又は表の再編成も含む）時にデータを格納しません。

ページ内の未使用領域の比率を小さくすれば，データの格納効率が向上します。ページ内の未使用領域の比率を大きくすれば，性能が向上することがあります。例えば，データの更新時，次に示すどちらかの条件を満たす場合はデータの入出力回数を削減できます。

- 更新前よりも行長が長くなる場合
- クラスタキーを指定した表に行を追加する場合

ページ内の未使用領域の比率の設計については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

4.3.4　ページの確保と解放

(1)　ページの確保

表を定義したときにはページを確保しません。表にデータを格納するときに必要に応じてページを確保します。一度確保したページ（一度使用したページ）はそのページを解放しないかぎり，再使用できません。

インデクスを定義した場合はデータ件数に応じてページを確保します。データ件数 0 件の場合は 1 ページ（ルートページ）だけを確保します。ただし，CREATE INDEX に EMPTY オプションを指定した場合（インデクスの実体を作成しない場合）はページを確保しません。

参考

- 非 FIX 表で行長が変わるデータ更新をした場合は行長が減った分の領域を再使用できません。

- インデクスページは削除ページに格納されていたキー値と同じキー値が追加されないかぎり，そのページを再使用しません。

注意事項

データの削除によって発生した空き領域があるページを再使用するときは次に示す制限があるので注意してください。

- 256 バイト以上の VARCHAR，BINARY 型，抽象データ型，及び繰返し列の分岐行は，そのページを使用できません。
- セグメントの使用率が 100%になるまで，データの挿入時にそのページを使用できません。
- DELETE を実行したトランザクションが COMMIT を発行するまで，DELETE によって発生した空き領域を使用できません。

(2) ページの解放

- セグメントが解放されるとセグメント内のページは解放されます。
- EXCLUSIVE 指定の LOCK 文で排他を掛けた表に対して，UAP がページ内の全行を削除した場合，そのページは解放されます。
- PURGE TABLE 文を実行すると表及びインデクスが使用中のセグメントが解放されるため，そのセグメント内のページも解放されます。ただし，インデクスのルートページは残ります。
- 空きページ解放ユティリティ（pdreclaim コマンド）で使用中空きページを解放できます。使用中空きページの解放については，マニュアル「HiRDB システム運用ガイド」を参照してください。

5 SQL によるデータベースアクセス

この章では，データベース操作言語 SQL によるデータベースのデータ操作について説明します。

5.1 HiRDB で使える SQL

ここでは，SQL を使ったデータベースへのアクセスの概要について説明します。SQL の文法についてはマニュアル「HiRDB SQL リファレンス」を，UAP の作成，設計方法についてはマニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.1.1 HiRDB の SQL の種類

表のデータを操作するにはデータベース操作言語 **SQL** を使用します。HiRDB の SQL で使う機能を次に示します。

データの基本操作

- データの検索
- データの更新
- データの削除
- データの挿入
- 特定データの探索
- データの演算
- データの加工
- 抽象データ型を含む表のデータ操作

UAP の開発工数の削減，通信，解析処理のオーバヘッド削減のための機能

- ストアドプロシジャ
- ストアドファンクション

データベースをアクセスする処理性能向上のための機能

- ブロック転送機能
- グループ分け高速化機能
- 配列を使用した FETCH 機能
- ホールダブルカーソル
- SQL 最適化オプション

5.1.2 SQL を実行する方法

HiRDB のデータベースに SQL でアクセスする方法を次に示します。

- **UAP** に SQL を記述して，その UAP の実行形式ファイルからデータベースにアクセスする方法（SQL を直接記述する UAP のことを**埋込み型 UAP** といいます）
- **HiRDB SQL Executer** で SQL を対話式で実行してデータベースにアクセスする方法
- **データベース定義ユティリティ（pddef）**で SQL を実行してデータベースにアクセスする方法

複雑な定型業務や演算処理が必要なときは，UAP を作成して表データを参照したり，単純なデータ操作によって対話型で SQL を実行したりと運用を使い分けられます。

埋込み型 UAP に使用できる高級言語を次に示します。

- C 言語
- C++言語
- COBOL 言語
- OOCOBOL 言語

5.2　データの基本操作

5.2.1　カーソル

表の検索結果は一般に複数行にわたります。UAP で複数行の検索結果を 1 行ずつ取り出すために最新の取り出し位置を保持するものを**カーソル**といいます。カーソルは，データの検索，更新及び削除で使用できます。

カーソルを使用する場合は，**DECLARE CURSOR** を指定します。また，カーソルは **OPEN 文**で開いて，**CLOSE 文**で閉じます。カーソルの位置を進めるには **FETCH 文**を使用します。

5.2.2　データの検索

データの検索とは，表を構成する行の中から，ある列に対して指定した条件を満たす行を選び出すことです。データの検索方法と SQL の指定例を次に示します。

データの検索方法

データの検索をするためには，**SELECT 文**を使用します。データの検索には，次に示す三つの方法があります。

- カーソルを使用した検索
- 複数の表の検索（SELECT 文の FROM 句指定）
- 行単位の検索（SELECT 文の選択式の ROW 指定）

データの検索の SQL 指定例

ここでは，カーソルを使用した検索の例を説明します。

（例）

1. カーソルの定義

カーソル名称を CUR1 として，在庫表（ZAIKO）から商品名（SNAME）が’スカート’である商品名，色（COL）及び単価（TANKA）を検索する例を次に示します。

```
DECLARE CUR1 CURSOR FOR
SELECT SNAME,COL,TANKA FROM ZAIKO
WHERE SNAME=N'スカート'
```

2. カーソルを開く

カーソル CUR1 を開きます。

```
OPEN CUR1
```

3. データの取り出し

カーソル CUR1 が開いた状態からカーソルを 1 行進めて，その行の内容を UAP 内の指定した領域（:XSNAME,:XCOL,:XTANKA）に格納します。

```
FETCH CUR1 INTO
  :XSNAME,
  :XCOL,
  :XTANKA
```

4. カーソルを閉じる

カーソル CUR1 を閉じます。

```
CLOSE CUR1
```

なお，ORDER BY 句の後に LIMIT を指定すると，先頭から n 行の検索結果を取得できます。LIMIT を指定した場合，SQL の検索性能を向上できることがあります。先頭から n 行の検索結果を取得する機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.2.3 データの更新

データの更新とは，表中の情報を変更することです。データの更新方法と SQL の指定例を次に示します。

データの更新方法

データを更新するときは，**UPDATE 文**を使用します。データの更新には，次に示す三つの方法があります。

- カーソルが指している行の更新
- 条件を満たす行だけの更新（UPDATE 文の WHERE 句指定）
- 行単位での更新（SET 句の ROW 指定）

データの更新の SQL 指定例

ここでは，条件を満たす行だけを更新する例を説明します。

（例）

在庫表（ZAIKO）から，商品コード（SCODE）が **411M** の在庫量（ZSURYO）を **20** に更新する UPDATE 文の例を示します。

```
UPDATE ZAIKO
  SET ZSURYO=20
  WHERE SCODE='411M'
```

5.2.4 データの削除

データの削除とは，表を構成する行の中からある列に対して指定した条件を満たす行又は表を構成するすべての行を取り除くことです。データの削除方法と SQL の指定例を次に示します。

データの削除方法

データを削除するときは，**DELETE 文**又は **PURGE TABLE 文**を使用します。データの削除には，次に示す三つの方法があります。

- カーソルが指している行の削除
- 条件を満たす行だけの削除（DELETE 文の WHERE 句指定）
- 表の全行削除（PURGE TABLE 文）

データの削除の SQL 指定例

ここでは，条件を満たす行だけを削除する例を説明します。

（例）

条件を満たす行だけを削除する場合として，在庫表（ZAIKO）から商品名（SNAME）がスカートのデータだけを削除する DELETE 文の例を次に示します。

```
DELETE FROM ZAIKO
  WHERE SNAME=N'スカート'
```

5.2.5 データの挿入

データの挿入とは，表に行を挿入することです。データの挿入方法と SQL の指定例を次に示します。

データの挿入方法

データを挿入するためには，**INSERT 文**を使用します。表中への行の挿入は，次に示す二つの方法があります。

- 列単位での行の挿入
- 行単位での行の挿入（INSERT 文の ROW 指定）

データの挿入の SQL 指定例

ここでは，列単位で行を挿入する例を説明します。

（例）

表と UAP との間で，値の受け渡しのために使用する埋込み変数（:ZSCODE,:ZSNAME,:ZCOL,:ZTANKA,:ZZSURYO）に設定されている値を在庫表（ZAIKO）の各列に挿入する INSERT 文の例を次に示します。

```
INSERT INTO ZAIKO (SCODE,SNAME,COL,TANKA,ZSURYO)
  VALUES(:ZSCODE,:ZSNAME,:ZCOL,:ZTANKA,:ZZSURYO)
```

5.2.6　特定データの探索

表中のデータを条件付きで操作するには，**探索条件**を指定します。探索条件とは行の選択条件のことで，「一定の範囲のデータ」や「ナル値でないデータ」などの条件があります。これらの条件を論理演算子を使用して複数組み合わせることもできます。特定データの探索方法と SQL の指定例を次に示します。

(1)　特定データの探索方法

表中のデータの探索には次に示す方法があります。

- 特定範囲内のデータの探索
- 特定の文字パターンの探索
- ナル値でないデータの探索
- 複数の条件を満たすデータの探索
- 副問合せを使用した検索

(2)　特定データの探索の SQL 指定例

(a)　特定範囲内のデータを探索する SQL の指定例

特定範囲内のデータを探索する方法には次に示す三つの方法があります。

- 比較述語（等価，大小比較のために使用します）
- BETWEEN 述語（一定の範囲のデータを取り出すときに使用します）
- IN 述語（指定した複数の値と一致するデータだけ取り出すときに使用します）

ここでは，比較述語を使用したデータの探索の例を説明します。

（例）

在庫表（ZAIKO）から，在庫量（ZSURYO）が 50 以下の，商品コード（SCODE）と商品名（SNAME）を検索する SELECT 文の例を次に示します。

```
SELECT SCODE,SNAME FROM ZAIKO
WHERE ZSURYO<=50
```

(b)　特定の文字パターンを探索する SQL の指定例

ここでは，特定の文字パターンを含む列がある行の探索の例を説明します。

（例）

LIKE 述語を使用します。在庫表（ZAIKO）から商品コード（SCODE）の 2 文字目が **L** の商品名（SNAME）と在庫量（ZSURYO）を検索する SELECT 文の例を次に示します。

```
SELECT SNAME,ZSURYO FROM ZAIKO
  WHERE SCODE LIKE '_L%'
```

(c) ナル値でないデータを探索する SQL の指定例

ここでは，表の列中にナル値が含まれない行の探索の例を説明します。

（例）

NULL 述語の NOT を組み合わせて使用します。在庫表（ZAIKO）から商品名（SNAME）が未設定（ナル値）ではない商品コード（SCODE）を検索する SELECT 文の例を次に示します。

```
SELECT SCODE FROM ZAIKO
  WHERE SNAME IS NOT NULL
```

(d) 複数の条件を満たすデータを探索する SQL の指定例

ここでは，複数の条件を組み合わせて，該当するデータを含む行の探索の例を説明します。

（例）

論理演算子（AND，OR，NOT）を使用します。在庫表（ZAIKO）から商品名（SNAME）がブラウス又はポロシャツで，在庫量（ZSURYO）が 50 以上の商品の商品コード（SCODE）と在庫量を検索する SELECT 文の例を次に示します。

```
SELECT SCODE,ZSURYO FROM ZAIKO
  WHERE (SNAME=N'ブラウス'
        OR SNAME=N'ポロシャツ')
        AND ZSURYO=>50
```

(e) 副問合せを使用した探索の SQL の指定例

探索結果を SELECT 文の条件の中に指定して，より複雑な問い合わせを記述できます。これを**副問合せ**といいます。副問合せには次に示す二つの方法があります。

- **限定述語**
  副問合せの結果が，指定した比較条件を満たしているかどうかを判定し，副問合せの結果の範囲を絞り込むときに限定述語を使用します。
- **EXISTS 述語**
  副問合せの結果が空集合でないかどうかを判定するときに使用します。

ここでは，限定述語を使用した副問合せを使用した探索の例を説明します。

（例）

限定述語を使用した場合として，在庫表（ZAIKO）からブラウスのどの在庫量（ZSURYO）よりも多く在庫がある商品の商品コード（SCODE）と商品名（SNAME）を検索する SELECT 文の例を次に示します。

```
SELECT SCODE,SNAME FROM ZAIKO
   WHERE ZSURYO>ALL
```

```
        (SELECT ZSURYO FROM ZAIKO
              WHERE SNAME=N'ブラウス')
```

5.2.7　データの演算

表中の列の数値や日時を検索して，演算した結果を取り出せます。データの演算方法と SQL の指定例を次に示します。

データの演算方法

データの演算には次に示す方法があります。

- 文字列データに対する連結演算
- 数値データの四則演算
- 日付又は時刻データの演算
- スカラ関数を使用した演算
- CASE 式を使用した条件付き値の指定
- CAST 指定を使用した明示的型変換

データの演算の SQL 指定例

ここでは，数値データの四則演算の例を説明します。

（例）

在庫表（ZAIKO）から商品名（SNAME）が'ソックス'の，単価（TANKA）及び在庫量（ZSURYO）から売上げ見込みを演算し，商品コード（SCODE）及び演算結果（百円単位）を取り出す SELECT 文の例を次に示します。

```
SELECT SCODE,TANKA*ZSURYO/100,N'百円' FROM ZAIKO
   WHERE SNAME=N'ソックス'
```

5.2.8　データの加工

表中のデータを取り出すとき，グループ分けや，昇順や降順に並べ替えるなど**データの加工**ができます。データの加工方法と SQL の指定例を次に示します。

データの加工方法

表中のデータの加工には次に示す方法があります。

- データのグループ分け（**GROUP BY 句**指定，及び集合関数）
- データの昇順又は降順での並べ替え（**ORDER BY 句**指定）
- 重複したデータの排除（**DISTINCT 指定**）
- 行の集合（導出表）間の集合演算（UNION 指定，又は EXCEPT 指定）

データの加工の SQL 指定例

ここでは，データを昇順に並べ替える（ソートする）例を説明します。

（例）

在庫表（ZAIKO）から，商品コード（SCODE），在庫量（ZSURYO）を検索し，商品コード（SCODE）を昇順に並べ替える（ソートする）SELECT 文の例を次に示します。

```
SELECT SCODE,ZSURYO FROM ZAIKO
  ORDER BY SCODE
```

5.2.9　抽象データ型を含む表データの操作

抽象データ型を含む表データの操作方法について説明します。

(1)　プラグインから提供される抽象データ型の場合

プラグインを使用する場合には，プラグインから提供される関数を指定して UAP を作成すると，文書，空間データなどのマルチメディアデータを操作できます。

ここでは，全文検索プラグイン（HiRDB Text Search Plug-in）を使用した場合の例を説明します。

(a)　データの検索

ここでは，論理述語を使用したデータの検索の例を説明します。

（例）

薬品管理表の列「取扱説明書」から「効能」という構造部分に，**頭痛**というキーワードがある，薬品 ID を検索する SELECT 文の例を次に示します。なお，SQL 中には，プラグインから提供される全文構造検索条件に一致する文書を抽出するための関数 contains を使用しています。

```
SELECT 薬品ID FROM 薬品管理表
  WHERE contains(取扱説明書,'添付文書データ[効能{"頭痛"}]') IS TRUE
```

(b)　データの更新

ここでは，データの更新の例を説明します。

（例）

薬品管理表の列「薬品 ID」が薬品 2 の「取扱説明書」のデータを UPDATE 文で更新する例を次に示します。なお，SQL 中には，プラグインから提供される関数 SGMLTEXT を使用しています。

```
UPDATE 薬品管理表 SET 取扱説明書 = SGMLTEXT(:sgml)
  WHERE 薬品ID = '薬品2'
```

なお，UPDATE 文の前に，あらかじめ次に示す BLOB 型の埋込み変数「sgml」を定義しているものとします。

```
EXEC SQL BEGIN DECLARE SECTION;                                    1
  SQL TYPE IS BLOB(300K)sgml;                                      1
EXEC SQL END DECLARE SECTION;                                      1
strcpy(sgml.sgml_data,char_ptr_pointing_to_a_sgml_text);           2
sgml.sgml_length = strlen(char_ptr_pointing_to_a_sgml_text);       3
```

〔説明〕

1. BLOB 型の埋込み変数「sgml」を定義します。
2. 埋込み変数「sgml」に，更新する新しいデータを格納します。
3. 作成した BLOB データの属性値 sgml_length を，格納したデータの長さにセットします。

(c) データの削除

ここでは，データ型の削除の例を説明します。

（例）

薬品管理表の列「薬品 ID」が薬品 2 の行を DELETE 文で削除する例を次に示します。

```
DELETE FROM 薬品管理表
  WHERE 薬品ID = '薬品2'
```

(d) データの挿入

ここでは，データの挿入の例を説明します。

（例）

薬品管理表に，列「薬品 ID」が薬品 25 の行を INSERT 文で挿入する例を次に示します。なお，SQL 中には，プラグインから提供される関数 SGMLTEXT を使用しています。

```
INSERT INTO 薬品管理表(薬品ID,取扱説明書)
  VALUES('薬品25',SGMLTEXT(:sgml))
```

なお，INSERT 文の前に，あらかじめ次に示す BLOB 型の埋込み変数「sgml」を定義しているものとします。

```
EXEC SQL BEGIN DECLARE SECTION;                                    1
  SQL TYPE IS BLOB(300K)sgml;                                      1
EXEC SQL END DECLARE SECTION;                                      1
strcpy(sgml.sgml_data,char_ptr_pointing_to_a_sgml_text);           2
sgml.sgml_length = strlen(char_ptr_pointing_to_a_sgml_text);       3
```

〔説明〕

1. BLOB 型の埋込み変数「sgml」を定義します。
2. 埋込み変数「sgml」に，更新する新しいデータを格納します。
3. 作成した BLOB データの属性値 sgml_length を，格納したデータの長さにセットします。

(2) HiRDB XML Extension から提供される抽象データ型の場合

HiRDB XML Extension から提供される関数を指定して UAP を作成すると，XML 形式のデータを操作できます。

(a) 検索

ここでは，XML 形式のデータの検索例について説明します。

（例 1）

書籍管理表から，書籍 ID が「126513592」である書籍情報を VARCHAR 型の値として取り出します。SQL 文は，次のように記述できます。

```
SELECT 書籍ID, XMLSERIALIZE(書籍情報 AS VARCHAR(32000))
    FROM 書籍管理表
    WHERE 書籍ID = 126513592
```

（例 2）

書籍管理表から，カテゴリが「データベース」の書籍のタイトルを取り出します。XQuery 式による評価結果を取り出すため，XMLQUERY 関数を使用します。また，XQuery 式の評価結果が空のシーケンスである行を出力しないように，XMLEXISTS 述語を使用します。SQL 文は，次のように記述できます。

```
  SELECT 書籍ID,
         XMLSERIALIZE(
           XMLQUERY('/書籍情報/タイトル'
                    PASSING BY VALUE 書籍情報
                    RETURNING SEQUENCE EMPTY ON EMPTY)
           AS VARCHAR(32000))
    FROM 書籍管理表
    WHERE XMLEXISTS('/書籍情報[カテゴリ="データベース"]'
                    PASSING BY VALUE 書籍情報)
```

（例 3）

書籍管理表から，カテゴリが「データベース」の書籍のタイトルを結合して取り出します。各行の XML 型の値を一つの XML 型の値として出力するため，XMLAGG 集合関数を使用します。SQL 文は，次のように記述できます。

```
SELECT XMLSERIALIZE(
         XMLAGG(
           XMLQUERY('/書籍情報/タイトル'
                     PASSING BY VALUE書籍情報
                     RETURNING SEQUENCE EMPTY ON EMPTY)
         )
         AS VARCHAR(32000))
       FROM 書籍管理表
       WHERE XMLEXISTS('/書籍情報[カテゴリ="データベース"]'
                       PASSING BY VALUE 書籍情報)
```

（例 4）

書籍管理表から，タイトルが「SQL 徹底解説」の書籍情報と，カテゴリが同じ書籍のタイトルを取り出します。各行の XML 型の値を一つの XML 型の値にして，その値に対して XQuery 式を評価するため，

XMLQUERY 関数の XML 問合せ引数に XMLAGG 集合関数を指定します。SQL 文は，次のように記述できます。

```
SELECT
  XMLSERIALIZE(
    XMLQUERY(
      '$BOOKS/書籍情報[カテゴリ=$BOOKS/書籍情報[タイトル="SQL徹底解説"]/カテゴリ]'
      PASSING BY VALUE XMLAGG(書籍情報) AS BOOKS
      RETURNING SEQUENCE EMPTY ON EMPTY)
    AS VARCHAR(32000))
  FROM 書籍管理表
```

(3)　ユーザが定義した抽象データ型の場合

ユーザが定義した抽象データ型を含む表のデータ操作をする場合には，ルーチン又はコンポネント指定を使用します。コンポネント指定は，抽象データ型を構成する列の属性を操作する場合に使用します。ここでは，ユーザが定義した抽象データ型を含む表のデータ操作の例を説明します。

(a)　抽象データ型の列の検索

ユーザが定義した抽象データ型を含む表の列を検索する例を説明します。

（例）

社員表から，ユーザ定義関数「勤続年数」を使用して，勤続年数が 20 年以上の社員の社員番号を SELECT 文で検索するための例を次に示します。

```
SELECT 社員番号
  FROM 社員表
  WHERE 勤続年数(従業員)>=20
```

(b)　抽象データ型の列の更新

ユーザが定義した抽象データ型を含む表の列を更新する例を説明します。

（例）

社員表の列「社員番号」が **900123** の社員の属性「役職」を**主任**に更新する UPDATE 文の例を次に示します。従業員表の列「従業員 NO」が 900123 の，列「従業員」の属性「役職」をコンポネント指定「従業員..役職」を使用して，「主任」に更新する UPDATE 文の例を次に示します。

```
UPDATE 社員表
SET 従業員..役職 ='主任'
WHERE 社員番号 = '900123'
```

(c)　抽象データ型の列の削除

ユーザが定義した抽象データ型を含む表の列を削除する例を説明します。

（例）

従業員表の列「従業員」の属性「役職」が「一般」のデータをコンポネント指定「従業員..役職」を使用して，削除する DELETE 文の例を次に示します。

```
DELETE FROM 従業員表
  WHERE 従業員..役職='一般'
```

(d) データの挿入

ユーザが定義した抽象データ型を含む表にデータを挿入する例を説明します。

（例）

社員表にコンストラクタ関数「t_従業員」を使用して，列「社員番号」が **990070** の行を挿入する INSERT 文の例を次に示します。なお，:x 顔写真は BLOB 型の埋込み変数で，顔写真の画像が設定されているものとします。

```
INSERT INTO 従業員表
  VALUES ('990070',t_従業員('タシロケイコ',
                            'F',
                            '一般',
                            '1999-04-01',
                            :x顔写真 AS BLOB,
                            140000
                            ))
```

5.3　ストアドプロシジャ，ストアドファンクション

5.3.1　ストアドプロシジャ，ストアドファンクションの概要

データベースへの一連の操作を**手続き**として定義したものを**ストアドプロシジャ**，データベースへの一連の操作を**関数**として定義したものを**ストアドファンクション**といいます。

ストアドプロシジャ及びストアドファンクションを定義すると，アクセス手順を記述した SQL オブジェクトが作成され，それらの定義情報とともにデータベースに格納されます。ストアドプロシジャ及びストアドファンクションは，処理手続きを SQL，Java，又は C 言語で記述できます。

SQL で記述したものを **SQL ストアドプロシジャ**，**SQL ストアドファンクション**，Java で記述したものを **Java ストアドプロシジャ**，**Java ストアドファンクション**，C 言語で記述したものを **C ストアドプロシジャ**，**C ストアドファンクション**といいます。

また，ストアドプロシジャとストアドファンクションを総称して，**ストアドルーチン**といいます。さらに，すべての利用者を示す PUBLIC を所有者として定義するストアドルーチンを**パブリックルーチン**といいます。パブリックルーチンとして定義すると，他ユーザが定義したストアドルーチンを使用する場合，UAP 中からストアドルーチンを呼び出すときに，所有者の認可識別子を指定する必要がなくなります（ルーチン識別子だけ指定します）。パブリックルーチンは，CREATE PUBLIC PROCEDURE 又は CREATE PUBLIC FUNCTION で定義できます。

Java ストアドプロシジャ，Java ストアドファンクションについては，「Java ストアドプロシジャ，Java ストアドファンクション」を参照してください。C ストアドプロシジャ，C ストアドファンクションについては，「C ストアドプロシジャ，C ストアドファンクション」を参照してください。

ストアドプロシジャは，0 個以上の，入力，出力又は入出力パラメタを持ち，SQL の CALL 文で呼び出せます。ストアドファンクションは，0 個以上の入力パラメタを持ち，戻り値を返せるため，SQL 中で値式として指定して呼び出せます。ただし，ストアドファンクションでは，データの加工などのデータ処理だけで，データベース中の表へのアクセスはできません。

(1)　ストアドプロシジャの業務への適用

ストアドプロシジャがどのような業務に適用できるかを説明します。例えば，商品管理業務で商品の売り上げ状況を分析するために，次に示す処理があるとします。

- **商品ごとに，月初めから月末までの 1 か月分の受注量の合計を計算し，商品の発売日から現在までの受注合計表に反映する**

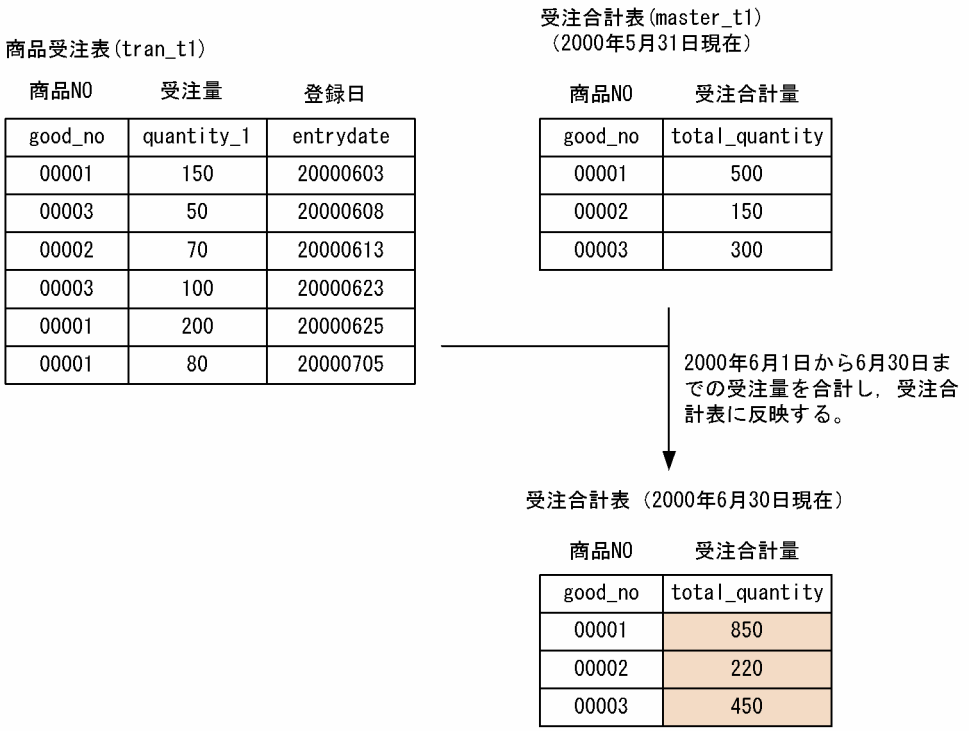
これは，次に示す複数のデータベースアクセス処理によって実現します。

1. カーソルを使用して，商品受注表から，月初めから月末までの間に登録された商品の商品 NO，受注量，登録日を検索する（SELECT 文）

2. カーソルを使用して，受注合計表と商品受注表とで，商品 NO が同じ商品の，受注合計量を検索する（SELECT 文）
3. 月初めから月末までの間に登録された商品の受注量の合計と，商品の発売日から先月までの受注合計量との和を計算し，受注合計表を更新する（INSERT 文，UPDATE 文）

このような業務の「一連のデータベースをアクセスする処理」を UAP 側に作成しないでデータベース側に登録して，複数の UAP からその処理を呼び出して使用できます。ストアドプロシジャを適用できる業務を次の図に示します。

図 5–1　ストアドプロシジャを適用できる業務



図「ストアドプロシジャを適用できる業務」のように「商品 NO」,「受注量」,「登録日」を列として持つ「商品受注表」から，「2000 年 6 月 1 日から 6 月 30 日までに登録された受注量」を合計し，「商品 NO」,「受注合計量」を列として持つ「受注合計表」に反映するというデータベースのアクセス処理を考えます。このような場合，次に示すようなデータベースアクセス処理をストアドプロシジャとして定義して，データベースに登録します。

ストアドプロシジャの処理内容

ある一定期間の受注量を合計し，受注合計表に反映する

UAP からは，次に示す引数を指定してストアドプロシジャを呼び出すだけで，図「ストアドプロシジャを適用できる業務」に示す商品管理業務を実現できます。

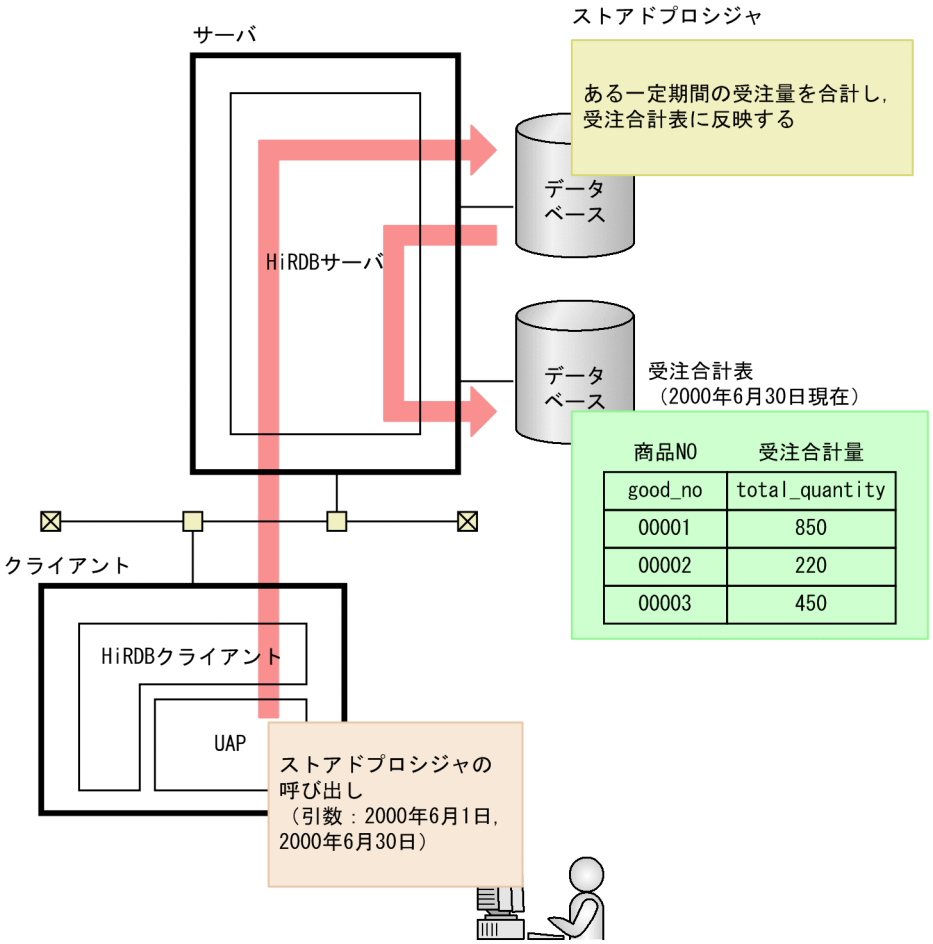
UAP 側で指定する引数

受注量の合計を計算する期間

（最初の日付「2000 年 6 月 1 日」と最後の日付「2000 年 6 月 30 日」）

ストアドプロシジャの使用形態を次の図に示します。

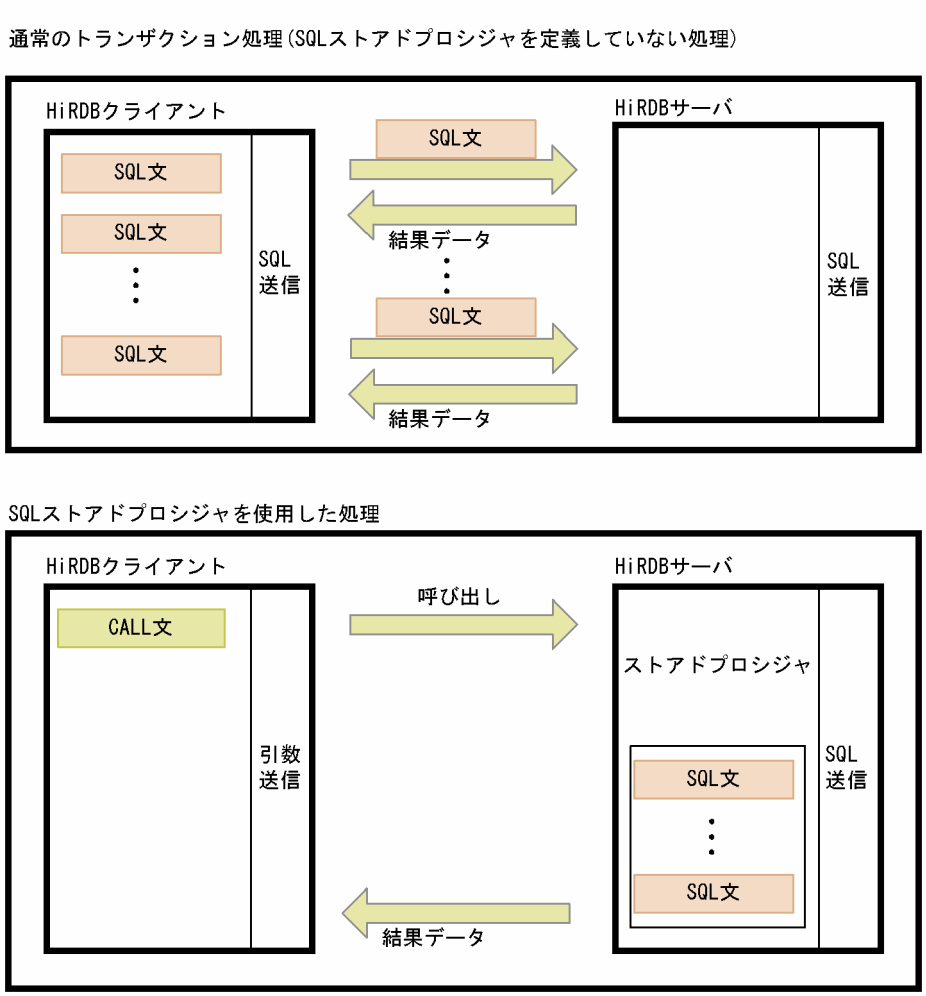
図 5-2　ストアドプロシジャの使用形態



このように，ストアドプロシジャを使用すると，データベースのアクセス処理をデータベース側に登録できるため，データベースのアクセス処理を部品化できます。また，このようなデータベースのアクセス処理が変更された場合でも，ストアドプロシジャを変更するだけで，UAP 側は変更する必要がないため，UAP の開発工数が削減できます。

さらに，SQL ストアドプロシジャを例にすると，複数の SQL を実行するためには SQL 実行回数分だけアプリケーションからデータベースをアクセスする必要があります。それに対して，実行する複数の SQL をストアドプロシジャとしてデータベースに登録しておけば，データベースに 1 回アクセスして呼び出すだけで，複数の SQL 文を実行できます。これによって，HiRDB サーバと HiRDB クライアント間で生じるアプリケーションのデータ受け渡しなどの通信処理やフロントエンドサーバの SQL 解析処理のオーバヘッドを削減できます。SQL ストアドプロシジャによる通信処理を次の図に示します。

図 5–3　SQL ストアドプロシジャによる通信処理



(2)　ストアドファンクションの適用

ストアドファンクションでは，条件分岐（IF 文）や SQL の繰り返し（WHILE 文）などのルーチン制御 SQL を使用し，「データベースの加工などのデータ処理」をユーザが任意に関数として定義して，データベースに登録できます。そのため，データ処理を部品化できます。また，プラグインを使用する場合には，プラグインから提供される関数が，ストアドファンクションとしてデータベースに登録されます。

(3)　ストアドプロシジャ，ストアドファンクションを格納するための RD エリアの作成

ストアドプロシジャ又はストアドファンクションを使用する場合は，次に示す RD エリアを作成する必要があります。

- データディクショナリ LOB 用 RD エリア
- データディクショナリ用 RD エリア

ストアドプロシジャとストアドファンクションを格納するための RD エリアの作成については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(4) ストアドプロシジャ，ストアドファンクションを呼び出すための UAP の作成

ストアドプロシジャ及びストアドファンクションを呼び出す方法について説明します。ストアドプロシジャとストアドファンクションを呼び出すための UAP の作成については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

ストアドプロシジャの呼び出し

UAP 中に SQL の **CALL 文**を指定して，ストアドプロシジャを呼び出せます。

ストアドファンクションの呼び出し

SQL 中に値として「**関数呼出し**」を指定して，ストアドファンクションを呼び出せます。

関数の呼び出し時には引数を指定できます。さらに，SQL の **RETURN 文**で値が返されます。

(5) ストアドファンクションのオーバロード

パラメタの数やデータ型が異なる場合，名前が同じ複数のストアドファンクションを定義できます。名前が同じストアドファンクションを互いに**オーバロード**されているといいます。このオーバロードの機能によって，パラメタのデータ型が異なる複数の関数に同じ名前を割り当てられるため，同じ機能の関数の名前を統一できます。ストアドファンクションを呼び出すと，「指定した名前と同じ名前で，指定した引数の数とパラメタの数が一致する関数」の中では，次に示す関数が実行される候補となります。

- 各引数のデータ型と，対応するパラメタのデータ型が完全に一致する関数
- 各引数のデータ型と，対応するパラメタのデータ型が完全に一致しない場合には，引数とパラメタのデータ型を左側から順番に比較して，より優先順位の低いパラメタの中で最も優先順位の高いパラメタを持つ関数

呼び出す関数の決定規則については，マニュアル「HiRDB SQL リファレンス」を参照してください。

(6) SQL での手続き，ユーザ定義関数の定義

SQL での手続き及びユーザ定義関数の定義について説明します。SQL での手続き，ユーザ定義関数の定義方法の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

ストアドプロシジャを作成するための手続きの定義

ストアドプロシジャを作成するためには，まず，次に示すどちらかの SQL で「手続き」を定義します。

- CREATE PROCEDURE
- CREATE TYPE で指定する「手続き本体」（抽象データ型に手続きを指定する場合）

ストアドファンクションを作成するためのユーザ定義関数の定義

ストアドファンクションを作成するためには，まず，次に示すどちらかの SQL でユーザが任意に定義する「ユーザ定義関数」を定義します。

- CREATE FUNCTION

- CREATE TYPE で指定する「関数本体」（抽象データ型に関数を指定する場合）

なお，手続き，ユーザ定義関数及びシステム定義関数を総称して**ルーチン**といいます。

(7)　ストアドプロシジャ，ストアドファンクションのデータベースへの登録

手続き又はユーザ定義関数を定義する SQL をデータベース定義ユティリティ（pddef）で実行すると，自動的に手続き又はユーザ定義関数がコンパイルされ，SQL オブジェクトが作成されます。さらに，SQL の定義のソースと SQL オブジェクトは，データディクショナリ LOB 用 RD エリアに格納されます。これで，ストアドプロシジャ又はストアドファンクションがデータベースに登録されたことになります。ストアドプロシジャとストアドファンクションの作成方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(8)　ストアドプロシジャ，ストアドファンクションの再作成

表又はインデクスの定義の変更や，DROP PROCEDURE でストアドプロシジャを削除した場合には，無効になったストアドプロシジャを ALTER PROCEDURE で再作成する必要があります。さらに，DROP FUNCTION でストアドファンクションを削除した場合には，無効になったストアドファンクションを ALTER ROUTINE で再作成する必要があります。ストアドプロシジャ又はストアドファンクションの再作成については，マニュアル「HiRDB システム運用ガイド」を参照してください。

5.4　Java ストアドプロシジャ，Java ストアドファンクション

処理手続きを Java で記述したストアドプロシジャ，ストアドファンクションを，**Java ストアドプロシジャ**，**Java ストアドファンクション**といいます。Java ストアドプロシジャ，Java ストアドファンクションを総称して，**外部 Java ストアドルーチン**といいます。ここでは，外部 Java ストアドルーチンについて説明します。

5.4.1　Java ストアドプロシジャ，Java ストアドファンクションを使用できる環境

Java ストアドプロシジャ，Java ストアドファンクションを使用できる環境（HiRDB サーバの適用 OS）を次の表に示します。

表 5–1　Java ストアドプロシジャ，Java ストアドファンクションを使用できる環境

HiRDB サーバの適用 OS	使用可否	
	Type2 JDBC ドライバ使用時	Type4 JDBC ドライバ使用時
AIX	×	○
Linux	×	○
Windows (x86)	○	○
Windows (x64)	×	○

（凡例）

○：使用できます。

×：使用できません。

5.4.2　外部 Java ストアドルーチンの特長

外部 Java ストアドルーチンの特長を次に示します。

1. **サーバ，クライアント間の通信オーバヘッドがありません**

   外部 Java ストアドルーチンは，SQL ストアドプロシジャ，SQL ストアドファンクションと同様に，サーバ側で処理をします。したがって，サーバ，クライアント間での通信によるオーバヘッドはありません。

2. **手続き本体，関数本体を Java で記述できます**

   記述言語が Java なので，SQL で記述するよりも高度な制御ができます。

3. **異種 DBMS でも動作できます**

Java はプラットフォームに依存しない言語です。したがって，Java で作成したプログラムは，外部 Java ストアドルーチンを提供する異種 DBMS でも動作できます。

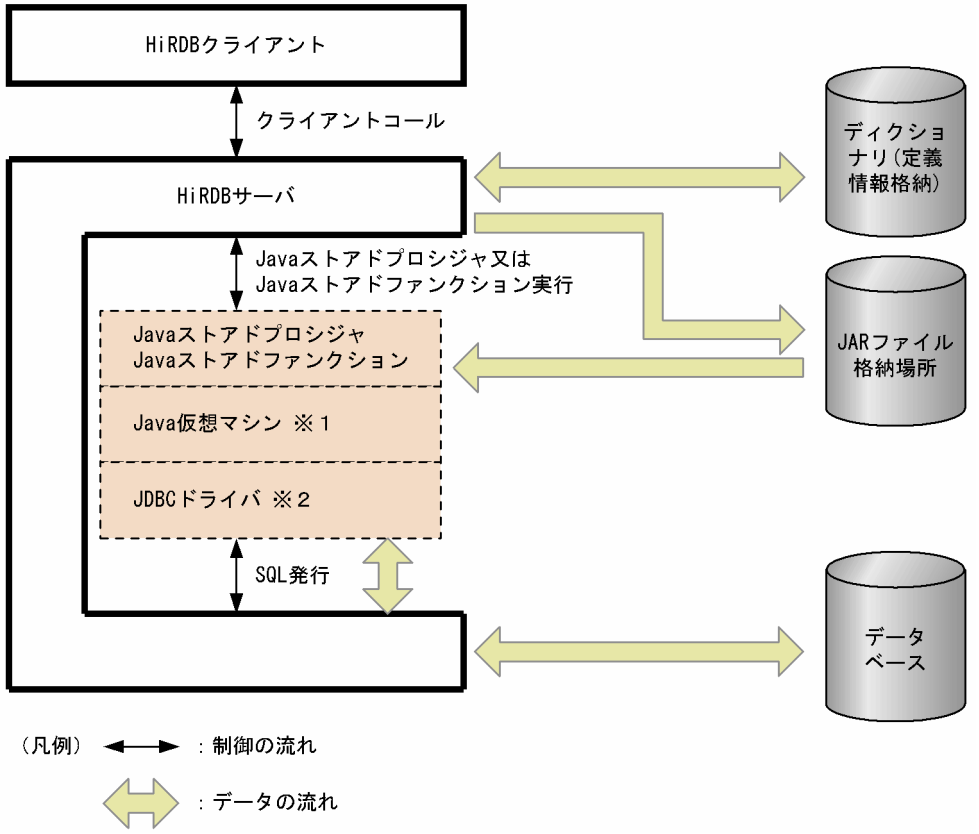
4. **デバッグが簡単です**

SQL ストアドプロシジャ，SQL ストアドファンクションのデバッグをする場合，実際にサーバ側で動作させる必要があります。これに対して，外部 Java ストアドルーチンのデバッグは，クライアント側に Java 言語のデバッガを用意することで，データベースアクセスを含めたデバッグができます。

5.4.3　システム構成（Java 仮想マシンの位置づけ）

HiRDB システムでの Java 仮想マシンの位置づけを次の図に示します。

図 5–4　HiRDB システムでの Java 仮想マシンの位置づけ



注※1

Java 仮想マシンは JRE（Java 実行環境）に含まれています。各プラットフォームのベンダのホームページから，JRE についての情報を取得し，JRE を入手してください。

注※2

JDBC ドライバは HiRDB が標準提供しています。

JRE の入手方法，及びシステム構成例については，マニュアル「HiRDB システム運用ガイド」を参照してください。

5.4.4　外部 Java ストアドルーチンを実行するためには

Java ストアドプロシジャ，Java ストアドファンクションを使用する場合の環境設定方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

5.4.5　外部 Java ストアドルーチンの実行手順

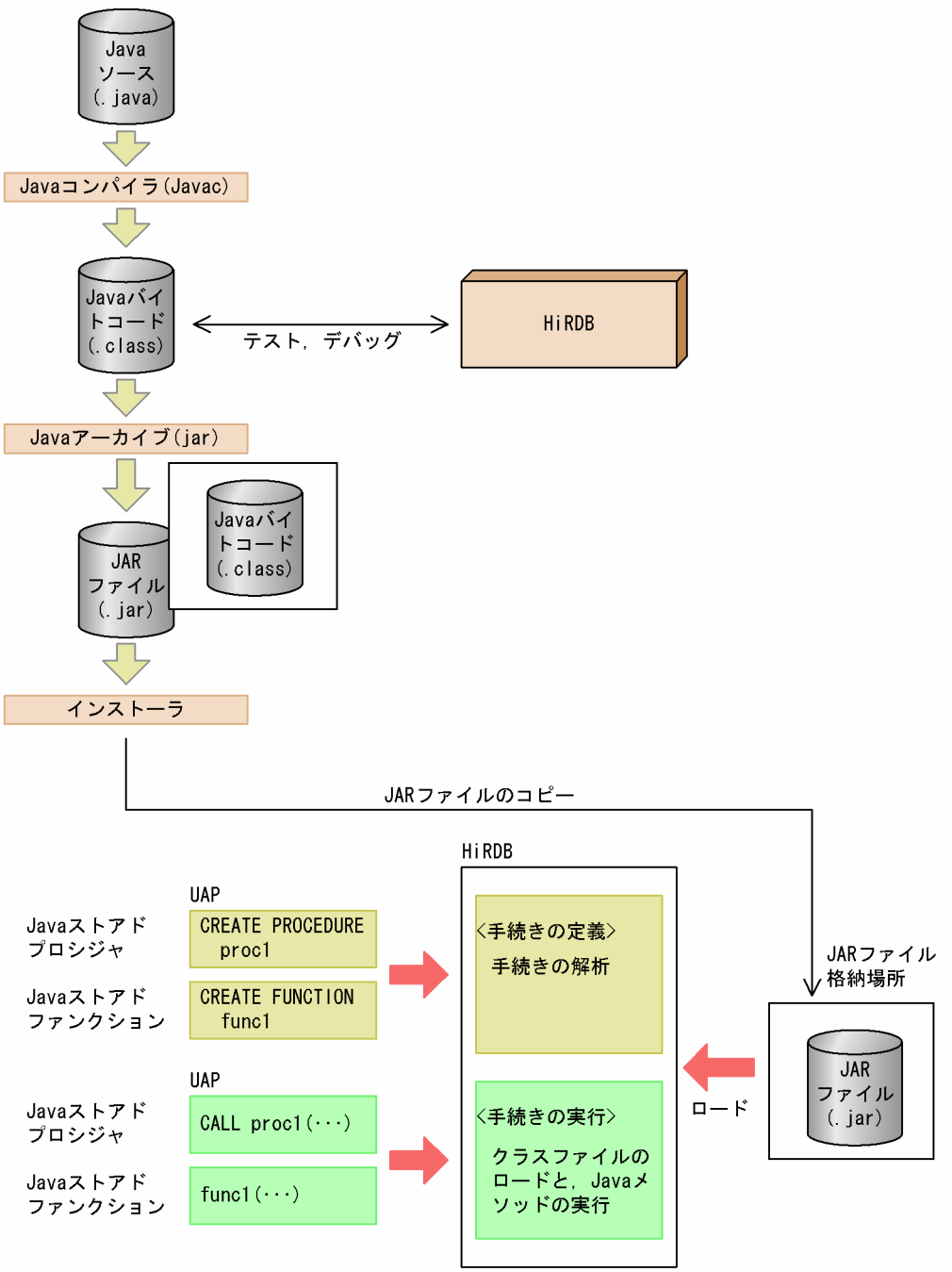
外部 Java ストアドルーチンを実行するときの手順を次に示します。

〈手順〉

1. 外部 Java ストアドルーチンの作成
2. JAR ファイルの新規登録
3. 外部 Java ストアドルーチンの定義
4. 外部 Java ストアドルーチンの実行

外部 Java ストアドルーチンの作成から実行までの流れを次の図に示します。

図 5–5　外部 Java ストアドルーチンの作成から実行までの流れ



(1)　外部 Java ストアドルーチンの作成

手続き又は関数を Java で記述し，その Java プログラムをコンパイルします。コンパイルすると，Class ファイルが作成されます。クライアント側の Java 仮想マシンで Class ファイルのテスト・デバッグをして，その後，Class ファイルから JAR ファイルを作成します。

(2)　JAR ファイルの新規登録

JAR ファイルを HiRDB へ新規登録します。

HiRDB 管理者が新規登録する場合

pdjarsync コマンドを使用します。

UAP 開発者が新規登録する場合

埋込み言語の INSTALL JAR 又は REPLACE JAR を使用します。これらの SQL を pddef 又は UAP に記述して実行します。

(3) 外部 Java ストアドルーチンの定義

JAR ファイルから外部 Java ストアドルーチンを定義します。外部 Java ストアドルーチンを定義する場合は，CREATE PROCEDURE 又は CREATE FUNCTION を使用します。

(4) 外部 Java ストアドルーチンの実行

ストアドプロシジャ，ストアドファンクションの実行と同様に，CALL 文又は関数呼出しを指定して SQL を実行します。CALL 文を実行すると，Java メソッドが Java ストアドプロシジャとして実行されます。また，関数呼出しを実行すると，Java メソッドが Java ストアドファンクションとして実行されます。外部 Java ストアドルーチンは，サーバ側の Java 仮想マシンで実行されます。

5.5　C ストアドプロシジャ，C ストアドファンクション

処理手続きを C 言語で記述したストアドプロシジャ，ストアドファンクションを，**C ストアドプロシジャ**，**C ストアドファンクション**といいます。C ストアドプロシジャ，C ストアドファンクションを総称して，**外部 C ストアドルーチン**といいます。ここでは，外部 C ストアドルーチンについて説明します。

5.5.1　外部 C ストアドルーチンの特長

外部 C ストアドルーチンの特長を次に示します。

- **サーバ，クライアント間の通信オーバヘッドがありません**
  外部 C ストアドルーチンは，SQL ストアドプロシジャ，SQL ストアドファンクションと同様に，サーバ側で処理をします。したがって，サーバ，クライアント間での通信によるオーバヘッドはありません。
- **手続き本体，関数本体を C 言語で記述できます**
  C 言語の命令を直接記述できるため，SQL 制御文よりも柔軟に処理を記述できます。
- **デバッグが簡単です**
  SQL ストアドプロシジャ，SQL ストアドファンクションのデバッグをする場合，実際にサーバ側で動作させる必要があります。これに対して，外部 C ストアドルーチンのデバッグは，クライアント側に C 言語のデバッガを用意することで，デバッグができます。

5.5.2　外部 C ストアドルーチンの実行手順

外部 C ストアドルーチンを実行するときの手順を次に示します

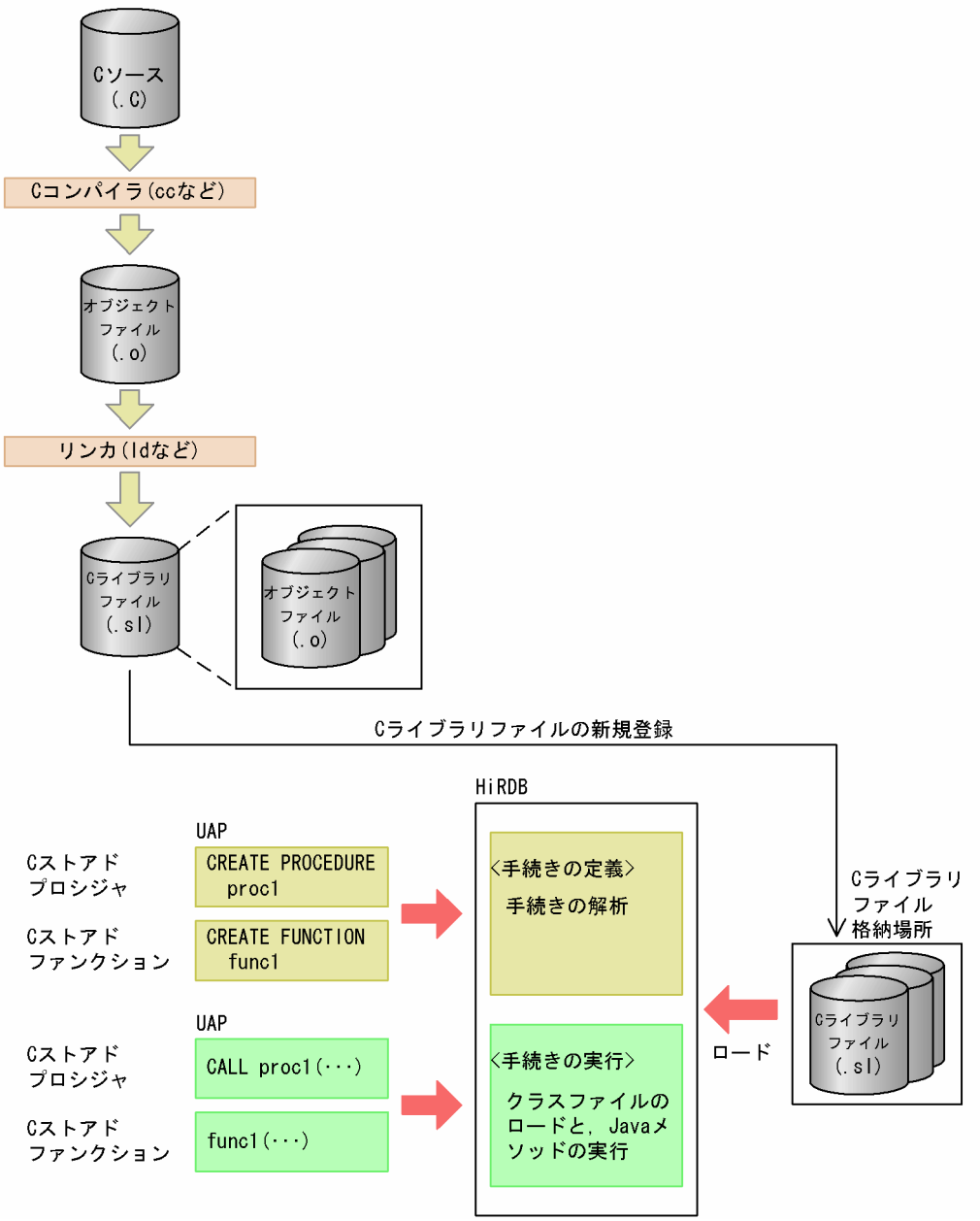
〈手順〉

1. 外部 C ストアドルーチンの作成
2. C ライブラリファイルの新規登録
3. 外部 C ストアドルーチンの定義
4. 外部 C ストアドルーチンの実行

外部 C ストアドルーチンの実行方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

外部 C ストアドルーチンの作成から実行までの流れを次の図に示します。

図 5–6　外部 C ストアドルーチンの作成から実行までの流れ



注

C ライブラリファイルの拡張子は OS によって異なります。

(1)　外部 C ストアドルーチンの作成

手続き又は関数を C 言語で記述し，その C プログラムをコンパイルします。コンパイルすると，オブジェクトファイルが作成されます。その後，複数のオブジェクトファイルをリンケージして C ライブラリファイルを作成します。

(2)　C ライブラリファイルの新規登録

C ライブラリファイルを HiRDB へ新規登録します。

HiRDB 管理者が新規登録する場合

pdclibsync コマンドを使用します。

UAP 開発者が新規登録する場合

SQL の INSTALL CLIB 又は REPLACE CLIB を UAP に記述して実行します。

(3) 外部 C ストアドルーチンの定義

CREATE PROCEDURE 又は CREATE FUNCTION を使用して，外部 C ストアドルーチンを定義します。

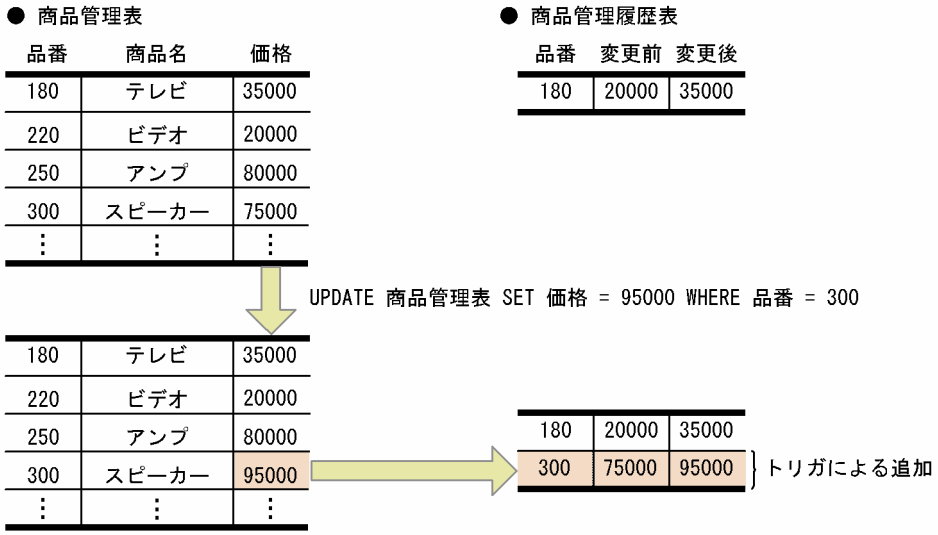
(4) 外部 C ストアドルーチンの実行

ストアドプロシジャ，ストアドファンクションの実行と同様に，CALL 文又は関数呼出しを指定して SQL を実行します。CALL 文を実行すると，C 関数が C ストアドプロシジャとして実行されます。また，関数呼出しを実行すると，C 関数が C ストアドファンクションとして実行されます。

5.6.1　トリガの概要

トリガを定義すると，ある表への操作（更新，挿入，及び削除）を契機に自動的に SQL 文を実行させることができます。トリガは，定義する表，トリガを動作させる契機となる SQL（**トリガ契機となる SQL**），自動的に実行させる SQL 文（**トリガ SQL 文**），その動作が実行される条件（**トリガ動作の探索条件**）などを指定して定義します。トリガを定義した表にトリガ動作の探索条件を満たす SQL 文が実行されると，トリガ SQL 文が自動的に実行されます。トリガの概要を次の図に示します。

図 5–7　トリガの概要



〔説明〕

商品管理表に，価格が変更されると商品管理履歴表に変更内容を蓄積するトリガを定義しています。UAP からトリガ契機となる SQL（この場合，UPDATE）が実行されると，自動的に商品管理履歴表に価格が変更になった品番，変更前の価格，及び変更後の価格が追加されます。

なお，トリガを表に定義すると，その表を使用する関数，手続き及びトリガの SQL オブジェクトは無効になるため，SQL オブジェクトを再作成する必要があります。また，トリガが使用しているリソース（表，インデクスなど）が定義，定義変更，又は削除された場合，トリガの SQL オブジェクトは無効になるため，SQL オブジェクトを再作成する必要があります。詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(1)　適用基準

UAP で次のような処理をする場合，トリガの使用をお勧めします。

- ある表の更新に伴ってほかの表を必ず更新する
- ある表の更新に伴って，その更新行中のある列を必ず更新する（列と列を関連づける）

(2)　トリガ定義の準備

トリガを定義すると，指定したトリガ動作手続きの SQL オブジェクトが自動的に作成され，データディクショナリ LOB 用 RD エリアに格納されます。そのため，トリガを定義する場合，データディクショナリ LOB 用 RD エリアに十分な容量を確保しておく必要があります。データディクショナリ LOB 用 RD エリアの容量の見積もりについては，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

また，トリガ契機となる SQL を実行するためには，SQL オブジェクト用バッファ長を指定するときにトリガの SQL オブジェクトについても考慮しておく必要があります。SQL オブジェクト用バッファ長の見積もりについては，マニュアル「HiRDB システム定義」を参照してください。

(3)　トリガの定義

トリガの定義，SQL オブジェクトの再作成，及び削除には次の定義系 SQL を使用します。

CREATE TRIGGER

トリガを定義します。トリガは所有する表にだけ定義でき，ほかのユーザが所有する表には定義できません。

ALTER TRIGGER

既に定義されているトリガの SQL オブジェクトを再作成します。定義系 SQL の **ALTER ROUTINE** でも再作成できます。

DROP TRIGGER

トリガを削除します。

5.7　整合性制約

データベースのデータが正しい状態であることを保証するために必要な制約を**整合性制約**といいます。データベースへのアクセスによって，データの不整合が発生しないように，整合性制約を設定する必要があります。HiRDB には，次に示す二つの整合性制約があります。

- 非ナル値制約
- 一意性制約

5.7.1　非ナル値制約

列中のデータにナル値を許さない制約を定義できます。これを**非ナル値制約**といいます。非ナル値制約は列単位に指定します。非ナル値制約を定義した列に対しては，ナル値を含む行の追加や更新ができません。非ナル値制約を定義した列に対しては，常に値が定まっていることが要求されるため，ナル値を与えようとすると制約違反となります。

非ナル値制約を定義する場合は，**CREATE TABLE** で **NOT NULL** オプションを指定します。非ナル値制約の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

備考

主キーを定義した列には非ナル値制約が定義されます。

5.7.2　一意性制約

列中のデータの重複を許さない（列中のデータが常に一意である）制約を定義できます。これを**一意性制約**といいます。一意性制約を定義した列に対しては，一意な値を含む行だけ，追加や更新ができます。一意性制約を定義した列に対しては，常に値が一意であることが要求されるため，一意ではない値を与えようとすると制約違反になります。一意性制約は次に示す列に指定できます。

- クラスタキーを定義する列
- インデクスを定義する列

クラスタキーを定義する列に一意性制約を定義する場合は，**CREATE TABLE** で **UNIQUE CLUSTER KEY** オプションを指定します。インデクスを定義する列に一意性制約を定義する場合は，**CREATE INDEX** で **UNIQUE** オプションを指定します。一意性制約の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

備考

主キーを定義した列には一意性制約が定義されます。

5.8　参照制約

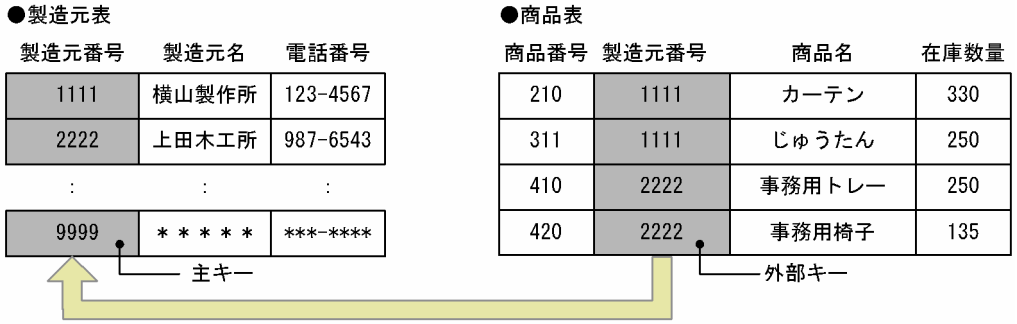
5.8.1　参照制約の概要

データベース中の表は，それぞれ独立しているのではなく，お互いに関連を持っている場合があります。一方の表に関連するデータがないと，ほかの表でそのデータの意味がないことがあります。表間のデータの参照整合性を保つため，表定義時に特定の列（**外部キー**といいます）に定義する制約が**参照制約**です。参照制約及び外部キーを定義した表を**参照表**，外部キーによって参照表から参照される表を**被参照表**といいます。なお，被参照表には外部キーによって参照される主キーを定義しておく必要があります。

なお，SQL やユティリティの実行などで被参照表と参照表間の参照整合性が保証できなくなる場合があります。この場合，参照表は検査保留状態になります。検査保留状態については，「検査保留状態」を参照してください。

被参照表と参照表の例を次の図に示します。この例では，商品表が参照表，製造元表が被参照表となります。参照表の外部キーから主キーを参照し，製造元名が分かります。

図 5–8　被参照表と参照表の例



参照制約については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

参照制約の効果

参照制約を定義すると，表間のデータの整合性チェック，及びデータ操作を自動化できるので UAP を作成するときの負荷を軽減できます。ただし，被参照表や参照表を更新する場合，データの整合性をチェックするため，チェックに掛かる処理時間が増加します。

(1)　参照制約の定義

参照制約を有効にするためには，外部キーによって参照される主キーを被参照表に定義しておく必要があります。定義系 SQL の **CREATE TABLE** で被参照表に **PRIMARY KEY**（主キー）を指定します。また，検査保留状態を使用するには，**pd_check_pending オペランド**に **USE** を指定するか，又はオペランドの指定を省略します。

参照表には，**FOREIGN KEY**（外部キー）を指定し，FOREIGN KEY 句中に次の指定をします。

- 参照する列
- 被参照表
- 参照制約動作

  参照制約動作は被参照表に対する挿入，更新，又は削除時の動作を **CASCADE**，又は **RESTRICT** で指定します。

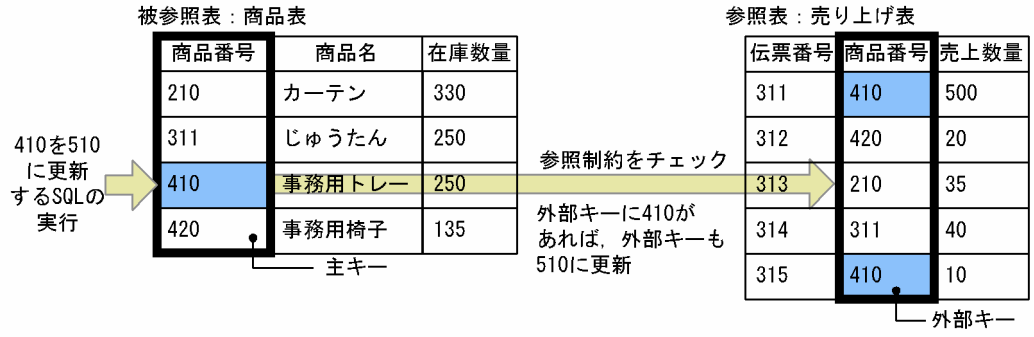
CASCADE，又は RESTRICT を指定した場合の被参照表と参照表の動作について説明します。

(a)　CASCADE を指定している場合

CASCADE を指定すると，被参照表の主キーに変更があった場合，外部キーも同じように変更されます。なお，参照表の外部キーに変更がある場合，主キーに変更後の値と同じ値の行があるかどうかをチェックして，参照制約違反エラーになれば外部キーは変更されません。

CASCADE を指定している場合，被参照表及び参照表に SQL を実行するときの動作の例を次の図に示します。

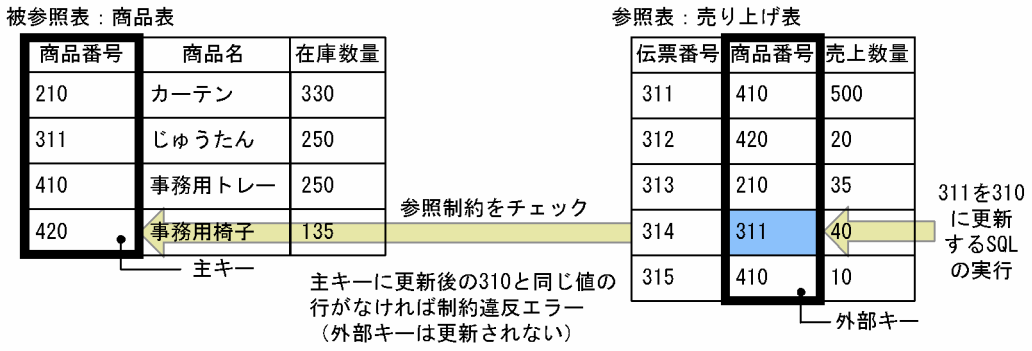
図 5–9　被参照表に更新 SQL を実行するときの動作の例（CASCADE 指定時）



〔説明〕

主キーの値と同じ値の行が外部キーにあれば，制約を保持するために，外部キーも主キーと同じように変更されます。この場合，被参照表への更新は実行されます。挿入及び削除も同じです。

図 5–10　参照表に更新 SQL を実行するときの動作の例（CASCADE 指定時）



〔説明〕

更新後の外部キーの値と同じ値の行が主キーにあれば，外部キーへの更新が実行されます。同じ値の行がなくても，外部キーにナル値があるときは外部キーへの更新が実行されます。ナル値がないときは参

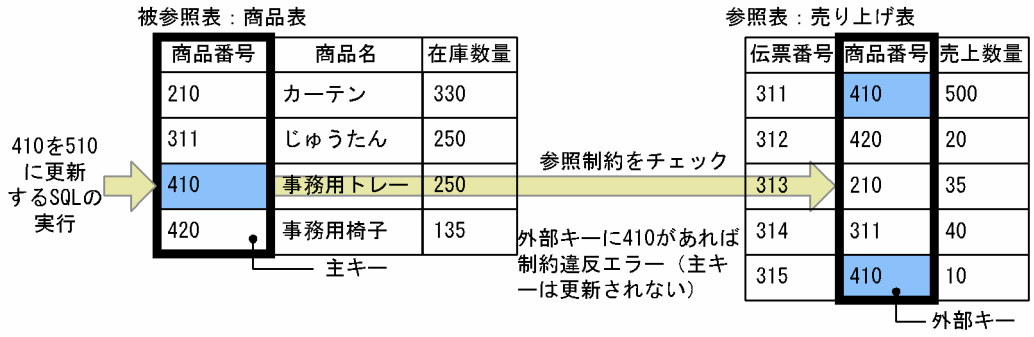
照制約違反エラーになります。この場合，被参照表には特に影響はありません。挿入及び削除も同じです。

(b)　RESTRICT を指定する場合

RESTRICT を指定すると，被参照表の主キーに変更がある場合，外部キーに同じ値の行があれば，参照制約違反エラーになり，主キーは変更されません。なお，外部キーに変更がある場合，主キーに同じ値の行があるかどうかをチェックして，参照制約違反エラーになれば外部キーは変更されません。

RESTRICT を指定している場合，被参照表に SQL を実行するときの動作を次の図に示します。参照表の動作は，CASCADE 指定時（図「参照表に更新 SQL を実行するときの動作の例（CASCADE 指定時）」を参照）と同じです。

図 5–11　被参照表に更新 SQL を実行するときの動作の例（RESTRICT 指定時）



(2)　データ操作と整合性

被参照表及び参照表に対する操作系 SQL（PURGE TABLE 文を除きます）によるデータ操作は，HiRDB が SQL 実行時にチェックし，整合性を保証します。ただし，次に示すデータ操作をした場合，整合性を保証できなくなることがあります。

- データベース作成ユティリティ（pdload）によるデータロード
- データベース再編成ユティリティ（pdrorg）による再編成，及びリロード
- データベース構成変更ユティリティ（pdmod）による RD エリアの再初期化（initialize rdarea）
- 更新可能なオンライン再編成の追い付き反映（pdorend コマンド）【UNIX 版限定】
- PURGE TABLE 文
- ALTER TABLE 文による表の分割格納条件の変更

これらの操作をした場合，データの整合性を確認する必要があります。データの整合性確認手順については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。また，pd_check_pending オペランドに USE を指定していて，これらの操作をした場合，参照表は検査保留状態になります。

5.9　検査制約

5.9.1　検査制約の概要

データベース中の表のデータは，値の範囲や条件など制限を持つ場合が多くあります。例えば，商品の情報をデータベースに格納する場合，商品価格として負の値はあり得ません。そのため，負の値はデータベースに存在してはいけない値であり，挿入又は更新時に値をチェックする必要があります。このように，データ挿入又は更新時に制約条件をチェックし，条件を満たさないデータの場合は操作を抑止することで表データの整合性を保つ機能が**検査制約**です。また，このマニュアルでは検査制約を定義した表を**検査制約表**といいます。

なお，ユティリティの実行などで検査制約表のデータの整合性が保証できなくなる場合があります。この場合，検査制約表は検査保留状態になります。検査保留状態については，「検査保留状態」を参照してください。

検査制約については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

検査制約の効果

検査制約を定義すると，データの挿入又は更新時のチェックを自動化できるので UAP を作成するときの負荷を軽減できます。ただし，検査制約表を更新する場合，データの整合性をチェックするため，チェックに掛かる処理時間が増加します。

(1)　検査制約の定義

検査制約は，定義系 SQL の **CREATE TABLE** で **CHECK** を指定し，表の値の制約条件を探索条件で指定します。また，検査保留状態を使用するには，**pd_check_pending オペランド**に **USE** を指定するか，又はオペランドの指定を省略します。

(2)　データ操作と整合性

検査制約表に対する操作系 SQL による更新，追加，又は削除は，HiRDB が SQL 実行時にチェックし，整合性を保証します。ただし，次に示すユティリティによる操作をした場合，チェックをしないため，整合性を保証できなくなることがあります。

- データベース作成ユティリティ（pdload）によるデータロード
- データベース再編成ユティリティ（pdrorg）によるリロード
- 更新可能なオンライン再編成の追い付き反映（pdorend コマンド）【UNIX 版限定】

これらの操作をした場合，データの整合性を確認する必要があります。データの整合性確認手順については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。また，pd_check_pending オペランドに USE を指定していて，これらの操作をした場合，検査制約表は検査保留状態になります。

5.10　検査保留状態

参照制約及び検査制約が定義された表に対する SQL やユティリティの実行などで，表データの整合性を保証できなくなった場合，HiRDB は参照表又は検査制約表に対するデータ操作を制限します。このように，整合性を保証できないためにデータ操作を制限された状態を**検査保留状態**といいます。参照表又は検査制約表を検査保留状態にして，データ操作を制限するためには，pd_check_pending オペランドに USE を指定するか，又はオペランドの指定を省略する必要があります。検査保留状態の表は，**整合性チェックユティリティ**（**pdconstck**）を使用して検査保留状態を解除します。また，整合性チェックユティリティを使用して，強制的に検査保留状態にもできます。

pd_check_pending オペランドに NOUSE を指定していると，表間で参照整合性を保証できない場合でもデータ操作を制限しません。そのため，整合性が保証できなくなる SQL やユティリティを実行した場合は，整合性チェックユティリティで強制的に検査保留状態にしてから，整合性を確認してください。

検査保留状態，及び整合性チェックユティリティを使用したデータの整合性確認手順についての詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

5.10.1　検査保留状態の設定又は解除の契機

整合性チェックユティリティ以外に，次に示すユティリティ，コマンド，及び SQL で参照表を検査保留状態に設定したり，解除したりできます。

- データベース作成ユティリティ（pdload）の constraint 文での指定
- データベース再編成ユティリティ（pdrorg）（リロード，再編成）の constraint 文での指定
- データベース構成変更ユティリティ（pdmod）（RD エリアの再初期化）
- 更新可能なオンライン再編成の追い付き反映（pdorend -p コマンド）【UNIX 版限定】
- PURGE TABLE 文
- ALTER TABLE（CHANGE RDAREA）

それぞれの詳細について，ユティリティ及びコマンドはマニュアル「HiRDB コマンドリファレンス」を，SQL はマニュアル「HiRDB SQL リファレンス」を参照してください。

5.10.2　検査保留状態の表に対して制限される操作

検査保留状態の表に対して実行できなくなる操作を次の表に示します。

表 5–2　検査保留状態の表に対する操作可否

検査保留状態の表に対する操作	操作可否
リバランスユティリティ（pdrbal）	実行できません。

検査保留状態の表に対する操作	操作可否
SELECT 文（対象表の検索，及び対象表から作成したリストの検索） INSERT 文（対象表への挿入） UPDATE 文（対象表の更新） DELETE 文（対象表からの行削除） ASSIGN LIST 文（対象表からのリスト作成）	次の条件をどちらも満たす場合だけ，操作できます。それ以外は操作できません。 • 操作対象表が分割表で，分割条件がキーレンジ分割又は FIX ハッシュ分割 • 操作対象となる RD エリアが検査保留状態でない
pdrorg（再編成）	フレキシブルハッシュ分割の分割表に対して再編成を実行する場合，操作できないときがあります。詳細は，マニュアル「HiRDB コマンドリファレンス」の「データベース再編成ユティリティ（pdrorg）」を参照してください。

5.11　データベースをアクセスする処理性能の向上

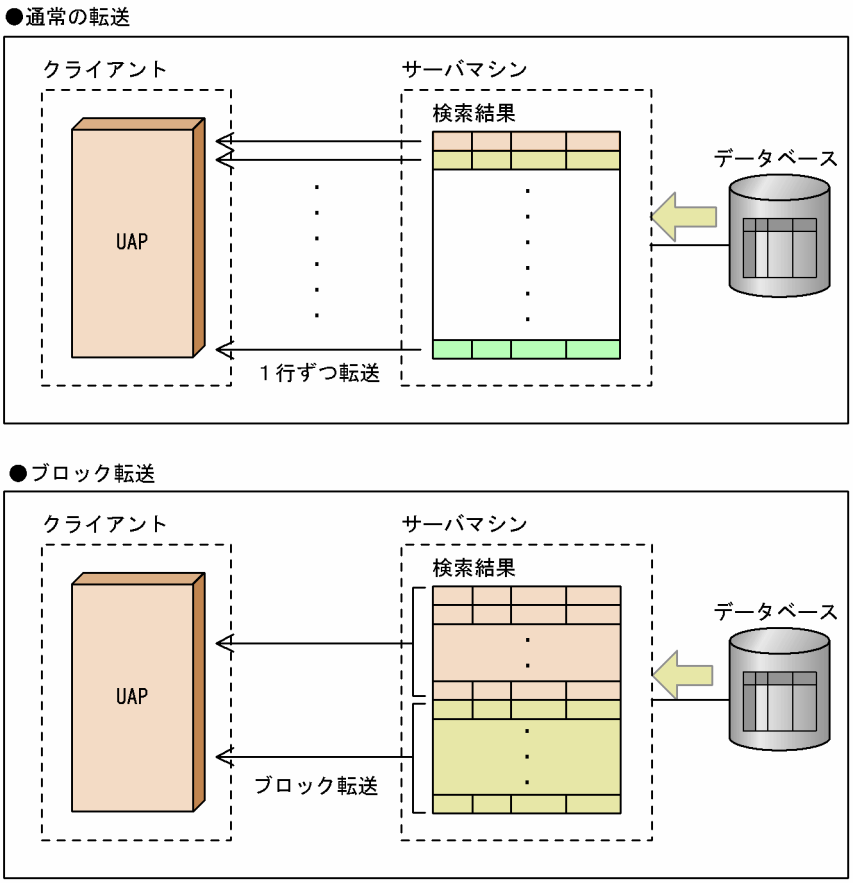
データベースをアクセスする処理性能を向上するため，HiRDB には次に示す機能があります。

- ブロック転送機能
- グループ分け高速化機能
- 配列を使用した機能
- ホールダブルカーソル
- SQL の最適化

5.11.1　ブロック転送機能

HiRDB サーバから HiRDB クライアントにデータを転送するときに，任意の行数単位で転送する機能を**ブロック転送機能**といいます。HiRDB クライアントから HiRDB サーバにアクセスし，大量のデータを検索する場合にブロック転送機能を使用すると，検索性能を向上できます。ただし，転送行数を多くすると通信のオーバヘッドが減少するので検索時間を短縮できますが，所要メモリが増加するので使用時には注意が必要です。また，転送行数を多くし過ぎると，通信で再送が発生し，再送のための時間が余分に掛かります。転送行数が多ければ多いほど再送も増えるため，転送行数が多いとブロック転送の効果がなくなります。したがって，転送行数に大き過ぎる値を指定しないようにしてください。ブロック転送機能の概要を次の図に示します。

図 5–12　ブロック転送機能の概要



ブロック転送機能の使用

ブロック転送機能は，次に示す条件を満たす場合に使用できます。

1. クライアント環境定義 PDBLKF に 2 以上の値を指定している場合，又は PDBLKBUFFSIZE に 1 以上の値を指定している場合
2. FETCH 文を指定している場合（ただし，次のどれかの条件に該当する場合を除きます）
   - カーソルを使用した更新
   - BLOB 型の選択式がある検索
   - クライアント環境定義 PDBINARYBLKF が NO（省略も含む）で，かつ定義長が 32,001 バイト以上の BINARY 型の選択式がある検索
   - BLOB 位置付け子型，又は BINARY 位置付け子型の変数を使用して結果を受け取る検索で，かつホールダブルカーソルを使用した検索

クライアント環境定義については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.11.2　グループ分け高速化機能

SQL の GROUP BY 句を指定してグループ分け処理をする場合，ソートしてからグループ分けをしています。これにハッシングを組み合わせてグループ分けすることで高速なグループ分け処理を実現する機能を

グループ分け高速化機能といいます。この機能は，グループ分けのグループ数が少なく，元の行数が多いほど，グループ分けの処理時間が短縮できます。

グループ分け高速化機能の指定

次に示すオペランドなどの SQL 最適化オプションで，グループ分け高速化機能を使用する指定をします。

- システム定義の pd_optimize_level オペランド
- クライアント環境定義の PDSQLOPTLVL
- CREATE PROCEDURE，CREATE TRIGGER，CREATE TYPE，ALTER PROCEDURE，ALTER ROUTINE，及び ALTER TRIGGER の SQL 最適化オプション

pd_optimize_level オペランドについては，マニュアル「HiRDB システム定義」を参照してください。PDSQLOPTLVL の指定とグループ分け高速化機能の指定については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.11.3　配列を使用した機能

(1)　配列を使用した FETCH 機能

FETCH 文の **INTO** 句に配列型の埋込み変数を指定すると，検索結果を一度に複数行取得できます。HiRDB クライアントから HiRDB サーバにアクセスし，大量のデータを検索する場合に配列を使用した FETCH 機能を使用すると，検索性能を向上できます。ブロック転送機能とは異なり，複数行の検索結果を取得することをプログラム内で明示的に記述します。また，配列を使用した FETCH は INTO 句に指定した埋込み変数と標識変数がすべて配列型の変数の場合に有効です。配列を使用した FETCH 機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(2)　配列を使用した INSERT 機能

INSERT 文を実行する場合，複数行分のデータを設定した配列型の変数を指定すると，一つの SQL 文で複数行分のデータを挿入できます。配列を使用した INSERT 機能を使用すると，HiRDB クライアントと HiRDB サーバとの間の通信回数を削減できます。また，HiRDB/パラレルサーバの場合には，更に HiRDB サーバ内のサーバ間の通信回数も削減できます。そのため，HiRDB クライアントから HiRDB サーバにアクセスし，大量のデータを高速に挿入したい場合に有効となります。

配列を使用した INSERT 機能の使用方法

静的に実行する場合

INSERT 文で FOR 句に埋込み変数を指定し，かつ VALUES 句に指定した埋込み変数と標識変数をすべて配列型の変数にしてください。一括して挿入する行数は，FOR 句に指定した埋込み変数で制御します。

動的に実行する場合

次に示す手順で実行してください。

1. PREPARE 文で，INSERT 文（VALUES 句のすべての挿入値に？パラメタを指定）を前処理します。
2. EXECUTE 文の USING 句に，前処理した INSERT 文の入力？パラメタに与える値を配列で指定し，かつ BY 句に埋込み変数を指定してください。一括して挿入する行数は，BY 句に指定した埋込み変数で制御します。

配列を使用した INSERT 機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(3) 配列を使用した UPDATE 機能

UPDATE 文を実行する場合，複数回分のデータを設定した配列型の変数を指定すると，一つの SQL 文で複数回分の表の列を更新できます。HiRDB クライアントと HiRDB サーバとの間の通信回数を削減できるため，HiRDB クライアントから HiRDB サーバにアクセスし，大量データを高速に更新する場合に有効です。

配列を使用した UPDATE 機能の使用方法

静的に実行する場合

UPDATE 文で，FOR 句に埋込み変数を指定し，かつ探索条件中に指定した埋込み変数と標識変数をすべて配列型の変数にしてください。一括して更新する回数は，FOR 句に指定した埋込み変数で制御します。

動的に実行する場合

次に示す手順で実行してください。

1. PREPARE 文で，UPDATE 文（更新値や探索条件中に？パラメタを指定）を前処理します。
2. EXECUTE 文の USING 句に，前処理した UPDATE 文の入力？パラメタに与える値を配列で指定し，かつ BY 句に埋込み変数を指定してください。一括して更新する回数は，BY 句に指定した埋込み変数で制御します。

配列を使用した UPDATE 機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4) 配列を使用した DELETE 機能

DELETE 文を実行する場合，複数回分のデータを設定した配列型の変数を指定すると，一つの SQL 文で複数回分の行の削除ができます。HiRDB クライアントと HiRDB サーバとの間の通信回数を削減できるため，HiRDB クライアントから HiRDB サーバにアクセスし，大量データを高速に削除する場合に有効です。

配列を使用した DELETE 機能の使用方法

静的に実行する場合

DELETE 文で，FOR 句に埋込み変数を指定し，かつ探索条件中に指定した埋込み変数と標識変数をすべて配列型の変数にしてください。一括して削除する回数は，FOR 句に指定した埋込み変数で制御します。

動的に実行する場合

次に示す手順で実行してください。

1. PREPARE 文で，DELETE 文（探索条件中に？パラメタを指定）を前処理します。

2. EXECUTE 文の USING 句に，前処理した DELETE 文の入力？パラメタに与える値を配列で指定し，かつ BY 句に埋込み変数を指定してください。一括して削除する回数は，BY 句に指定した埋込み変数で制御します。

配列を使用した DELETE 機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.11.4　ホールダブルカーソル

COMMIT 文を実行しても閉じないようにするカーソルを**ホールダブルカーソル**といいます。

Type4 JDBC ドライバでは，コミット実行後も ResultSet オブジェクトを有効にする場合に，ホールダブルカーソル機能を使用します。

ホールダブルカーソルを使用すると，大量のデータを検索又は更新する場合に，検索又は更新の途中で COMMIT 文を実行できるため，排他資源を削減できます。また，カーソルを開いたまま，COMMIT 文を実行できるため，大量のデータを検索又は更新する場合でも，シンクポイントを有効にして，再開始時の時間を短縮できます。ホールダブルカーソルについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.11.5　SQL の最適化

データベースの状態を考慮して，最も効率的なアクセスパスを決定することを **SQL の最適化**といいます。SQL の最適化には，**SQL 最適化指定**，**SQL 最適化オプション**，及び **SQL 拡張最適化オプション**があります。これらについては，次の表を参照してください。

- SQL 最適化指定の機能
- SQL 最適化オプションの機能
- SQL 拡張最適化オプションの機能

なお，SQL の最適化にはここで説明する項目以外の機能もあります。その機能は常に性能が向上するため，オプションとしていません（必ず適用されます）。SQL 最適化指定，SQL 最適化オプション，及び SQL 拡張最適化オプションについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

表 5–3　SQL 最適化指定の機能

SQL 最適化指定の機能	説明
使用インデクスの SQL 最適化指定	表の検索で使用するインデクスを指定できます。また，表の検索でインデクスを利用しないようにすることもできます（テーブルスキャン）。 使用インデクスの SQL 最適化指定は次に示す SQL で指定できます。 • 表式 • DELETE 文

SQL 最適化指定の機能	説明
	• UPDATE 文形式 1
結合方式の SQL 最適化指定	結合表に対する結合方式（ネストループジョイン，ハッシュジョイン，又はマージジョイン）を指定できます。 結合方式の SQL 最適化指定は次に示す SQL で指定できます。 • 表式
副問合せ実行方式の SQL 最適化指定	述語中の副問合せに対する副問合せの実行方式をハッシュ実行にするか，又はハッシュ実行以外にするかを指定できます。 副問合せ実行方式の SQL 最適化指定は次に示す SQL で指定できます。 • 副問合せ

表 5–4　SQL 最適化オプションの機能

SQL 最適化オプションの機能	説明
ネストループジョイン強制	結合条件の列にインデクスを定義してある場合，結合処理にネストループジョインだけを使用します。
複数の SQL オブジェクト作成	あらかじめ複数の SQL オブジェクトを作成し，実行時に埋込み変数，又は?パラメタの値によって，最適な SQL オブジェクトを選択します。
フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）	通常はデータ取り出しに使用しないバックエンドサーバをフロータブルサーバとして使用しています。この最適化方法を適用すると，データ取り出しに使用するバックエンドサーバについてもフロータブルサーバとして使用します。ただし，フロータブルサーバとして使用するバックエンドサーバ数は HiRDB が計算して求めるため，すべてのバックエンドサーバを使用するとは限りません。すべてのバックエンドサーバを使用したい場合は，「フロータブルサーバ候補数の拡大」とともに指定してください。
ネストループジョイン優先	結合条件の列にインデクスを定義してある場合，結合処理にネストループジョインを優先して使用します。
フロータブルサーバ候補数の拡大	通常使用するフロータブルサーバ数は，利用できるフロータブルサーバから必要数を HiRDB が計算して割り当てます。この最適化方法を適用すると，利用できるフロータブルサーバをすべて利用します。ただし，データ取り出しに使用するバックエンドサーバはフロータブルサーバとして使用できません。データ取り出しに使用するバックエンドサーバもフロータブルサーバとして使用したいときは，「フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）」とともに指定してください。
OR の複数インデクス利用の優先	OR の複数インデクスを利用して検索する方法を，優先して適用したい場合に指定します。OR の複数インデクス利用とは，探索条件中の OR で結ばれた複数の条件に対して，インデクスを使用してそれぞれの条件を検索し，検索結果の和集合で探索条件を評価する方式をいいます。
自バックエンドサーバでのグループ化，ORDER BY，DISTINCT 集合関数処理	グループ化，ORDER BY，及び DISTINCT 集合関数処理を，フロータブルサーバを使用しないで，表が定義されているバックエンドサーバ（自バックエンドサーバ）で処理します。
AND の複数インデクス利用の抑止	AND の複数インデクス利用をするアクセスパスを常に使わないようにします。

SQL 最適化オプションの機能	説明
	AND の複数インデクス利用とは，探索条件に AND で結ばれた条件が複数あり，それぞれの列に異なるインデクスが定義してある場合（例えば，SELECT ROW FROM T1 WHERE C1 = 100 AND C2 = 200），それぞれのインデクスを使って条件を満たす行の作業表を作成し，これらの積集合を求める方式です。
グループ分け高速化処理	SQL の GROUP BY 句で指定したグループ分けを，ハッシングを使って高速に処理します。
フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）	通常はデータ取り出しに使用しないバックエンドサーバをフロータブルサーバとして使用しています。この最適化方法を適用すると，データ取り出しに使用するバックエンドサーバだけをフロータブルサーバとして使用します。
データ収集用サーバの分離機能	「フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）」，及び「フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）」を指定した場合は，データ収集用サーバの分離機能となります。データ収集用サーバの分離機能を適用した場合，複数のバックエンドサーバから 1 か所のバックエンドサーバにデータを集める必要がある SQL に対しては，データ転送元以外のバックエンドサーバをデータ収集用に割り当てます。これ以外の用途のフロータブルサーバとしては，データ収集用以外のバックエンドサーバ（データ取り出しバックエンドサーバも含みます）を割り当てます。
インデクス利用の抑止（テーブルスキャン強制）	通常はインデクスの利用判定を HiRDB が決定します。この最適化方法を適用すると，インデクスを使用しない方式を強制的に使用するようにします。ただし，ジョインを使用してネストループ結合になる場合，構造化繰返し述語を探索条件に指定した場合，又はインデクス型プラグイン専用関数の条件の場合は，インデクス利用を抑止できません。
複数インデクス利用の強制	AND の複数インデクス利用を強制的に選択して，表を検索する場合に指定します。AND で結ばれた条件を複数指定している場合，この最適化を指定しないと AND の複数インデクス利用が選択されたときでも，通常は最大二つ程度のインデクスしか使用しません。使用するインデクスの数は，表定義，インデクス定義，及び探索条件によって多少変わります。この最適化を指定すると，インデクスによって検索範囲が絞り込める条件をすべて使用するようになります。AND の複数インデクス利用は，それぞれのインデクスを使用してある程度の件数に絞り込めます。さらに，積集合によって重なっている部分が少ない場合に有効となります。
更新 SQL の作業表作成抑止	インデクスキー値無排他を適用している場合にこの最適化を指定すると，FOR UPDATE 句を指定した検索，UPDATE 文，又は DELETE 文に対してインデクスを使用したときでも，HiRDB は内部処理のための作業表を作成しません。したがって，高速に SQL 文を処理できます。なお，インデクスキー値無排他を適用していない場合，作業表は作成されます。インデクスを使用しているかどうかについては，アクセスパス表示ユティリティで確認できます。
探索高速化条件の導出	この最適化を指定すると，探索高速化条件の導出をします。探索高速化条件とは，WHERE 句の探索条件，FROM 句の ON 探索条件から，CNF 変換，又は条件推移で新たに導出される条件のことをいいます。探索高速化条件を導出すると，検索する行が早い段階で絞り込まれるため，検索性能が向上します。ただし，探索高速化条件の生成及び実行に時間が掛かるか，又はアクセスパスが意図したとおりにならない場合があるため，できるだけこの最適化は指定しないで，探索高速化条件を直接 SQL 文中に指定するようにしてください。
スカラ演算を含むキー条件の適用	この最適化を指定すると，スカラ演算を指定した制限条件のうち，スカラ演算中に含まれる列がすべてインデクス構成列の場合に，インデクスのキー値ごとに判定して絞り込みをします。この条件はキー条件として評価されます。

SQL 最適化オプションの機能	説明
プラグイン提供関数からの一括取得機能	探索条件にプラグイン提供関数を指定し，HiRDB がプラグインインデクスを使用して検索する場合，通常，HiRDB はプラグイン提供関数からの返却結果（行位置情報と，必要に応じて受渡し値）を 1 行ごとに取得します。この最適化を適用すると，プラグイン提供関数の返却結果を複数行まとめて取得できるため，プラグイン提供関数の呼び出し回数を削減できます。
導出表の条件繰り込み機能	通常，導出表のための作業表には，導出表の列に対する探索条件に一致しない行も格納します。この最適化方法を適用すると，導出表の列に対する探索条件に一致しない行を除いた後に，導出表のための作業表を作成します。これによって，作業表の容量，及び作業表への入出力回数を削減できます。また，このようなアクセスパスを使用することによって，検索する行をより有効に絞り込めるインデクスを利用できる場合があります。なお，バージョン 08-02 以降の HiRDB を初めて導入する場合は，この機能の使用をお勧めします。

表 5–5　SQL 拡張最適化オプションの機能

SQL 拡張最適化オプションの機能	説明
コストベース最適化モード 2 の適用	コストベース最適化モード 2 を使用して最適化処理をします。
ハッシュジョイン，副問合せのハッシュ実行	結合検索の場合にハッシュジョインを適用して最適化をします。副問合せを伴った検索については，ハッシングによって副問合せを処理します。ハッシュジョイン，副問合せのハッシュ実行を適用するかどうかは，結合方式及び外への参照を考慮して決めてください。
値式に対する結合条件適用機能	値式を含む条件に対して結合条件を作成します。
スカラ演算を含む条件に対するサーチ条件適用	探索前に評価できるスカラ演算を，列に対する条件値に含まれる探索条件に対して，インデクスを使用して検索することで，実行性能が向上します。
パラメタを含む XMLEXISTS 述語への部分構造インデクスの有効化	次の二つの条件を満たしている場合に，部分構造インデクスを使用できるようにします。 • XMLEXISTS 述語の XML 問合せ変数に CAST 指定を指定し，かつ CAST 指定の値式に?パラメタ，SQL パラメタ，又は SQL 変数を指定している。 • 定義されているインデクスの部分構造指定と，XMLEXISTS 述語の XQuery 問合せ中で条件として指定した部分構造パスが一致している。
FROM 句の導出表のマージ適用	FROM 句に行の選択及び列の射影だけをするような単純な導出表を指定した SQL が，作業表を作成しないで実行できるようになります。これによって，FROM 句に導出表を指定した SQL の実行性能が向上します。 作業表を作成しない条件については，マニュアル「HiRDB SQL リファレンス」の「表参照」の「FROM 句の導出表についての規則」を参照してください。
外結合内結合変換機能	内結合を指定しても等価な結果が得られることが明らかな問合せに対して外結合を指定している場合に，外結合を内結合に変換します。 不要な処理を省略できるだけでなく，WHERE 句に指定した内表の列に対する探索条件へインデクスを適用できるようになり，実行性能の向上が期待できます。

SQL 最適化オプション，SQL 拡張最適化オプションの設定

SQL 最適化オプション及び SQL 拡張最適化オプションを設定するためには，次のどれかで指定します。

1. システム共通定義又はフロントエンドサーバ定義の pd_optimize_level オペランド，pd_additional_optimize_level オペランド
2. クライアント環境定義の PDSQLOPTLVL オペランド，PDADDITIONALOPTLVL オペランド
3. ストアドルーチン中及びトリガ中の SQL 文の SQL 最適化オプション，SQL 拡張最適化オプション

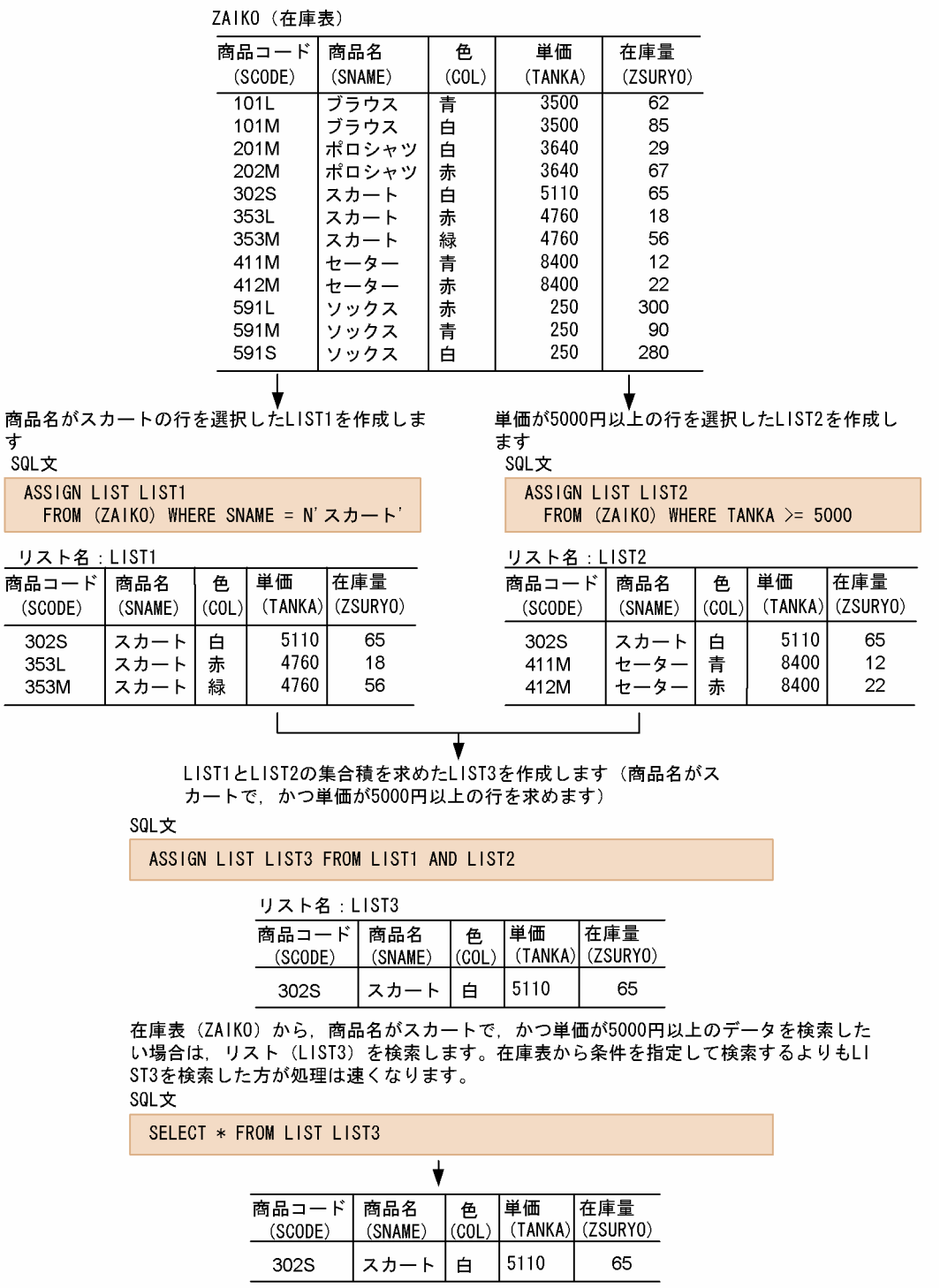
同時に指定した場合の優先順位は，3，2，1 の順となります。また，SQL 最適化指定を同時に指定した場合は，SQL 最適化オプション，SQL 拡張最適化オプションよりも SQL 最適化指定が優先されます。

5.12　絞込み検索

絞込み検索とは，段階的に対象レコードを絞り込む検索のことをいいます。絞込み検索をする場合，**ASSIGN LIST** 文で**リスト**を作成します。リストとは，適当な件数になるまで条件を指定して段階的にデータを絞り込んでいくような情報検索をするために，その途中段階のデータの集合を一時的に名前（リスト名）を付けて保存したデータ又は保存したデータの集合を意味します。

ある条件で作成したリストがあれば，そのリストを使用することで処理速度の向上が図れます。また，複数の条件を指定する場合は，複数のリストを組み合わせた検索もできます。絞込み検索については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。リストを使用した検索の例を次の図に示します。

図 5–13　リストを使用した検索の例



絞込み検索をするための準備

絞込み検索をするためには，あらかじめ次に示す準備をしておく必要があります。

・システム定義の指定

絞込み検索をする場合，システム共通定義の絞込み検索に関するオペランドを指定する必要があります。絞込み検索をする場合に必ず指定するオペランドを次に示します。

- pd_max_list_users：最大リスト作成ユーザ数を指定します。
- pd_max_list_count：1 ユーザ当たりの最大作成可能リスト数を指定します。

システム定義の絞込み検索に関するオペランドについては，マニュアル「HiRDB システム定義」を参照してください。

・リスト用 RD エリアの作成

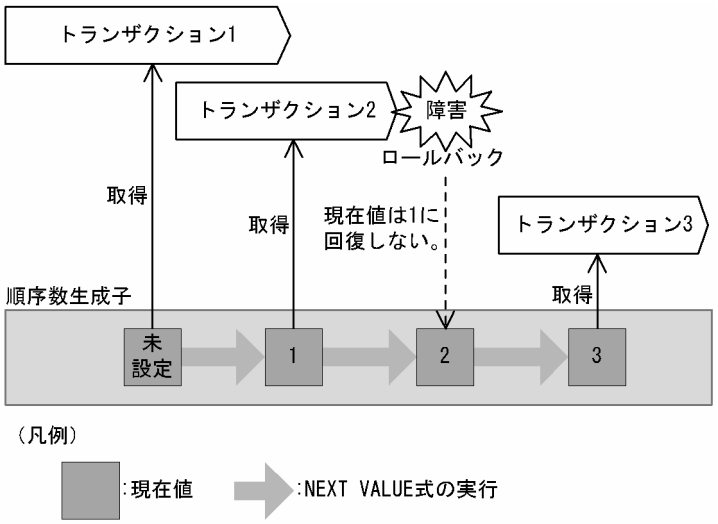
データベース初期設定ユティリティ（pdinit）又はデータベース構成変更ユティリティ（pdmod）でリスト用 RD エリアを作成します。リスト用 RD エリアについては，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

5.13　自動採番機能

自動採番機能とは，データベース中でデータを呼び出すごとに一連の整数値を返す機能です。この機能は，順序数生成子を定義することで使用できます。自動採番機能を使用すると，採番を行う UAP の開発効率が向上します。また，順序数生成子をサポートしている他 DBMS で作成した UAP からの移行性も向上します。このため，採番業務では自動採番機能を使用することを推奨します。

順序数生成子は，ユーザやトランザクションの状態にかかわらず，連続した番号（順序番号）を一度に一つ生成します。順序数生成子の概要を次の図に示します。

図 5–14　順序数生成子の概要



[説明]

順序数生成子が生成する順序番号を取得するためには，NEXT VALUE 式を使用します。NEXT VALUE 式は，順序数生成子が生成した最新の値（現在値）の次の値を取得し，現在値を次の値に更新します。

順序数生成子が定義されてから一度も NEXT VALUE 式を使用していない場合，現在値は未設定となります。現在値が未設定の状態で NEXT VALUE 式を使用すると，順序数生成子の開始値が返り，現在値には順序数生成子の開始値が格納されます。

注意事項

現在値はロールバックが発生しても回復されません。トランザクションの状態に関係なく，連続した番号を生成します。

自動採番機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.14　DB アクセス部品を使用したデータベースアクセス

データベースをアクセスする場合，DB アクセス部品も使用できます。ここでいう **DB アクセス部品**とは，次のものを指します。

- ODBC ドライバ
- HiRDB OLE DB プロバイダ
- HiRDB データプロバイダ for .NET Framework
- JDBC ドライバ
- SQLJ

これらの DB アクセス部品は，HiRDB/Run Time，及び HiRDB/Developer's Kit に含まれています。

DB アクセス部品からのアクセスについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

5.14.1　ODBC ドライバ

ODBC 対応 AP から HiRDB をアクセスする場合，**ODBC ドライバ**を使用します。ODBC 対応 AP には，Microsoft Access，Microsoft Excel などがあります。

また，ODBC 関数を使用した UAP から HiRDB をアクセスする場合も ODBC ドライバを使用します。なお，使用できる ODBC 関数は，HiRDB が提供している関数だけです。

5.14.2　HiRDB OLE DB プロバイダ

OLE DB 対応 AP から HiRDB をアクセスする場合，**HiRDB OLE DB プロバイダ**を使用します。

OLE DB は，ODBC と同様に広範囲なデータソースにアクセスするための API で，SQL データ以外のデータアクセスに適したインタフェースも定義されています。

5.14.3　HiRDB データプロバイダ for .NET Framework

ADO.NET を使用して HiRDB をアクセスする場合，**HiRDB データプロバイダ for .NET Framework** を使用します。HiRDB データプロバイダ for .NET Framework は，ADO.NET 仕様に準拠したデータプロバイダです。

HiRDB データプロバイダ for .NET Framework は，.NET Framework の System.Data 空間で提供されている共通基本インタフェース群を実装しています。また，独自の拡張機能として，配列を使用した INSERT 機能，及び繰返し列へのアクセスを実装しています。

5.14.4　JDBC ドライバ

Java で記述したプログラムから HiRDB をアクセスする場合，**JDBC ドライバ**を使用します。

5.14.5　SQLJ

SQLJ とは，Java で静的 SQL を埋込み型 SQL として記述，及び実行するための言語仕様です。SQLJ は，SQLJ トランスレータと SQLJ ランタイムライブラリから構成されます。

SQLJ トランスレータは，SQLJ ソースプログラムを解析して，Java ソースファイルと，SQL の情報を格納したプロファイルを生成します。ユーザは，Java ソースファイルを Java コンパイラでコンパイルして class ファイル（実行ファイル）を作成します。

生成されたプロファイルと class ファイルを実行する場合，**SQLJ ランタイムライブラリ**を使用します。

6 HiRDB のアーキテクチャ

この章では，HiRDB のアーキテクチャについて説明します。

6.1　HiRDB の環境設定

Windows 版 HiRDB の各環境設定方法のメリット及びデメリットを次の表に示します。それぞれの環境設定方法の詳細については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

なお，UNIX 版については，コマンドを使用する方法で環境設定を行ってください。

表 6–1　各環境設定方法のメリット及びデメリット

環境設定方法	概要	メリット	デメリット
簡易セットアップツールを使用する方法【Windows 版限定】	HiRDB の環境設定情報を，表示される画面に従って入力していきます。その入力情報を基に HiRDB の環境設定が行われます。	ほかの方法よりも手軽に HiRDB の環境設定ができます。簡易セットアップツールを使用して HiRDB の環境設定をすれば，とりあえず HiRDB を開始できるようになります。また，既存のシステム定義の指定値を変更することもできます。	HiRDB のシステム構成が簡易セットアップツールで設定できる範囲に限定されます。
コマンドを使用する方法※1	HiRDB のコマンドを使用して HiRDB の環境設定を行います。	HiRDB のシステム構成を自由に決められます。	HiRDB の環境設定をするのに，ある一定以上の知識が必要になります。具体的には，このマニュアルで説明している機能や設定内容を理解しておく必要があります。そのため，ほかの方法よりも，環境設定方法が難しくなります。
バッチファイル（SPsetup.bat）を使用する方法【Windows 版限定】※2	バッチファイルを実行して HiRDB の環境設定を行います。	コマンドを使用する方法よりも手軽に HiRDB の環境設定ができます。	HiRDB のシステム構成がバッチファイルで設定できる範囲に限定されます。

注※1【Windows 版の場合】

本番用のシステムを構築する前に簡易導入をお試しください。サンプルファイルを使ってテスト用のシステムで HiRDB の構築手順を一通り実行しておけば，本番用のシステムをより適切に構築できます。簡易導入の方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

注※2

バッチファイルを使用する方法は，HiRDB/シングルサーバのときだけ使えます。HiRDB/パラレルサーバの環境設定については，%PDDIR%¥HiRDEF¥readme.txt を参照してください。

注意事項

- 簡易セットアップツールの場合，プラグインの環境設定はできません。

- バッチファイルの場合，HiRDB の環境設定情報を自動的に設定します。設定し終わった内容を基に，HiRDB 管理者が適切な環境に変更します。

6.2　HiRDB ファイルシステム領域

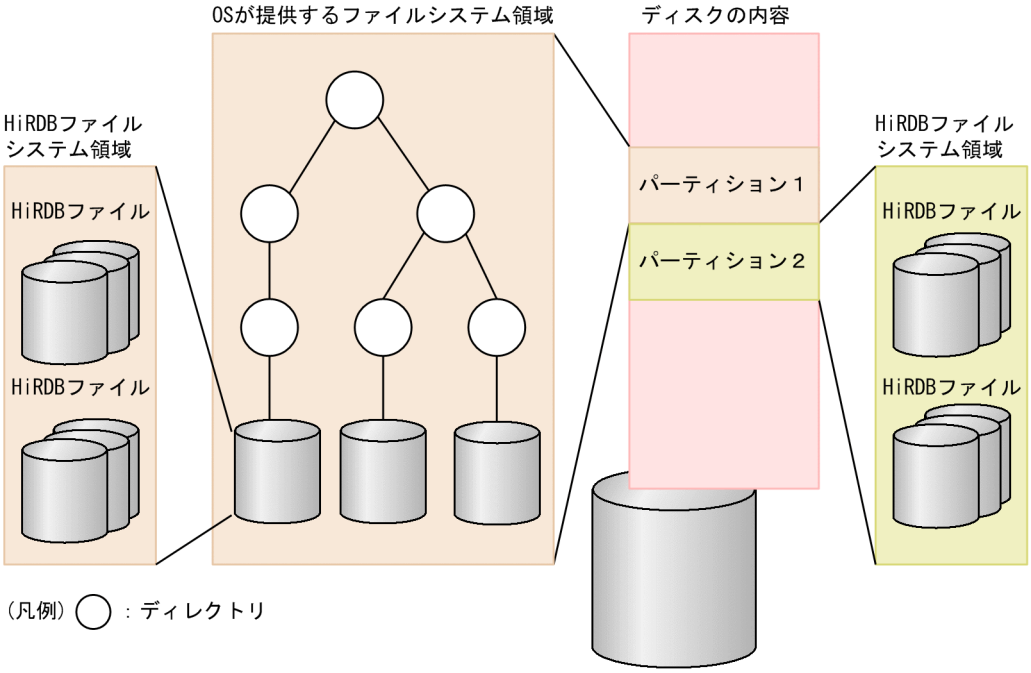
6.2.1　HiRDB ファイルシステム領域の概要

表，インデクス，障害発生時にシステムの状態を回復させるのに必要な情報など，HiRDB の様々な情報を格納するための HiRDB 専用のファイルを**HiRDB ファイル**といいます。また，HiRDB ファイルを作成する領域のことを**HiRDB ファイルシステム領域**といいます。HiRDB ファイルシステム領域は，システムファイルや RD エリアなどを構成する HiRDB 専用のファイルを作成する前に準備しておく必要があります。

(1)　HiRDB ファイルシステム領域と OS が提供するファイルシステム領域の関係

OS が入出力処理をするディスクは連続領域ごとに分割され，それぞれの領域を**パーティション**といいます。それぞれのパーティションを OS が提供するファイルシステム領域又は HiRDB ファイルシステム領域に使用できます。HiRDB ファイルシステム領域と OS が提供するファイルシステム領域の関係を次の図に示します。

図 6–1　HiRDB ファイルシステム領域と OS が提供するファイルシステム領域の関係



(2)　HiRDB ファイルシステム領域に使用するファイル

(a)　UNIX 版の場合

HiRDB ファイルシステム領域はキャラクタ型スペシャルファイル上又は通常ファイル上に作成できます。

また，Linux の場合はブロック型スペシャルファイルも使用できます。HiRDB は，ブロック型スペシャルファイルをキャラクタ型スペシャルファイルと同様に扱います。そのため，ブロック型スペシャルファイルを使用する場合は，本文中の「キャラクタ型スペシャルファイル」を「ブロック型スペシャルファイル」に読み替えてください。

通常ファイルは，カーネルバッファを経由してデータを入出力しますが，キャラクタ型スペシャルファイルは，HiRDB のバッファから直接データを入出力します。キャラクタ型スペシャルファイルの使用を基本と考えていますが，次に示す場合は通常ファイルの方が性能が優れています。

- 大量検索を主体とする表を格納する RD エリアの HiRDB ファイル
- 作業表用ファイル

ただし，通常ファイルはシステムダウンに弱いため，次に示すファイルは必ずキャラクタ型スペシャルファイルに作成してください。

- システムログファイル
- シンクポイントダンプファイル
- ステータスファイル
- システム用 RD エリアを構成する HiRDB ファイル
- 更新頻度の高いユーザ用 RD エリアを構成する HiRDB ファイル

また，系切り替え機能を使用する場合，共有ディスク装置に作成する HiRDB ファイルシステム領域にはキャラクタ型スペシャルファイルを使用してください。通常ファイルを使用すると，系切り替え発生時に更新内容が失われるおそれがあります。なお，キャラクタ型スペシャルファイルでも，プリフェッチ機能を使うことで大量検索性能を向上させることができます。

(b) Windows 版の場合

Windows のパーティション上に，HiRDB ファイルシステム領域を作成します。

通常の Windows のファイル

通常の Windows のパーティション上にファイルを作成して HiRDB ファイルシステム領域を作成します。pdfmkfs コマンドを実行して作成します。

ダイレクトディスクアクセス（raw I/O）

通常のパーティション上だけでなく，Windows のダイレクトディスクアクセス（raw I/O）を使用した HiRDB ファイルシステム領域を作成できます。この機能を **raw I/O 機能**といいます。raw I/O を使用する場合でも，パーティション又は論理ドライブをファイルと同様にアクセスできます。raw I/O 機能を使用すると，HiRDB は Windows のファイルキャッシュの動作による影響を受けなくなります。そのため，グローバルバッファの制御などによって，安定した性能を維持できるようになります。ただし，一部の HiRDB ファイルシステム領域は raw I/O を使用できません。

raw I/O 機能を使用するには，未フォーマット状態のパーティションを用意する必要があります。パーティションは Windows の［コンピュータの管理］−［ディスクの管理］で作成します。

raw I/O 機能を使用した HiRDB ファイルシステム領域の作成については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(3) HiRDB ファイルシステム領域の種類

HiRDB ファイルシステム領域は，次の表に示す種類ごとに作成することをお勧めします。各 HiRDB ファイルシステム領域の設計方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

表 6–2 HiRDB ファイルシステム領域の種類

HiRDB ファイルシステム領域の種類	オプション※	説明
RD エリア用	DB	RD エリア（リスト用 RD エリアを除く）を作成する HiRDB ファイルシステム領域です。必須です。
共用 RD エリア用	SDB	共用 RD エリアを作成する HiRDB ファイルシステム領域です。共用 RD エリアを使用する場合に必要です。
システムファイル用	SYS	システムログファイル，シンクポイントダンプファイル，及びステータスファイルを作成する HiRDB ファイルシステム領域です。必須です。
監査証跡ファイル用		監査証跡ファイルを作成する HiRDB ファイルシステム領域です。セキュリティ監査機能を使用する場合に必要です。
作業表用ファイル用	WORK	作業表用ファイルを作成する HiRDB ファイルシステム領域です。必須です。
ユティリティ用	UTL	ユティリティで使用するファイル（バックアップファイル，アンロードデータファイル，アンロードログファイル，インデクス情報ファイル，又は差分バックアップ管理ファイル）を作成する HiRDB ファイルシステム領域です。 【Windows 版限定】 Windows のキャッシュを使用しません。 <バージョン 09-50 より前の HiRDB で作成した HiRDB ファイルシステム領域について> • NUTL で作成した HiRDB ファイルシステム領域は，バージョン 09-50 以降も動作に変更はありません。 • UTL で作成した HiRDB ファイルシステム領域は，バージョン 09-50 以降は Windows のキャッシュを使用しません。
	NUTL	【Windows 版限定】 バージョン 09-50 より前の HiRDB との互換のための指定です。 NUTL を指定すると UTL を指定したものとみなします。
リスト用 RD エリア用	WORK	リスト用 RD エリアを作成する HiRDB ファイルシステム領域です。絞込み検索をするときに必要です。

注※

pdfmkfs コマンドで HiRDB ファイルシステム領域を作成するときに指定する-k オプションの指定値です。

(4)　HiRDB ファイルシステム領域の作成方法

pdfmkfs コマンドで HiRDB ファイルシステム領域を作成します。

> **参考**
>
> HiRDB の初期導入時，次に示す環境設定支援ツールを使用すると，入力した情報を基に HiRDB ファイルシステム領域が作成されます。
>
> - 簡易セットアップツール【Windows 版限定】
> - バッチファイル（SPsetup.bat）【Windows 版限定】

HiRDB ファイルシステム領域の作成方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(5)　HiRDB ファイルシステム領域の最大長

HiRDB ファイルシステム領域の最大長を次の表に示します。

表 6–3　HiRDB ファイルシステム領域の最大長

HiRDB の種類	ファイルの種類	HiRDB ファイルシステム領域の 最大長（単位：メガバイト）
AIX 版	通常ファイル（JFS）	65,411
	通常ファイル（JFS2）	1,048,575
	キャラクタ型スペシャルファイル	
Linux 版	通常ファイル	1,048,575
	キャラクタ型スペシャルファイル	
Windows 版	通常の Windows ファイル	1,048,575
	ダイレクトディスクアクセス（raw I/O）	

6.3　システムファイル

障害時にシステムの状況を回復するための情報などを格納するファイルを**システムファイル**といいます。システムファイルは次に示すファイルの総称です。

- システムログファイル
- シンクポイントダンプファイル
- ステータスファイル

6.3.1　システムログファイル

システムログを格納するファイルを**システムログファイル**といいます。**システムログ**とは，一般的にはログ（メインフレーム系ではジャーナル）と呼ばれているデータベースの更新履歴情報のことです。HiRDB はこのシステムログを次に示す目的のためにシステムログファイルに取得しています。

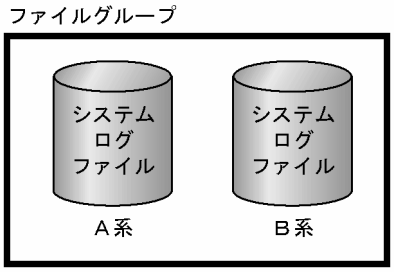
- HiRDB 又は UAP が異常終了したとき，HiRDB がデータベースを回復するのに使用します。
- HiRDB 管理者が pdrstr コマンドでデータベースを回復するときの入力情報になります。
- HiRDB 管理者が統計情報を取得するときの入力情報になります。

HiRDB 管理者は，障害発生又は統計情報の取得に備えてシステムログファイルを作成してください。

(1)　システムログファイルの構成

HiRDB はシステムログファイルを**ファイルグループ**という論理的な単位で運用します。一つのファイルグループは一つ又は二つのシステムログファイルで構成されます。二つのシステムログファイルで構成することを**システムログファイルの二重化**といい，それぞれのシステムログファイルを **A 系**，**B 系**と呼んで区別します。システムログファイルを二重化すると，HiRDB は両方の系に同じ内容のシステムログを取得します。片方のファイルに異常が発生しても，もう一方のファイルがあるため，システムの信頼性を向上できます。システムログファイルの構成を次の図に示します。

図 6–2　システムログファイルの構成



(2)　システムログファイルの作成

pdloginit コマンドでシステムログファイルを作成します。また，HiRDB システム定義の次に示すオペランドを指定してシステムログファイルを使用できる状態にしてください。

- pdlogadfg オペランド（システムログファイルのファイルグループ名を指定します）
- pdlogadpf オペランド（ファイルグループを構成するシステムログファイル名を指定します）

システムログファイルの設計及び作成方法についてはマニュアル「HiRDB システム導入・設計ガイド」を，システムログファイルの運用方法についてはマニュアル「HiRDB システム運用ガイド」を参照してください。

> **参考**
>
> HiRDB の初期導入時，次に示す環境設定支援ツールを使用すると，入力した情報を基にシステムログファイルが作成されます。また，pdlogadfg 及び pdlogadpf オペランドも設定されます。
>
> - 簡易セットアップツール【Windows 版限定】
> - バッチファイル（SPsetup.bat）【Windows 版限定】

(3)　システムログファイルの自動拡張機能

システムログファイルの容量不足が発生すると，HiRDB システム（又はユニット）が異常終了します。これを回避するため，自動的にシステムログファイルの容量を拡張する機能（**システムログファイルの自動拡張機能**）を提供しています。この機能を適用することで，システムログファイルの容量不足による HiRDB システム（又はユニット）の異常終了の頻度を低減できます。システムログファイルの自動拡張機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

6.3.2　シンクポイントダンプファイル

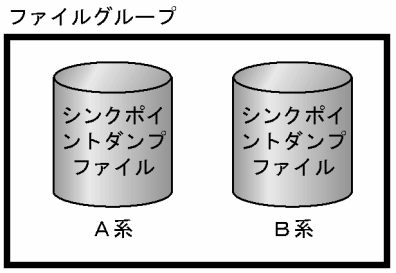
HiRDB が異常終了したときにシステムログだけで回復処理をすると，HiRDB を開始した時点からの全システムログが必要となり，システムの回復に多大な時間が掛かります。そこで，HiRDB の稼働中に一定の間隔でポイント（**シンクポイント**）を設けて，そのポイントで回復時に必要な管理情報（**シンクポイントダンプ**）を保存します。そうすると，シンクポイント以前のシステムログが不要になるため，システムの回復時間を短縮できます。

HiRDB は前回のシンクポイント以降又は HiRDB 開始以降のデータベース更新内容をシンクポイント時にデータベースに反映します。HiRDB 管理者は，障害発生に備えてシンクポイントダンプファイルを作成してください。

(1)　シンクポイントダンプファイルの構成

HiRDB はシンクポイントダンプファイルを**ファイルグループ**という論理的な単位で運用します。一つのファイルグループは一つ又は二つのシンクポイントダンプファイルで構成されます。二つのシンクポイントダンプファイルで構成することを**シンクポイントダンプファイルの二重化**といい，それぞれのシンクポイントダンプファイルを **A 系**，**B 系**と呼んで区別します。シンクポイントダンプファイルを二重化すると，HiRDB は両方の系に同じ内容のシンクポイントダンプを取得します。片方のファイルに異常が発生しても，もう一方のファイルがあるため，システムの信頼性を向上できます。シンクポイントダンプファイルの構成を次の図に示します。

図 6–3　シンクポイントダンプファイルの構成



(2)　有効保証世代数

シンクポイントダンプファイルに障害が発生して最新世代のファイルを読み込めない場合は，1 世代前のファイルを読み込みます。1 世代前のファイルも読み込めない場合は，もう 1 世代前のファイルを読み込みます。このように，ファイルを読み込めない場合は世代をさかのぼっていきます。

有効保証世代数とは，幾つ前の世代までのシンクポイントダンプファイルを上書きできない状態にするかです。例えば，有効保証世代数を 2 とした場合，2 世代前（最新世代とその一つ前の世代）までのシンクポイントダンプファイルを上書きできない状態にします。したがって，有効保証世代数を多くすれば，シンクポイントダンプファイルに障害が発生しても，有効保証世代数分のシンクポイントダンプファイルは必ず使用できるため，システムの信頼性を向上できます。

なお，シンクポイントダンプファイル数は**有効保証世代数＋ 1** 個必要になります。

(3)　シンクポイントダンプファイルの作成

pdloginit コマンドでシンクポイントダンプファイルを作成します。また，HiRDB システム定義の次に示すオペランドを指定してシンクポイントダンプファイルを使用できる状態にしてください。

- pdlogadfg オペランド（シンクポイントダンプファイルのファイルグループ名を指定します）
- pdlogadpf オペランド（ファイルグループを構成するシンクポイントダンプファイル名を指定します）

シンクポイントダンプファイルの設計及び作成方法についてはマニュアル「HiRDB システム導入・設計ガイド」を，シンクポイントダンプファイルの運用方法についてはマニュアル「HiRDB システム運用ガイド」を参照してください。

参考

HiRDB の初期導入時，次に示す環境設定支援ツールを使用すると，入力した情報を基にシンクポイントダンプファイルが作成されます。また，pdlogadfg 及び pdlogadpf オペランドも設定されます。

- 簡易セットアップツール【Windows 版限定】
- バッチファイル（SPsetup.bat）【Windows 版限定】

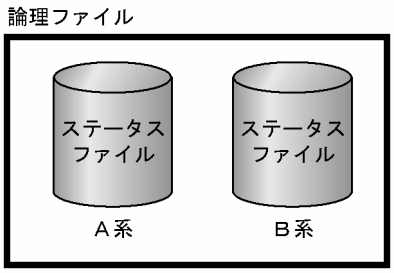
6.3.3　ステータスファイル

HiRDB がシステムを再開始するときに必要とするシステムステータス情報を格納するファイルを**ステータスファイル**といいます。ステータスファイルには，ユニット単位の再開始の情報を格納する**ユニット用ステータスファイル**及びサーバ単位の再開始の情報を格納する**サーバ用ステータスファイル**があります。HiRDB 管理者は，HiRDB を再開始する場合に備えてステータスファイルを作成してください。

(1)　ステータスファイルの構成

HiRDB はステータスファイルを**論理ファイル**という論理的な単位で運用します。一つの論理ファイルは二つのステータスファイルで構成されます。このようにステータスファイルは二重化されていて，それぞれのステータスファイルを **A 系**，**B 系**と呼んで区別します。HiRDB は，両方の系に同じシステムステータス情報を取得します。片方のファイルに異常が発生しても，もう一方のファイルがあるため，システムの信頼性を向上できます。ステータスファイルの構成を次の図に示します。

図 6–4　ステータスファイルの構成



(2)　ユニット用ステータスファイルの作成

pdstsinit コマンドでユニット用ステータスファイルを作成します。また，ユニット制御情報定義の pd_syssts_file_name オペランドを指定してユニット用ステータスファイルを使用できる状態にしてください。pd_syssts_file_name オペランドには，ステータスファイルの論理ファイル名とその論理ファイルに属するステータスファイル名を指定します。

ユニット用ステータスファイルの設計及び作成方法についてはマニュアル「HiRDB システム導入・設計ガイド」を，ユニット用ステータスファイルの運用方法についてはマニュアル「HiRDB システム運用ガイド」を参照してください。

参考

HiRDB の初期導入時，次に示す環境設定支援ツールを使用すると，入力した情報を基にユニット用ステータスファイルが作成されます。また，pd_syssts_file_name オペランドも設定されます。

- 簡易セットアップツール【Windows 版限定】
- バッチファイル（SPsetup.bat）【Windows 版限定】

(3)　サーバ用ステータスファイルの作成

pdstsinit コマンドでサーバ用ステータスファイルを作成します。また，サーバ定義の pd_sts_file_name オペランドを指定してサーバ用ステータスファイルを使用できる状態にしてください。pd_sts_file_name オペランドには，ステータスファイルの論理ファイル名とその論理ファイルに属するステータスファイル名を指定します。

サーバ用ステータスファイルの設計及び作成方法についてはマニュアル「HiRDB システム導入・設計ガイド」を，サーバ用ステータスファイルの運用方法についてはマニュアル「HiRDB システム運用ガイド」を参照してください。

参考

HiRDB の初期導入時，次に示す環境設定支援ツールを使用すると，入力した情報を基にサーバ用ステータスファイルが作成されます。また，pd_sts_file_name オペランドも設定されます。

- 簡易セットアップツール【Windows 版限定】
- バッチファイル（SPsetup.bat）【Windows 版限定】

6.3.4　システムファイルの構成単位

システムファイルの構成単位を次の表に示します。

表 6–4　システムファイルの構成単位

システムファイルの種類		作成単位	個数	二重化
HiRDB/シングルサーバの場合	システムログファイル	必須です。	2～200 グループ	△
	シンクポイントダンプファイル		2～60 グループ	△
	サーバ用ステータスファイル		1～7 個×二重化分	○
	ユニット用ステータスファイル		1～7 個×二重化分	○

システムファイルの種類		作成単位	個数	二重化
HiRDB/パラレルサーバの場合	システムログファイル	システムマネジャを除く各サーバに必要です。	2〜200 グループ（1 サーバ当たり）	△
	シンクポイントダンプファイル	システムマネジャを除く各サーバに必要です。	2〜60 グループ（1 サーバ当たり）	△
	サーバ用ステータスファイル	システムマネジャを除く各サーバに必要です。	1〜7 個×二重化分（1 サーバ当たり）	○
	ユニット用ステータスファイル	各ユニットに必要です。	1〜7 個×二重化分（1 ユニット当たり）	○

（凡例）

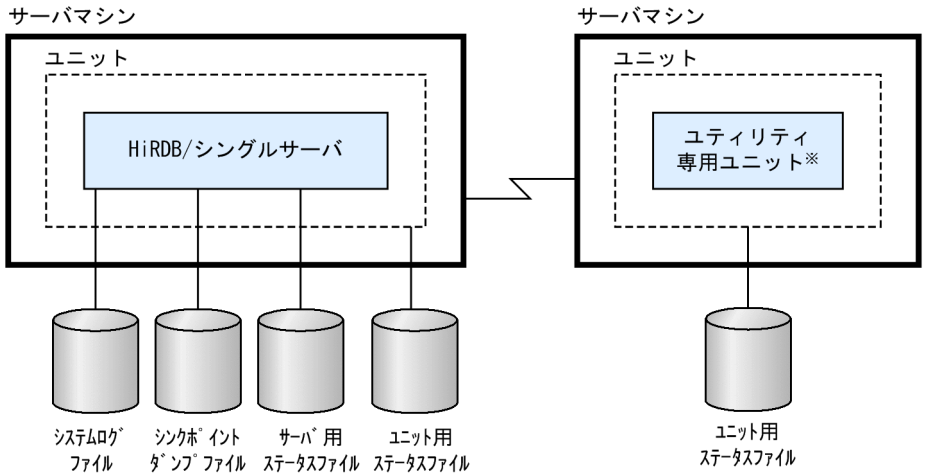
○：必ず二重化となります。

△：二重化するかどうかを選択できます。

(1)　HiRDB/シングルサーバのシステムファイルの構成

システムファイルの構成（HiRDB/シングルサーバの場合）を次の図に示します。

図 6–5　システムファイルの構成（HiRDB/シングルサーバの場合）



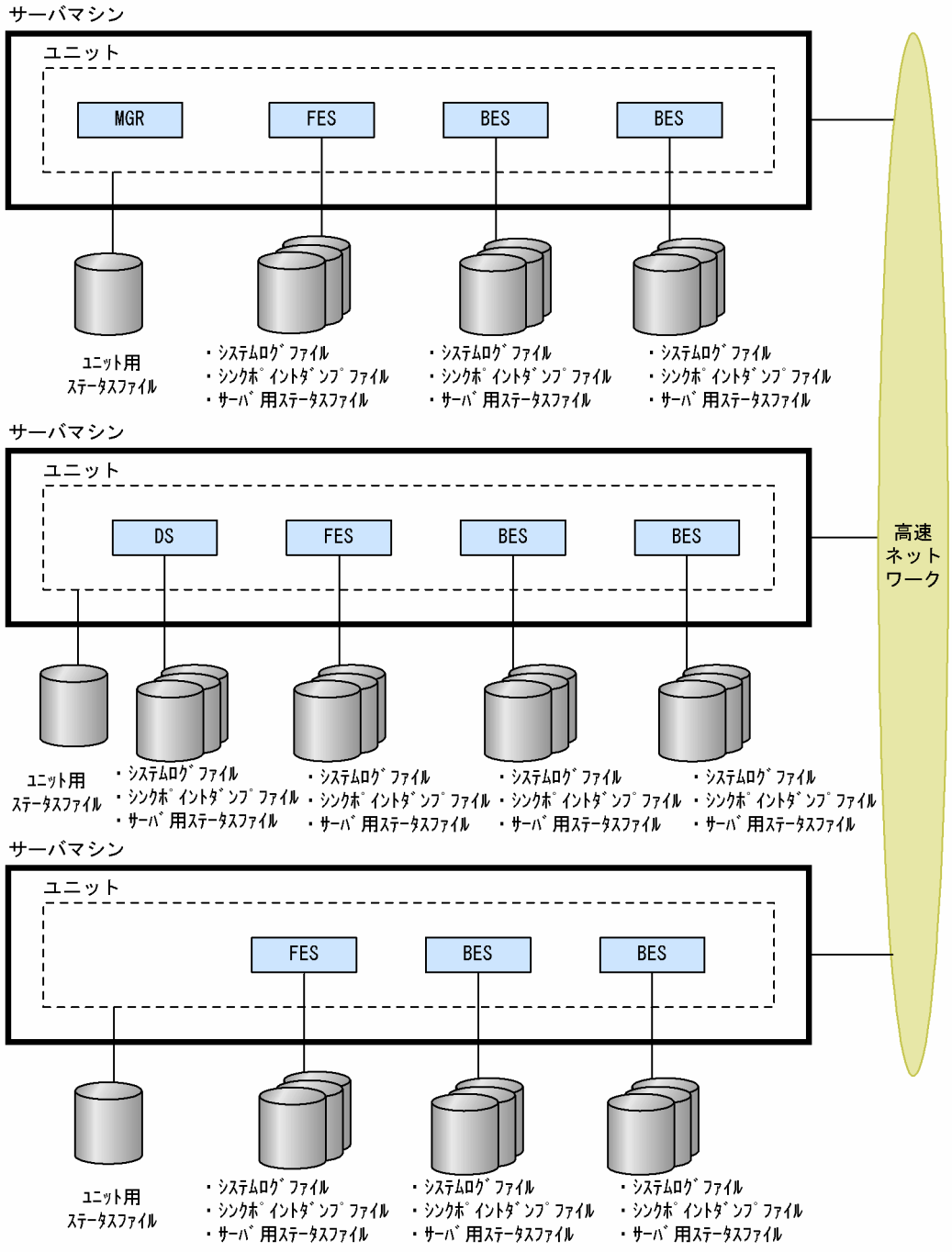
注※

ユティリティ専用ユニットは UNIX 版限定の機能です。

(2)　HiRDB/パラレルサーバのシステムファイルの構成

システムファイルの構成（HiRDB/パラレルサーバの場合）を次の図に示します。

図 6–6　システムファイルの構成（HiRDB/パラレルサーバの場合）



6.4　作業表用ファイル

6.4.1　作業表用ファイルの概要

SQL を実行するときに必要な一時的な情報を格納するためのファイルを**作業表用ファイル**といいます。作業表用ファイルは HiRDB が自動的に作成します。HiRDB 管理者は，作業表用ファイル用の HiRDB ファイルシステム領域を作成してください。

(1)　作業表用ファイルを必要とする操作

作業表用ファイルを必要とする操作を次に示します。

- SQL 文の実行
- インデクスの一括作成
- インデクスの再作成
- インデクスの再編成
- リバランスユティリティの実行

(2)　作業表用ファイルの構成

次に示すサーバに作業表用ファイルが必要になります。

- シングルサーバ
- ディクショナリサーバ
- バックエンドサーバ

(3)　作業表用ファイル用の HiRDB ファイルシステム領域の作成

pdfmkfs コマンドで作業表用ファイル用の HiRDB ファイルシステム領域を作成します。また，サーバ定義の pdwork オペランドを指定して，作業表用ファイル用の HiRDB ファイルシステム領域を使用できる状態にしてください。pdwork オペランドには，作業表用ファイル用の HiRDB ファイルシステム領域名を指定します。

作業表用ファイルの設計（サイズなど）及び作成方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

参考

HiRDB の初期導入時，次に示す環境設定支援ツールを使用すると，入力した情報を基に作業表用ファイル用の HiRDB ファイルシステム領域が作成されます。また，pdwork オペランドも設定されます。

- 簡易セットアップツール【Windows 版限定】
- バッチファイル（SPsetup.bat）【Windows 版限定】

6.5　HiRDB システム定義

HiRDB の稼働環境を HiRDB システム定義で設定します。HiRDB システム定義を格納するファイルを **HiRDB システム定義ファイル**といいます。

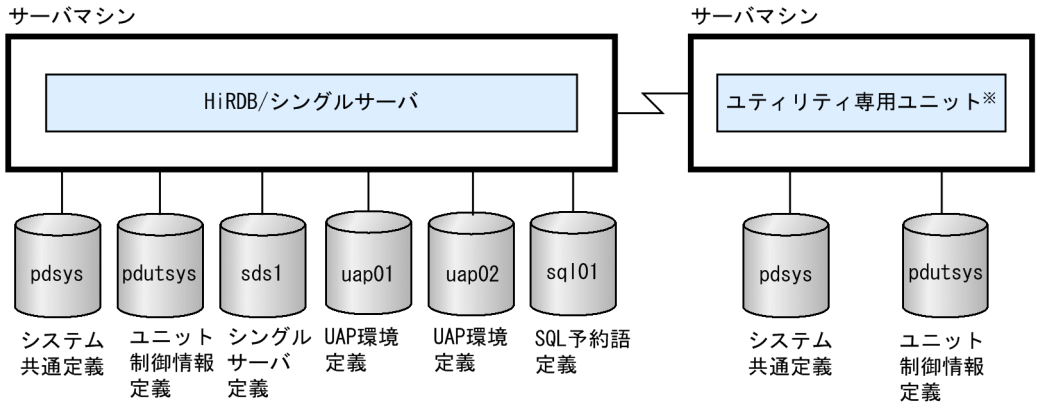
6.5.1　HiRDB/シングルサーバの HiRDB システム定義の体系

HiRDB システム定義の体系を次の表に示します。また，HiRDB システム定義ファイルの構成例を次の図に示します。

表 6–5　HiRDB システム定義の体系（HiRDB/シングルサーバの場合）

定義の種類	格納ファイル名	内容
システム共通定義	%PDDIR%¥conf¥pdsys	HiRDB の構成及び共通情報を定義します。一つの HiRDB/シングルサーバに一つ必要です。
ユニット制御情報定義	%PDDIR%¥conf¥pdutsys	ユニットの制御情報を定義します。一つのユニットに一つ必要です。
サーバ共通定義	%PDDIR%¥conf¥pdsvrc	シングルサーバ定義のオペランドのデフォルト値を定義します。必要に応じて定義してください。
シングルサーバ定義	%PDDIR%¥conf¥サーバ名	シングルサーバの実行環境を定義します。一つのシングルサーバに一つ必要です。 【UNIX 版の場合】 ユティリティ専用ユニットには定義する必要はありません。
UAP 環境定義	%PDDIR%¥conf¥pduapenv¥任意の名称	UAP の実行環境を定義します。必要に応じて定義してください。UAP 環境定義は最大 4,096 個作成できます。
SQL 予約語定義	%PDDIR%¥conf¥pdrsvwd¥任意の名称	SQL の予約語を定義します。SQL 予約語削除機能使用時に必要です。SQL 予約語削除機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

図 6–7　HiRDB システム定義ファイルの構成例（HiRDB/シングルサーバの場合）



注 ユティリティ専用ユニットにシングルサーバ定義及びUAP環境定義は必要ありません。

注※

ユティリティ専用ユニットは UNIX 版限定の機能です。

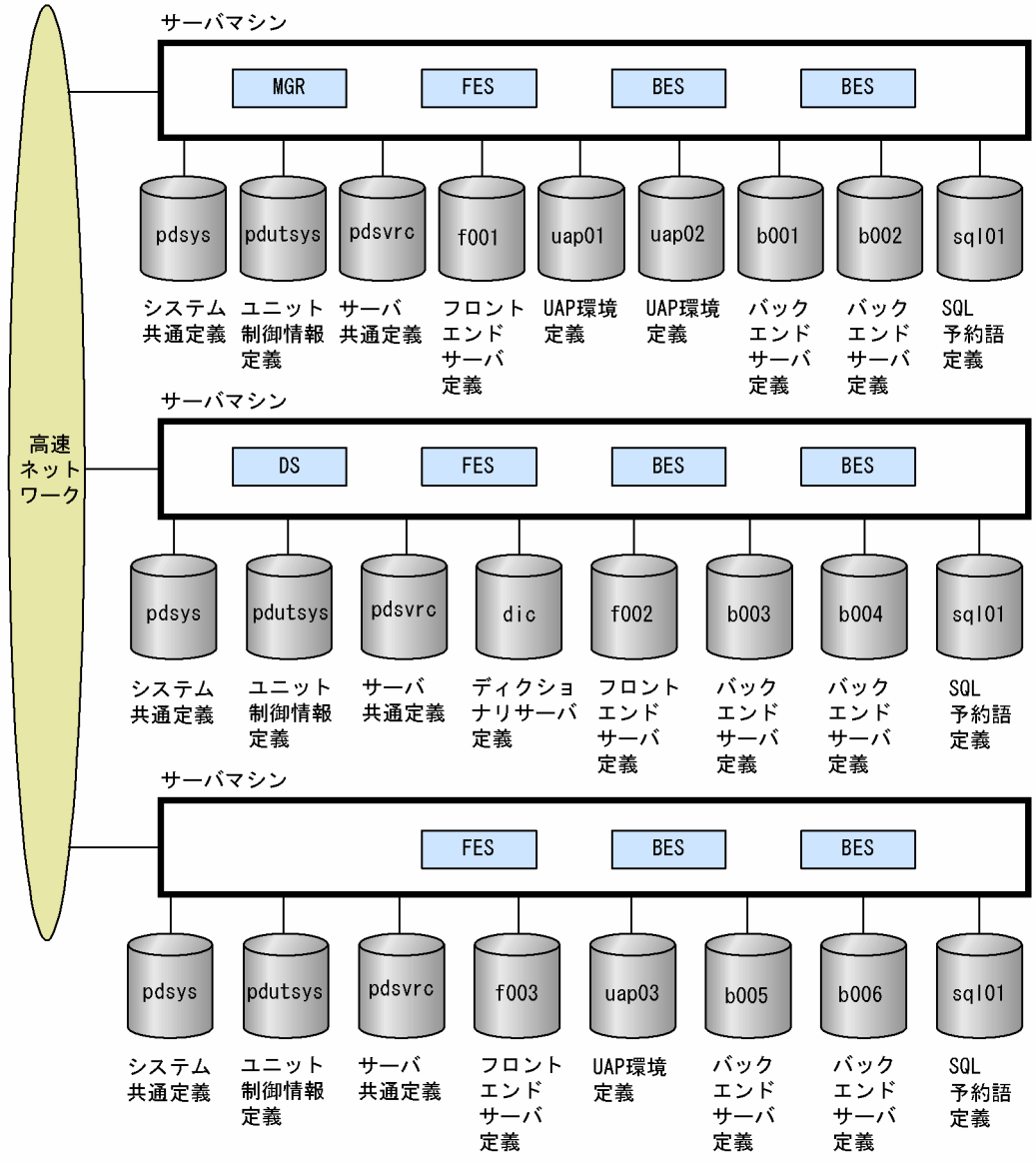
6.5.2 HiRDB/パラレルサーバの HiRDB システム定義の体系

HiRDB システム定義ファイル一覧を次の表に示します。また，HiRDB システム定義ファイルの構成例を次の図に示します。

表 6–6 HiRDB システム定義ファイル一覧（HiRDB/パラレルサーバの場合）

定義の種類	格納ファイル名	内容
システム共通定義	%PDDIR%¥conf¥pdsys	HiRDB の構成及び共通情報を定義します。一つのユニットに一つ必要です。なお，**各ユニットのシステム共通定義の内容は同一にしてください。**
ユニット制御情報定義	%PDDIR%¥conf¥pdutsys	ユニットの制御情報を定義します。一つのユニットに一つ必要です。
サーバ共通定義	%PDDIR%¥conf¥pdsvrc	各サーバ定義のオペランドのデフォルト値を定義します。必要に応じて定義してください。
フロントエンドサーバ定義	%PDDIR%¥conf¥サーバ名	フロントエンドサーバの実行環境を定義します。一つのフロントエンドサーバに一つ必要です。
ディクショナリサーバ定義	%PDDIR%¥conf¥サーバ名	ディクショナリサーバの実行環境を定義します。一つのディクショナリサーバに一つ必要です。
バックエンドサーバ定義	%PDDIR%¥conf¥サーバ名	バックエンドサーバの実行環境を定義します。一つのバックエンドサーバに一つ必要です。
UAP 環境定義	%PDDIR%¥conf¥pduapenv¥任意の名称	UAP の実行環境を定義します。必要に応じて定義してください。UAP 環境定義はフロントエンドサーバがあるユニットに作成します。マルチフロントエンドサーバの場合は，UAP 環境定義を適用したいフロントエンドサーバに定義してください。UAP 環境定義は最大 4,096 個作成できます。
SQL 予約語定義	%PDDIR%¥conf¥pdrsvwd¥任意の名称	SQL の予約語を定義します。SQL 予約語削除機能使用時に必要です。SQL 予約語削除機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

図 6–8　HiRDB システム定義ファイルの構成例（HiRDB/パラレルサーバの場合）



6.5.3　HiRDB システム定義ファイルの作成方法

HiRDB システム定義ファイルは，システム構築時に HiRDB 管理者が次に示すどれかの方法で作成します。

- **簡易セットアップツールで作成【Windows 版限定】**

  簡易セットアップツールでの作成方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

- **OS のエディタで作成**

  HiRDB システム定義中の必要なオペランドを指定して，OS のエディタで%PDDIR%¥conf 下に HiRDB システム定義ファイルを作成してください。

- **バッチファイル（SPsetup.bat）で作成【Windows 版限定】**

バッチファイルを実行すると，HiRDB システム定義ファイルが%PDDIR%¥conf 下に自動的に作成されます。

HiRDB システム定義ファイルの作成方法についてはマニュアル「HiRDB システム導入・設計ガイド」を，HiRDB システム定義の各オペランドについてはマニュアル「HiRDB システム定義」を参照してください。

HiRDB システム定義を作成した後に

pdconfchk コマンドで HiRDB システム定義の内容の整合性をチェックできます。HiRDB を開始するために必要な定義の整合性をチェックします。HiRDB システム定義を作成した後に（特に OS のエディタで HiRDB システム定義を作成した場合），pdconfchk コマンドを実行することをお勧めします。

6.5.4 システム構成変更コマンド（pdchgconf コマンド）

HiRDB システム定義（UAP 環境定義を除く）を変更する場合は HiRDB を終了する必要がありますが，システム構成変更コマンドを使用すると，HiRDB の稼働中に HiRDB システム定義を変更できます。このため，次に示すことを HiRDB の稼働中に実行できます。

- ユニットの追加，削除，及び移動
- サーバの追加，削除，及び移動
- システムファイルの追加
- グローバルバッファの追加，削除，及び変更

24 時間連続で稼働するようなシステムにはこのシステム構成変更コマンドが非常に便利です。システム構成変更コマンドの使用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

なお，システム構成変更コマンドを使用する場合は **HiRDB Advanced High Availability** が必要になります。

6.6　HiRDB の開始・終了

HiRDB の開始と終了について説明します。HiRDB を開始するときは **pdstart** コマンドを，HiRDB を終了するときは **pdstop** コマンドを実行します。そのほか，OS が起動したときに自動的に開始する方法（**自動開始**）や HiRDB/パラレルサーバのときに起動できないユニット以外のユニットだけを開始する方法（**縮退起動**）があります。

6.6.1　開始モードと終了モード

HiRDB を開始するときには**開始モード**という概念があり，終了するときには**終了モード**という概念があります。ここでは，この開始モード及び終了モードについて説明します。HiRDB の開始及び終了方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(1)　開始モード

HiRDB の開始モードを次の表に示します。

表 6–7　HiRDB の開始モード

開始モード	説明	入力するコマンド		
		システム単位の開始	ユニット単位の開始※1	サーバ単位の開始※1
正常開始	前回の終了モードが正常終了の場合の開始モードです。このとき，前回稼働時の情報を引き継ぎません。ただし，次に示す情報は引き継ぎます。 • レプリカ RD エリア（UNIX 版限定）のレプリカステータス • 障害閉塞している RD エリアの状態	pdstart	pdstart -u pdstart -x	pdstart -s
再開始	前回の終了モードが次に示す場合の開始モードです。このとき，前回稼働時の情報を引き継ぎます。 • 計画停止 • 異常終了 • 強制終了 再開始時に引き継ぐ情報の詳細は，マニュアル「HiRDB システム運用ガイド」の「HiRDB が再開始するときに引き継ぐ情報」を参照してください。			×
強制開始※2	HiRDB を再開始できないときに，強制的に HiRDB を開始する場合の開始モードです。このとき，前回稼働時の情報を引き継がないため，HiRDB はデータベースの内容を回復できません。したがって，HiRDB 管理者がデータベースの内容を回復してください。	pdstart dbdestroy	pdstart -u dbdestroy pdstart -x dbdestroy	×

（凡例）

×：この開始方法は実行できません。

注※1

HiRDB/パラレルサーバの場合，ユニット単位又はサーバ単位でも，開始及び終了できます。

注※2

HiRDB を強制開始すると，前回の HiRDB 開始後に更新したすべての RD エリア（システム用 RD エリアも含みます）が破壊されます。したがって，強制開始をする場合は，破壊された RD エリアをデータベース回復ユティリティ（pdrstr）で回復する必要があります。RD エリアを回復しないと，その後の HiRDB の動作を保証できません。このとき，RD エリアはシステムログだけで回復できます。前回の pdstart コマンドが失敗したときに出力された KFPS01262-I メッセージを参照し，メッセージに表示されているログ読み込み開始のファイルグループ名及びそれ以降に発生したシステムログをデータベース回復ユティリティ（pdrstr）の入力情報にしてください。

(2) 終了モード

HiRDB の終了モードを次の表に示します。

表 6–8 HiRDB の終了モード

終了モード	説明	入力するコマンド		
		システム単位の終了	ユニット単位の終了	サーバ単位の終了
正常終了	CONNECT 要求を禁止し，すべてのユーザの処理が終了した後に HiRDB を終了します。ユティリティが実行中のため，pdstop コマンドを実行しても HiRDB を終了できない場合は，KFPS05074-E メッセージが出力されます。ユニットを終了できない場合は，KFPS05070-E メッセージが出力されます。このとき，pdstop コマンドはリターンコード 8 で終了します。	pdstop	pdstop -u pdstop -x	pdstop -s
計画停止	トランザクションの受け付けを禁止し，ユティリティを含むすべてのトランザクションが終了した後に HiRDB を終了します。	pdstop -P	×	×
強制終了	処理中のトランザクションの完了を待たないで，HiRDB を直ちに終了します。処理中のトランザクションは，再開始時にロールバックの対象※となります。	pdstop -f	pdstop -f -u pdstop -f -x	×
異常終了	何らかの異常によって HiRDB が終了する場合の終了モードです。処理中のトランザクションの完了を待たないで，HiRDB は直ちに終了します。処理中のトランザクションは，再開始時にロールバックの対象※となります。	−	−	−

（凡例）

−：該当しません。

×：この終了方法は実行できません。

注※

処理中のトランザクションは，再開始時にロールバックの対象となります。ただし，次に示す場合のトランザクションはロールバックの対象となりません。

- データベース作成ユティリティ（pdload コマンド）又はデータベース再編成ユティリティ（pdrorg コマンド）をログレスモードで実行している場合
- ログレスモードで UAP を実行している場合

したがって，HiRDB を再開始した後に，HiRDB 管理者が RD エリアをバックアップから回復するか，ユティリティを再実行する必要があります。ログレスモード及びログレスモードで運用しているときの RD エリアの回復方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

6.6.2 HiRDB の自動開始

OS を起動すると，HiRDB が自動的に開始する方法を**自動開始**といいます。OS の起動後に pdstart コマンドで HiRDB を開始する方法を**手動開始**といいます。どちらの開始方法にするかは **pd_mode_conf** オペランドで指定します。

自動開始を選択しておくと，HiRDB（ユニット）が異常終了した場合も，自動的に HiRDB（ユニット）が再開始されます。ただし，再開始時に 3 回連続して異常終了すると，自動的に再開始しなくなります。

参考

- pd_mode_conf に AUTO や MANUAL1 を指定した場合でも，HiRDB の開始処理中又は終了処理中に HiRDB が異常終了したときは，次回の開始は必ず手動開始となります。

  異常終了時のメッセージを確認して対処したあと，手動で HiRDB を開始してください。
- 自動開始の場合，pdstart コマンドの引数（オプション）を指定する開始モード（強制開始，ユニット単位の開始，サーバ単位の開始など）は指定できません。

6.6.3 縮退起動（HiRDB/パラレルサーバ限定）

障害などで起動できないユニットがある場合，縮退起動機能を使用すると，残りの正常なユニットだけで HiRDB を開始できます。これを**縮退起動**といいます。縮退起動をするには次に示すオペランドを指定します。

- pd_start_level
- pd_start_skip_unit

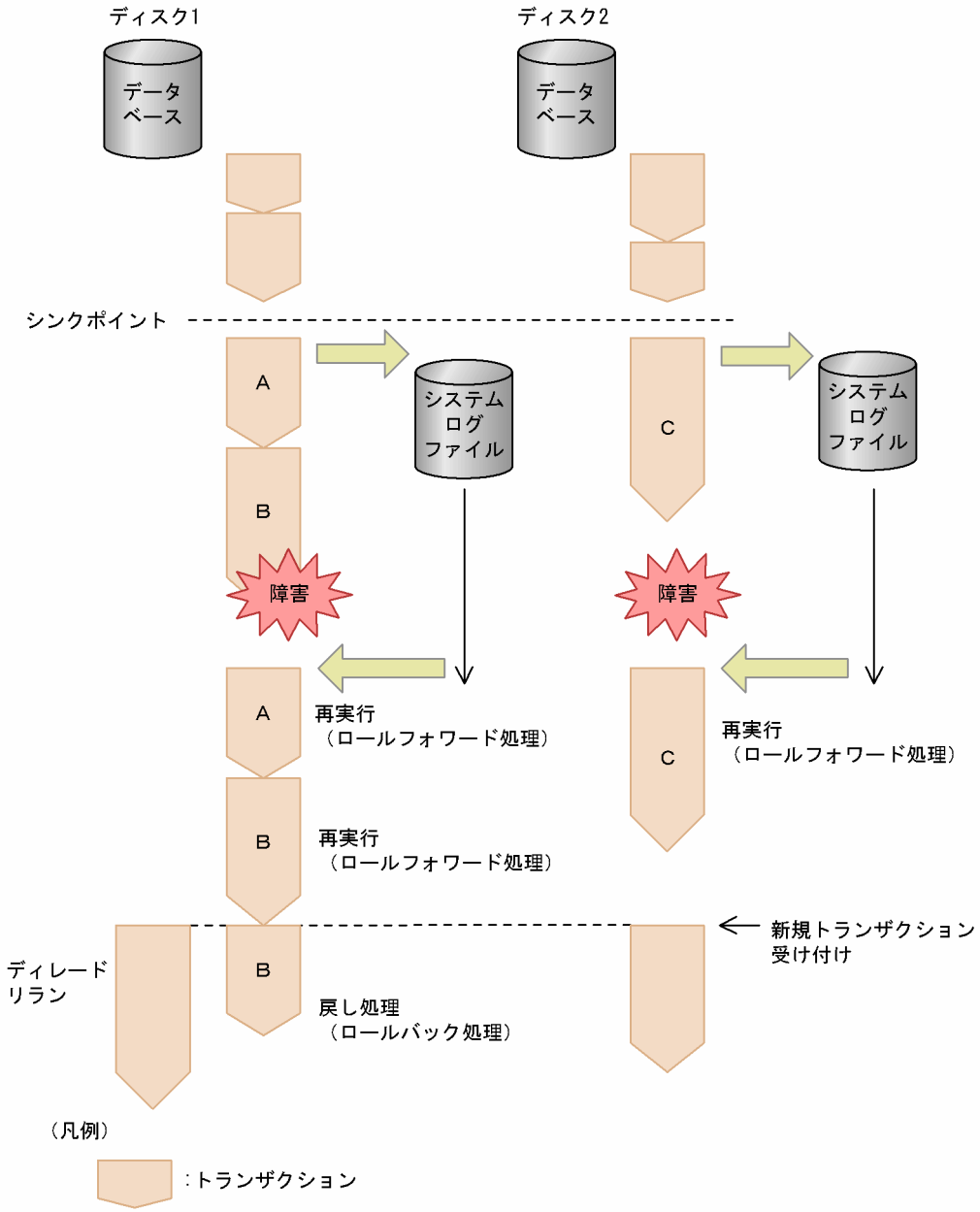
縮退起動の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

6.7　ディレードリラン

HiRDB では，ディレードリランという回復処理によって，システム障害が発生したときの，トランザクション処理の停止時間を短縮します。データベースに障害が発生した場合，HiRDB はシステムログファイルを利用して，直前のシンクポイントから障害発生時点までの更新処理を再実行します。このとき，障害発生時にデータベースを更新中だったトランザクションは，いったん障害発生時点まで再実行（**ロールフォワード処理**）され，その後データベースはその更新処理の実行前の状態に戻されます（**ロールバック処理**）。

HiRDB は，障害発生時に更新中だったディスク内のデータに対してロールバック処理を実行しますが，ロールバック処理の対象外のデータに対しては，新規のトランザクションを受け付けます。これによって，システムとしてのトランザクション処理の停止時間を短縮します。ディレードリランの概念を次の図に示します。

図 6–9　ディレードリランの概念



〔説明〕

ディスク 1 のデータベースに対するトランザクション B の途中及びディスク 2 に対するトランザクション C の終了後に，障害が発生しています。そのため，直前のシンクポイント以降に実行されたトランザクション A，B，C がロールフォワードされています。

その後，障害発生時に実行の途中だったトランザクション B はロールバックされています。また，ディスク 2 のデータベースに対するトランザクションはロールバック対象外であるため，新規のトランザクションを受け付けています。

6.8　データベースのアクセス処理方式

HiRDB はデータベースの入出力処理にグローバルバッファを使用しています。ここでは，グローバルバッファを使用した HiRDB のデータベースのアクセス処理方式について説明します。データベースのアクセス処理方式の性能を向上するための機能を次に示します。

- グローバルバッファ
- インメモリデータ処理
- プリフェッチ機能
- 非同期 READ 機能
- デファードライト処理
- デファードライト処理の並列 WRITE 機能の指定
- コミット時反映処理
- グローバルバッファの LRU 管理方式
- スナップショット方式によるページアクセス
- グローバルバッファの先読み入力
- ローカルバッファ
- BLOB データのファイル出力機能
- BLOB データ，BINARY データの部分的な更新・検索
- 位置付け子機能

6.8.1　グローバルバッファ

グローバルバッファとは，ディスク上の RD エリアに格納されているデータを入出力するためのバッファのことで，共用メモリ上に確保されます。データを更新するためにバッファ上で更新され，データベースには未反映のバッファを**更新バッファ**といいます。また，データを参照するためのバッファ，及びデータベースに反映済みのバッファを**参照バッファ**といいます。

データを格納する RD エリア又はインデクスには，必ずグローバルバッファを割り当てます。グローバルバッファの種類を次の表に示します。

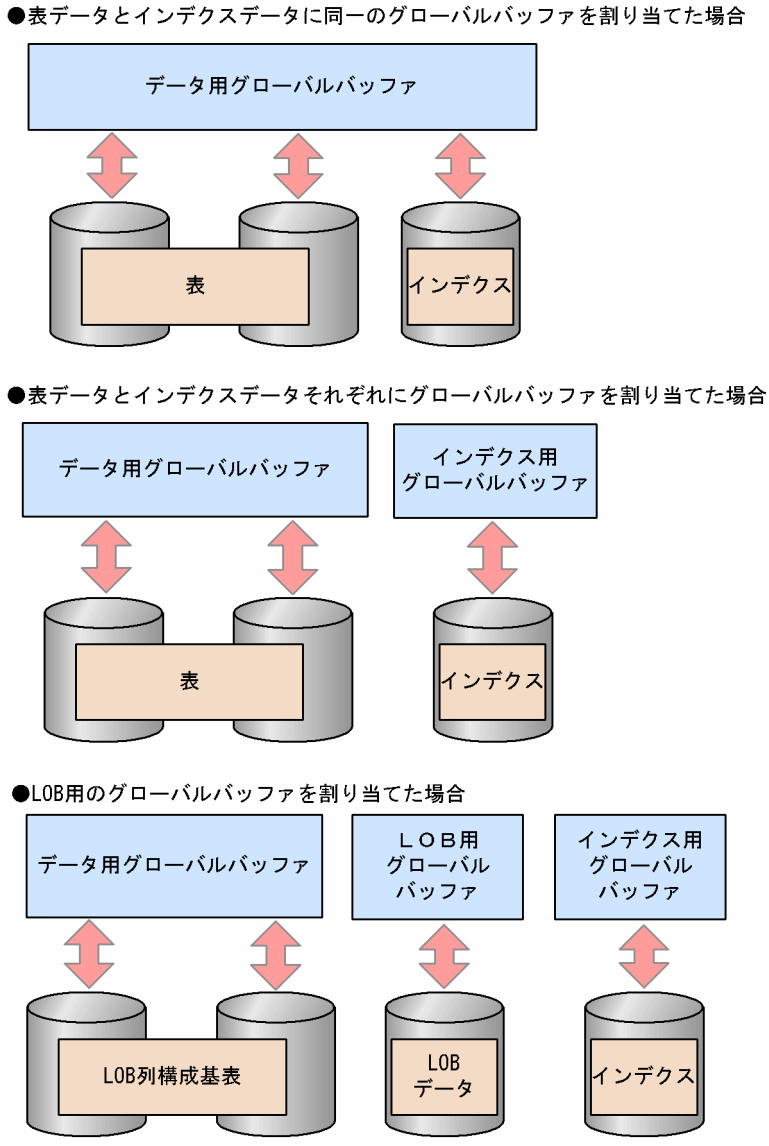
表 6–9　グローバルバッファの種類

グローバルバッファの種類	説明
データ用グローバルバッファ	データの入出力に使用されるグローバルバッファです。データ用グローバルバッファは RD エリア単位に割り当てます。
インデクス用グローバルバッファ	インデクスデータの入出力に使用されるグローバルバッファです。インデクス用グローバルバッファはインデクス単位に割り当てます。

グローバルバッファの種類	説明
LOB 用グローバルバッファ	LOB 属性のデータの入出力に使用されるグローバルバッファです。LOB 用グローバルバッファは LOB 用 RD エリア単位に割り当てます。

これらのグローバルバッファを組み合わせると性能を向上できます。例えば，データ用グローバルバッファとインデクス用グローバルバッファを分けて定義すると，データの検索とインデクスの検索を同時に実行しても互いが独立して動作します。そのため，大量データの全件検索時などでもインデクスの入出力回数を削減でき，処理時間を短縮できます。グローバルバッファの概念を次の図に示します。

図 6–10　グローバルバッファの概念



(1)　グローバルバッファを割り当てる単位

- すべての RD エリアにデータ用グローバルバッファを割り当てる必要があります。一つのグローバルバッファには，一つ以上の RD エリアを割り当ててください。
- 必要に応じてインデクスをインデクス用グローバルバッファに割り当ててください。

- 必要に応じて LOB 用 RD エリアを LOB 用グローバルバッファに割り当ててください。
- システム用 RD エリアにもグローバルバッファを割り当ててください。
- グローバルバッファの設計方針（グローバルバッファをどのように RD エリアに割り当てるか）については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(2) グローバルバッファの割り当て方法

pdbuffer オペランドでグローバルバッファを割り当てます。pdbuffer オペランドでのグローバルバッファの割り当て例を次に示します。

（例）

```
pdbuffer -a gbuf01 -r RDAREA01,RDAREA02 -n 1000       1
pdbuffer -a gbuf01 -r LOBAREA01 -n 1000               2
pdbuffer -a gbuf01 -i USER01.INDX01 -n 1000           3
pdbuffer -a gbuf01 -b LOBAREA01 -n 1000               4
```

〔説明〕

1. RD エリア（RDAREA01，RDAREA02）にデータ用グローバルバッファを割り当てます。
2. LOB 用 RD エリア（LOBAREA01）にデータ用グローバルバッファを割り当てます。
3. インデクス（USER01.INDX01）にインデクス用グローバルバッファを割り当てます。
4. LOB 用 RD エリア（LOBAREA01）に LOB 用グローバルバッファを割り当てます。

pdbuffer オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

(3) グローバルバッファの動的変更

グローバルバッファを追加，変更，又は削除するには pdbuffer オペランドを変更する必要があるため，HiRDB を一度停止する必要があります。しかし，HiRDB Advanced High Availability を導入すると，HiRDB の稼働中に pdbufmod コマンドでグローバルバッファを追加，変更，又は削除できます。これを**グローバルバッファの動的変更**といいます。例えば，次に示す場合にグローバルバッファを動的変更してください。

- 追加した RD エリアにグローバルバッファを割り当てる場合
- RD エリアの割り当て先グローバルバッファを変更する場合
- グローバルバッファのチューニングの結果，グローバルバッファの定義を変更する場合

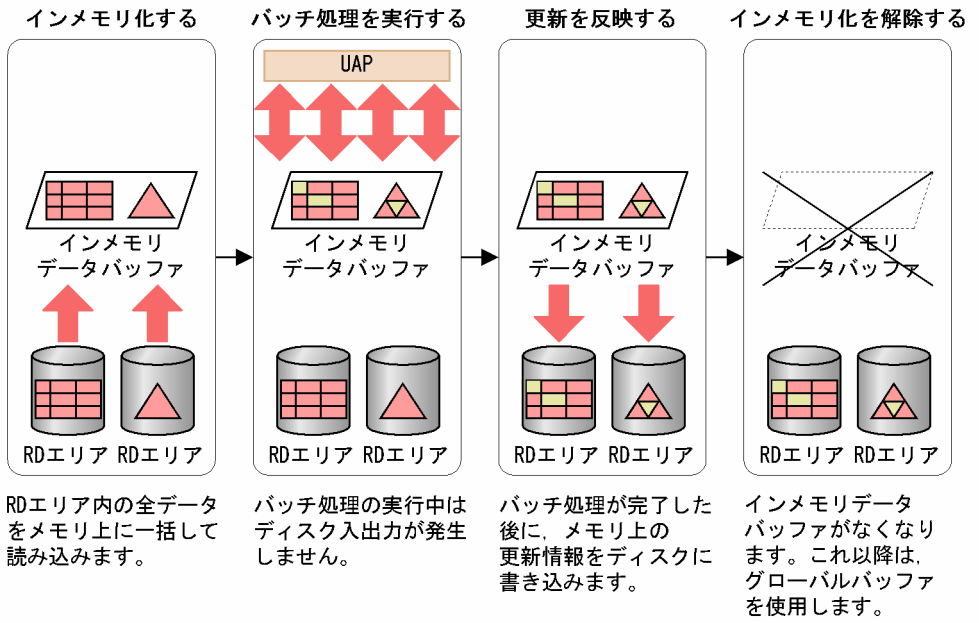
グローバルバッファの動的変更については，マニュアル「HiRDB システム運用ガイド」を参照してください。

6.8.2　インメモリデータ処理

インメモリデータ処理を適用するとバッチ処理の高速化を実現できます。インメモリデータ処理では，RD エリア内の全データをメモリ上に一括して読み込み，バッチ処理の実行中はメモリ上のデータだけを更新し，ディスク上のデータは更新しません。バッチ処理が完了した後にメモリ上の更新データを一括してディスクに書き込みます。バッチ処理中はディスク入出力が発生しません。ディスク入出力が発生するのは，RD エリア内のデータをメモリ上に読み込むときと，メモリ上の更新データをディスクに書き込むときの 2 回だけです。

インメモリデータ処理の概要を次の図に示します。

図 6–11　インメモリデータ処理の概要



[説明]

RD エリア内の全データをメモリ上に一括して読み込むことを**インメモリ化**といいます。インメモリ化は RD エリアごとに行います。インメモリ化している RD エリアを**インメモリ RD エリア**といいます。また，インメモリデータ処理で使用するデータバッファを**インメモリデータバッファ**といいます。

ポイント

インメモリデータ処理を実行する場合は，グローバルバッファを使用しないでインメモリデータバッファを使用します。インメモリデータバッファは，HiRDB が動的に共用メモリ上に確保するため，HiRDB 管理者がインメモリデータバッファを定義する必要はありません。

バッチ処理の実行中はインメモリデータバッファ上のデータだけが更新されて，ディスク上のデータは更新されません。バッチ処理が完了した後にコマンド（pdhold コマンド）を実行して，インメモリデータバッファ上の更新情報を一括してディスクに書き込みます。シンクポイント時もディスクへの書き込みが発生しません。

なお，インメモリデータ処理を適用するためには，HiRDB Accelerator が必要となります。

インメモリデータ処理の適用方法については，マニュアル「HiRDB バッチ高速化機能」を参照してください。

(1) グローバルバッファと比較した場合の利点

グローバルバッファを使用している場合はディスク入出力が定期的に発生しますが，インメモリデータバッファを使用している場合は，ディスク入出力が発生するのはインメモリ化時とディスクへのデータ書き込み時の 2 回だけです。シンクポイント時にもディスクへの書き込みが発生しません。そのため，インメモリデータバッファを使用する方が，グローバルバッファを使用したときに比べてディスク入出力回数が少なくなります。

そのほかにも，グローバルバッファと比べて次の利点があります。

- バッファの管理処理が少ない
- チューニングが不要

詳細については，マニュアル「HiRDB バッチ高速化機能」を参照してください。

(2) インメモリデータ処理の適用基準

次の場合にインメモリデータ処理を適用すると効果が期待できます。

- **処理時間の長いバッチ処理を実行している場合**

  大量のデータ更新を行う，処理時間の長いバッチ処理にインメモリデータ処理を適用すると，効果が期待できます。また，ログレスモードを適用すると，更に処理時間を短縮できます。

  バッチ処理に掛かる時間が長いほど，ディスク入出力回数に差が出るため，インメモリデータ処理を適用したときの効果が期待できます。

- **複数のバッチ処理を連続して実行している場合**

  複数のバッチ処理を連続して実行する場合にインメモリデータ処理を適用すると効果が期待できます。

  バッチ処理を連続して行うほど，ディスク入出力回数に差が出るため，インメモリデータ処理を適用したときの効果が期待できます。

- **グローバルバッファの排他競合が原因で性能が上がらない場合**

  グローバルバッファの排他競合が原因で性能が上がらない場合に，インメモリデータ処理を適用すると性能向上が期待できます。

  グローバルバッファを使用する場合（インメモリデータ処理を使用しない場合），参照又は更新処理の発生時，グローバルバッファ上にデータがキャッシュされているかどうかを検索する処理（キャッシュサーチ処理）が行われます。このとき，グローバルバッファの全ページに対して排他が掛かるため，ほかの参照又は更新処理は排他待ちとなります。インメモリデータ処理の場合は，インメモリデータバッファ上に全データがキャッシュされているため，キャッシュサーチ処理が発生しません。グローバルバッファの全ページに対する排他制御がなくなるため，同時実行性の向上が見込まれ，その分性能向上が期待できます。例えば，同じグローバルバッファを割り当てている複数の RD エリアを更新するバッ

チ処理などで排他競合が多発している場合に，インメモリデータ処理を適用すると性能向上が期待できます。

なお，排他競合の発生頻度については，統計解析ユティリティのグローバルバッファに関する統計情報で確認できます。

- **メインフレームからデータを移行する場合**

  メインフレームの膨大なデータを HiRDB に移行する場合（データを HiRDB のデータベースにデータロードする場合）に，インメモリデータ処理を適用すると効果が期待できます。

6.8.3 プリフェッチ機能

ディスク上の表データはグローバルバッファ（又はローカルバッファ）上に 1 ページごとに読み込まれます。この読み込みを 1 ページではなく，複数のページを一括して読み込むこともできます。これを**プリフェッチ機能**といいます。プリフェッチ機能を使用すると，ディスク上の表データとグローバルバッファ（又はローカルバッファ）との間の入出力回数を削減できます。

(1) プリフェッチ機能の適用基準

プリフェッチ機能は次に示す条件を満たすほど有効になります。

1. アクセスデータが raw I/O 機能を適用した HiRDB ファイルシステム領域上（UNIX 版の場合はキャラクタ型スペシャルファイル上）にある
2. 大量検索を行う
3. 次に示す SQL 文を実行する

- インデクスを使用しない SELECT，UPDATE，及び DELETE 文
- インデクス又はクラスタキーを使用した昇順検索※を行う SELECT，UPDATE，又は DELETE 文（ただし，=条件，IN 条件を除く）

注※　複数列インデクスの場合はインデクス定義で指定した順序になります。

(2) プリフェッチ機能の指定

グローバルバッファの場合

プリフェッチ機能を使用するには **pdbuffer** オペランドの**-m** オプションに 1 以上を指定します。また，一括して入力するページ数は，**pdbuffer** オペランドの**-p** オプションに指定します。pdbuffer オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

ローカルバッファの場合

プリフェッチ機能を使用するには，**pdlbuffer** オペランドの**-p** オプションに一括して入力するページ数を指定します。pdlbuffer オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

6.8.4　非同期 READ 機能

プリフェッチ機能を使用してグローバルバッファ上に複数のページを一括入力するとき，一括入力用のバッファにサーバプロセスから同期処理で一括入力して先読みをしています。**非同期 READ 機能**とは，プリフェッチ機能使用時に一括入力用のバッファを 2 面用意し，DB 処理が一つのバッファを使用中に DB 処理とは非同期に非同期 READ プロセスがもう一つのバッファに先読み入力をする機能です。DB 処理と先読み入力を同時に実行させることで処理時間を短縮できます。また，HiRDB/パラレルサーバの場合は入出力待ち時間にスレッドを切り替えて処理することで入出力待ち時間を削減できます。

なお，非同期 READ 機能はローカルバッファには使用できません。また，SCHEDULE 属性の RD エリアには適用されません。プリフェッチ機能で動作します。

非同期 READ 機能の設定

pd_max_ard_process オペランドで非同期 READ プロセス数を指定します。このオペランドに 0 を指定するか，又は省略した場合，非同期 READ 機能は動作しません。また，プリフェッチ機能の指定（pdbuffer オペランドの-m オプションに 1 以上を指定）をしている必要があります。

6.8.5　デファードライト処理

デファードライト処理とは，グローバルバッファ上で更新されたページを COMMIT 文が発行されてもディスクに書き込まないで，更新ページ数がある一定の値に達した時点でディスクに書き込む処理のことです。なお，更新ページ数がある一定の値（HiRDB が決定する値）に達した時点を**デファードライトトリガ**といいます。

ディスクに書き込む更新ページの数は **pdbuffer** オペランドの**-w** オプションで指定した，デファードライトトリガでの更新ページの出力比率を基に HiRDB が決定します。デファードライト処理を設定すると，COMMIT 文が発行されてもディスクに書き込まないため，入出力処理の負荷が軽減されます。

デファードライト処理の使用方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

デファードライト処理の設定

デファードライト処理の使用は **pd_dbsync_point** オペランド及び **pdbuffer** オペランドで指定します。pd_dbsync_point オペランド及び pdbuffer オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

6.8.6　デファードライト処理の並列 WRITE 機能

デファードライト処理の並列 WRITE 機能とは，デファードライトの書き込み処理を複数のプロセスで並列に実行する機能です。デファードライト処理の並列 WRITE 機能を使用すると，書き込み処理を複数のプロセスで実行するため，ディスクへの書き込み時間が短縮されます。デファードライト処理の並列 WRITE 機能については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

デファードライト処理の並列 WRITE 機能の指定

pd_dfw_awt_process オペランドに書き込み処理をするデファードライト処理用並列 WRITE プロセス数を指定します。また，デファードライトトリガの要求比率を **pd_dbbuff_rate_updpage** オペランドに指定します。なお，**pd_dfw_awt_process** オペランドの記述がない場合，デファードライト処理の並列 WRITE 機能は無効になります。

指定上の考慮点

デファードライト処理の並列 WRITE 機能を指定すると，プロセス数が増加するため，CPU 利用率が上がります。

6.8.7　コミット時反映処理

グローバルバッファ上で更新されたページは，COMMIT 文発行時にディスク上に書き込まれます。この処理を**コミット時反映処理**といいます。コミット時反映処理を設定すると，データベースの回復処理時にシンクポイント時点からデータベースを回復する必要がないため，データベースの回復処理時間が短縮できます。コミット時反映処理の設定方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

コミット時反映処理の設定

コミット時反映処理を使用するには **pd_dbsync_point** オペランドに **commit** を指定します。
pd_dbsync_point オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

6.8.8　グローバルバッファの LRU 管理方式

HiRDB では業務の種類（オンライン業務又はバッチ業務）に応じて，グローバルバッファの **LRU 管理方式**を選択できます。LRU の管理方式には，次に示す 2 種類があります。

(1)　参照バッファ及び更新バッファの独立した LRU での管理

参照バッファ及び更新バッファをそれぞれ独立した LRU で管理します。グローバルバッファの不足時には，グローバルバッファ内のアクセスした参照バッファの中で，最も古いバッファがメモリから追い出されます。オンライン業務のように 1 トランザクション当たりの参照，更新件数が比較的少ない場合，参照バッファ及び更新バッファをそれぞれ独立した LRU で管理すると，処理性能が向上します。

グローバルバッファを独立した LRU で管理する場合は，**pd_dbbuff_lru_option** オペランドに **SEPARATE** を指定します。pd_dbbuff_lru_option オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

(2)　グローバルバッファの一括した LRU での管理

グローバルバッファを一括した LRU で管理します。グローバルバッファの不足時には，グローバルバッファ内のアクセスしたバッファで，最も古いバッファがメモリから追い出されます。オンライン業務とバッ

チ業務の組み合わせなど，大量検索，大量更新が共存する場合，グローバルバッファを一括した LRU で管理すると，処理性能が向上します。

グローバルバッファを一括した LRU で管理する場合は，次に示すどちらかの指定をします。

- **pd_dbbuff_lru_option** オペランドに **MIX**（省略値）を指定します。
- **pdbuffer** オペランドの**-w** オプションに，デファードライトトリガでの更新ページの出力比率を指定します。

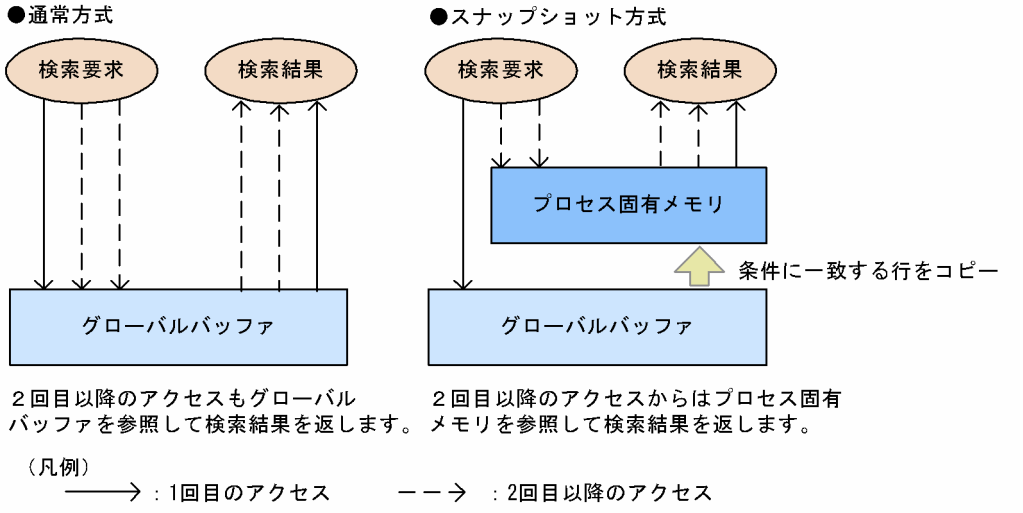
pd_dbbuff_lru_option オペランド及び pdbuffer オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

6.8.9　スナップショット方式によるページアクセス

性能向上を目的とした機能（グループ分け高速化機能など）を適用できない検索をするとき，条件に合致する行数とほぼ同数回グローバルバッファにアクセスしています。**スナップショット方式**では，最初のアクセス時にバッファ内の探索条件に一致するすべての行をプロセス固有メモリ上にコピーし，2 回目以降の同一ページのアクセスはプロセス固有メモリ上を参照して検索結果を返します。このため，2 回目以降のアクセス時に掛かる検索時間を短縮できます。また，グローバルバッファへのアクセス回数を削減し，同一グローバルバッファへのアクセスの集中を防ぎます。

スナップショット方式の概要を次の図に示します。

図 6–12　スナップショット方式の概要



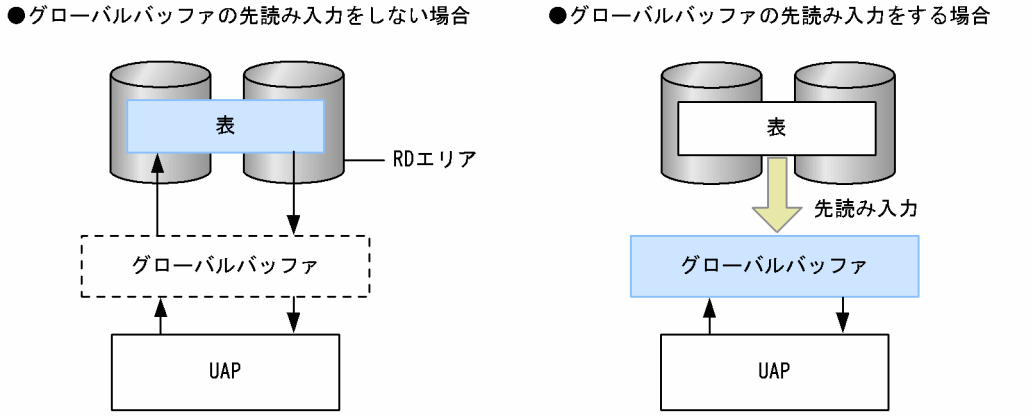
指定方法

pd_pageaccess_mode オペランドに **SNAPSHOT**（省略値）を指定します。

6.8.10　グローバルバッファの先読み入力

グローバルバッファの先読み入力とは，指定した表やインデクスのデータをあらかじめグローバルバッファに読み込んでおく機能です。概要を次の図に示します。

図 6–13　グローバルバッファの先読み入力の概要



〔説明〕

- グローバルバッファの先読み入力をしない場合

  HiRDB 開始直後に UAP が表にアクセスする時，グローバルバッファにはデータがないため，表からデータを読み込みます（物理的な入出力が発生します）。以降，この表のデータにアクセスする時は，グローバルバッファに読み込まれているページについては表からの読み込みは発生しません。ただし，ほかのページのデータにアクセスする時は，読み込み処理が発生します。

- グローバルバッファの先読み入力をする場合

  あらかじめ表のデータをグローバルバッファに読み込んでいるため，表からデータを読み込まないで表にアクセスできます（物理的な入出力は発生しません）。以降，この表にアクセスする時，表からの読み込みはありません。

(1)　グローバルバッファの先読み入力の効果

指定した表やインデクスのデータを先読み入力しておくため，バッファヒット率が高くなります。HiRDB 開始直後，オンライン業務開始前などに，入出力が多いと思われる表やインデクスを先読みしておくことで，高いバッファヒット率が期待できます。

(2)　実行方法

先読み入力する表やインデクスを指定し，グローバルバッファ常駐化ユティリティ（**pdpgbfon**）を実行します。

(3)　使用上の考慮点

グローバルバッファの先読み入力をする場合の考慮点を次に示します。

- グローバルバッファの面数は，先読み入力する表やインデクスが格納されているページ数より多く必要です。
- グローバルバッファの面数が十分でない場合，LRU 管理方式によってグローバルバッファから古いページ情報が追い出されます（システム定義の pd_dbbuff_lru_option オペランドの値に従って，アクセスしたグローバルバッファ中の最も古いページが追い出されます）。そのため，グローバルバッファの面数が十分でない場合，pdpgbfon を実行しても意味がありません。
- グローバルバッファ常駐化ユティリティ（pdpgbfon）で先読みする場合，格納ページ順の読み込みとなるため，プリフェッチ機能が有効となります。グローバルバッファを定義する場合，プリフェッチ数を指定することで実行時間の短縮が図れます。

6.8.11　ローカルバッファ

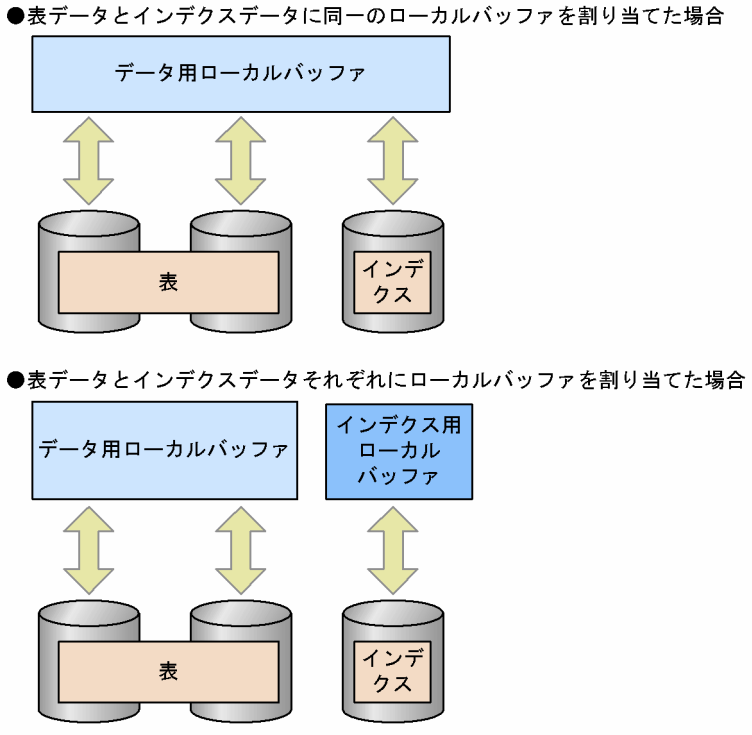
ローカルバッファとは，ディスク上の RD エリアに格納されているデータを入出力するためのバッファのことで，プロセス固有メモリ上に確保されます。ローカルバッファの種類を次の表に示します。

表 6–10　ローカルバッファの種類

ローカルバッファの種類	説明
データ用ローカルバッファ	データの入出力に使用されるローカルバッファです。データ用ローカルバッファは RD エリア単位に割り当てます。
インデクス用ローカルバッファ	インデクスデータの入出力に使用されるローカルバッファです。インデクス用ローカルバッファはインデクス単位に割り当てます。

これらのローカルバッファを使用すると性能を向上できます。例えば，データ用ローカルバッファとインデクス用ローカルバッファを分けて定義すると，データの検索とインデクスの検索を同時に実行しても互いが独立して動作します。そのため，大量データの全件検索時などでもインデクスの入出力回数を削減でき，処理時間を短縮できます。ローカルバッファの概念を次の図に示します。

図 6–14　ローカルバッファの概念



(1)　ローカルバッファの適用基準

次に示す条件をすべて満たす場合にローカルバッファを定義します。

- 大量のデータを検索又は更新する
- アクセス対象の RD エリアがほかの UAP からアクセスされない

また，HiRDB に常時接続する UAP はシステムへの影響（メモリの圧迫，プロセスの占有など）が大きいため，ローカルバッファを定義しないでください。

(2)　ローカルバッファの割り当て方法

pdlbuffer オペランドでローカルバッファを割り当てます。pdlbuffer オペランドでのローカルバッファの割り当て例を次に示します。

（例）

```
pdlbuffer -a localbuf01 -r RDAREA01,RDAREA02 -n 1000       1
pdlbuffer -a localbuf02 -i USER01.INDX01 -n 1000           2
```

〔説明〕

1. RD エリア（RDAREA01，RDAREA02）にデータ用ローカルバッファを割り当てます。
2. インデクス（USER01.INDX01）にインデクス用ローカルバッファを割り当てます。

pdlbuffer オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

6.8.12　BLOB データのファイル出力機能

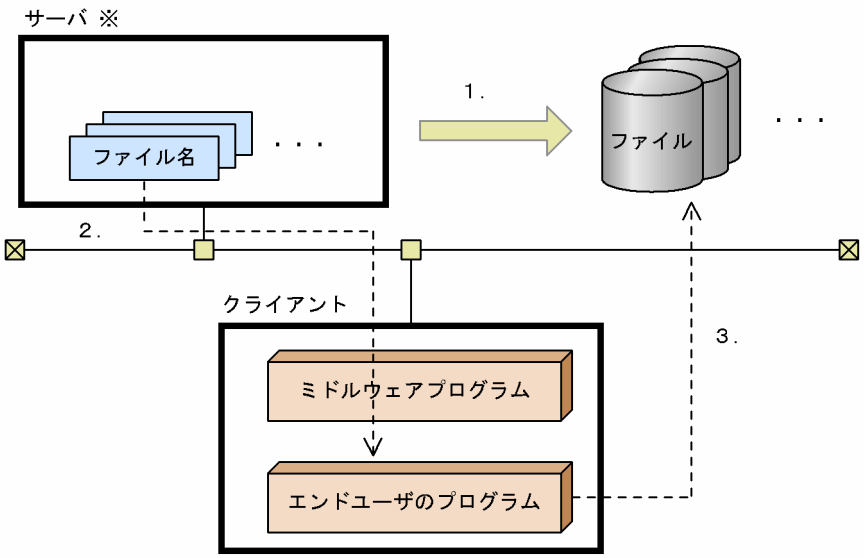
クライアントから HiRDB サーバに格納している BLOB データを検索する場合，次に示す不具合が起こります。

- BLOB データを格納するためのメモリ領域をクライアント側で用意する必要があります。
- サーバ側では BLOB データを返却するための送信バッファやクライアントで BLOB データを受け取るための受信バッファのメモリも必要になります。
- BLOB データに合わせた長大なメモリを確保するため，メモリ資源を圧迫します。
- エンドユーザのプログラムに HiRDB のクライアントとして動作するミドルウェアプログラムが介在するときは，BLOB データの受け渡しで更にメモリが増えることが考えられます。

このような BLOB データ検索時のメモリ増加を防ぐため，検索した BLOB データをクライアントに返却しないで，シングルサーバ又はフロントエンドサーバがあるユニットのファイルに出力して，サーバの IP アドレスとファイル名だけをクライアントに返却するように設定できます。この機能を **BLOB データのファイル出力機能**といいます。

BLOB データのファイル出力機能の概要を次の図に示します。

図 6–15　BLOB データのファイル出力機能の概要



注※　シングルサーバ又はフロントエンドサーバがあるサーバマシン

〔説明〕

1. クライアントから BLOB データを検索した場合，その BLOB データを１行１列ごとにファイルに出力します。
2. 1．で出力した BLOB データのファイル名をクライアントに返却します。
3. 返却されたファイル名を基に，サーバ側にある BLOB データのファイルにアクセスします。

(1) 適用基準

BLOB データを検索する場合で，メモリ所要量を削減したいときに適用します。ただし，クライアントプログラムのメモリ削減及びサーバ，クライアント間の通信バッファのメモリ削減に効果はありますが，サーバ側のディスク入出力時間と容量は増加します。メモリ所要量とディスク入出力の兼ね合いを考慮して利用してください。

(2) 指定方法

BLOB データのファイル出力機能を使うかどうかは，**SQL の WRITE 指定**で指定します。WRITE 指定は，カーソル指定及び問い合わせ指定の中に指定できます。WRITE 指定については，マニュアル「HiRDB SQL リファレンス」を参照してください。

SQL の検索結果として，サーバの IP アドレス，SQL に設定した BLOB データの格納場所と BLOB データのファイル名だけがクライアントで取得できます。クライアントではこれらの情報を基に，サーバに格納された BLOB データを特定します。

(3) BLOB データのファイル出力機能を使用する場合の注意

- 作成した BLOB データのファイルが不要になった場合は，ユーザが削除する必要があります。ファイルを削除する場合は，次に示すことに注意してください。なお，カーソルのクローズ後とトランザクション解決後は，無条件に削除できます。
  - **FETCH の直後に削除する場合**

    同じカーソル検索の直前の FETCH による結果と BLOB 値が同じときは，同じファイル名でファイルを再作成しないことがあります。この場合，直前のファイル名を覚えておいて，ファイル名が変わったときに削除するようにしてください。
- 障害又はロールバックが発生しても，作成した BLOB データのファイルは削除されません。また，ファイルを削除しないままでいると，ディスク容量など OS の資源を圧迫することになるので注意してください。
- 次に示す機能を使用する場合，事前にディスク容量に空きがあるかどうかを確認しておいてください。
  - **配列を使用した FETCH 機能を使用する場合**

    1 回の FETCH の実行で，配列用要素数分のファイルが作成されます。
  - **ブロック転送機能を使用する場合**

    最初の FETCH の実行でブロック転送行数分のファイルを作成し，以降ブロック転送行数分の FETCH 終了後，次回以降に FETCH が実行されるたびにブロック転送行数分のファイル作成を繰り返します。
- ほかのトランザクションやカーソル検索とファイル名が重複すると，ファイルを互いに上書きして破壊するおそれがあります。この場合，トランザクションごと又はカーソルごとにファイル接頭辞のディレクトリ名やファイル名を変えて，名称が重複しないようにしてください。

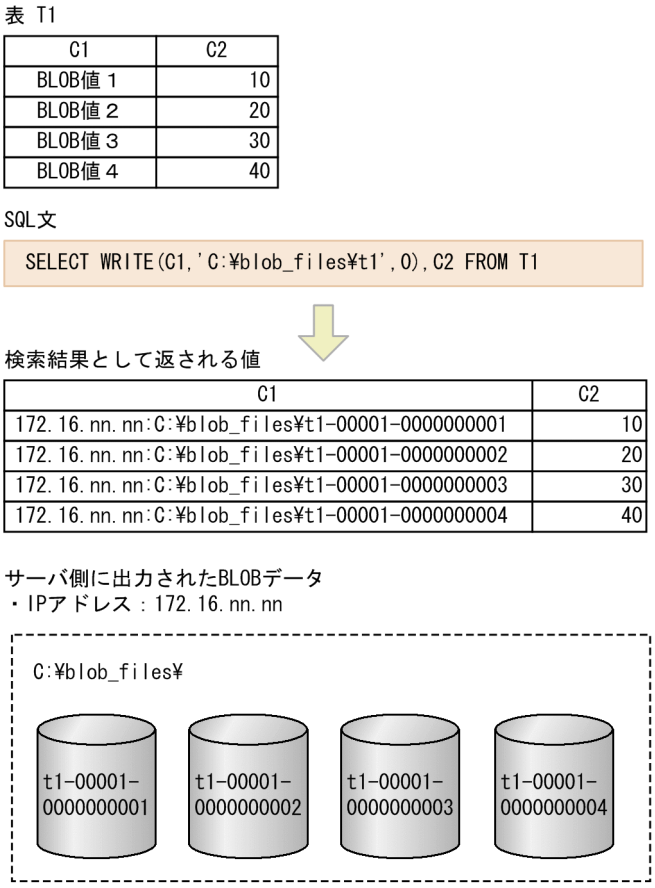
(4)　BLOB データのファイル出力機能を使用した例

BLOB データのファイル出力機能を使用した検索例を次に示します。

(a)　BLOB 列を検索する場合

表 T1 から，列 C1，C2 を検索します。このとき，C1 の BLOB データをファイル出力し，そのファイル名を取得します。BLOB データのファイル出力機能を使用した検索例（BLOB 列を検索する場合）を次の図に示します。

図 6–16　BLOB データのファイル出力機能を使用した検索例（BLOB 列を検索する場合）



(b)　BLOB 属性の抽象データ型を検索する場合

表 T2 から，CONTAINS 関数が真となる ADT1 列を検索します。このとき，列値を EXTRACTS 関数の引数に渡した結果の BLOB 値をファイルに出力し，ファイル名を取得します。なお，この例は全件ヒットした場合を示します。BLOB データのファイル出力機能を使用した検索例（BLOB 属性の抽象データ型を検索する場合）を次の図に示します。

図 6–17　BLOB データのファイル出力機能を使用した検索例（BLOB 属性の抽象データ型を検索する場合）

表 T2

ADT1
抽象データ型値１
抽象データ型値２
抽象データ型値３
抽象データ型値４

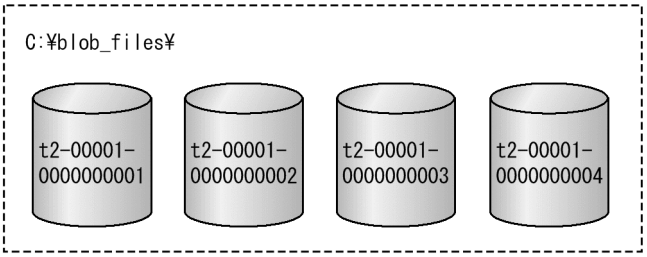
SQL文

```
SELECT WRITE(EXTRACTS(ADT1,…),'C:¥blob_files¥t2',0) FROM T2
   WHERE CONTAINS(ADT1,…) IS TRUE
```

検索結果として返される値

ADT1
172.16.nn.nn:C:¥blob_files¥t2-00001-0000000001
172.16.nn.nn:C:¥blob_files¥t2-00001-0000000002
172.16.nn.nn:C:¥blob_files¥t2-00001-0000000003
172.16.nn.nn:C:¥blob_files¥t2-00001-0000000004

サーバ側に出力されたBLOBデータ
・IPアドレス：172.16.nn.nn



6.8.13　BLOB データ，BINARY データの部分的な更新・検索

(1)　BLOB データ，BINARY データの部分的な更新・検索とは

登録されている BLOB データ又は BINARY データに対して，新たなデータを追加する場合に BLOB データ又は BINARY データを更新したり，BLOB データ又は BINARY データを検索する場合に BLOB データ又は BINARY データ全体を取得したりすると，サーバ及びクライアントの双方で BLOB データ又は BINARY データに合わせた長大なメモリを確保する必要があり，メモリ資源を圧迫します。

BLOB データ，BINARY データの部分的な更新・検索を行うと，メモリ使用量が BLOB データ，BINARY データ追加分，又は BLOB データ，BINARY データ抽出分だけになるため，メモリ資源の圧迫を防げます。

なお，BINARY データの場合は，定義長が 32,001 バイト以上のデータに限ります。

BLOB データ，BINARY データの追加更新：

UPDATE 文の SET 句に連結演算を指定することで，登録されている BLOB データ又は BINARY データに対して新たなデータを追加できます。

BLOB データ，BINARY データの部分抽出：

スカラ関数 SUBSTR を指定することで，BLOB データ又は BINARY データから，指定した部分だけを抽出できます。

BLOB データ，BINARY データの後方削除更新：

UPDATE 文の SET 句に，処理対象列，及び開始位置として定数 1 を指定したスカラ関数 SUBSTR を使用することで，BLOB データ又は BINARY データの後方部分だけを削除できます。

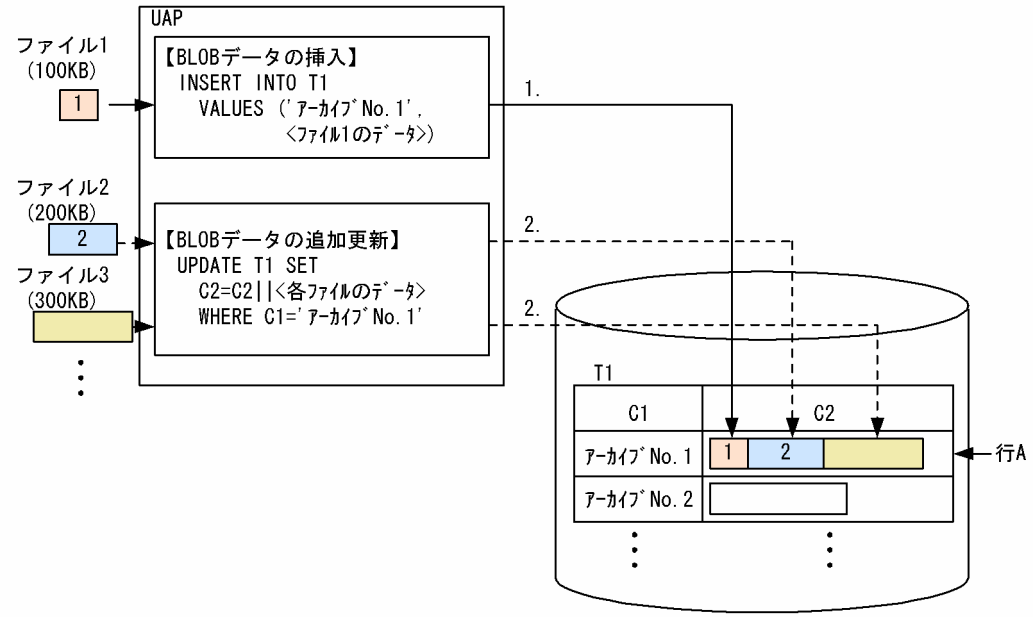
BLOB データ，BINARY データの部分的な更新・検索については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(2) BLOB データ，BINARY データの部分的な更新・検索の例

(a) BLOB データの追加更新

複数のファイルを一つの BLOB データとして格納します。BLOB データの追加更新の例を次の図に示します。

図 6–18 BLOB データの追加更新の例



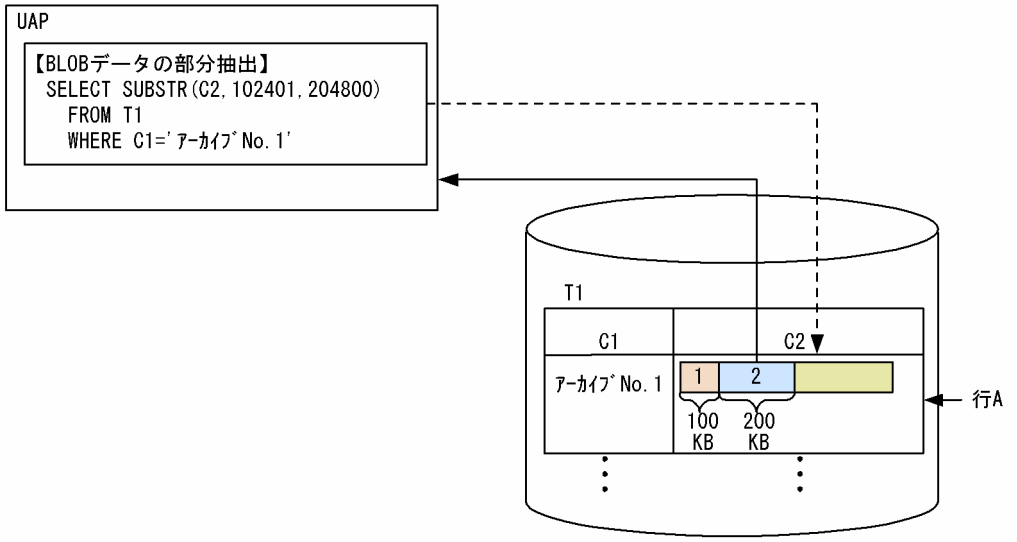
〔説明〕

1. 対象表（T1）の行 A の C2 列に，ファイル 1 の BLOB データを挿入します。
2. 行 A の C2 列に対して，ファイル 2 の BLOB データを連結することで追加更新されます。これ以降にデータを追加する場合も同様です。

(b) BLOB データの部分抽出

「BLOB データの追加更新」で格納した行 A の BLOB データ（C2 列）から，ファイル 2 の部分を抽出します。BLOB データの部分抽出の例を次の図に示します。

図 6–19　BLOB データの部分抽出の例



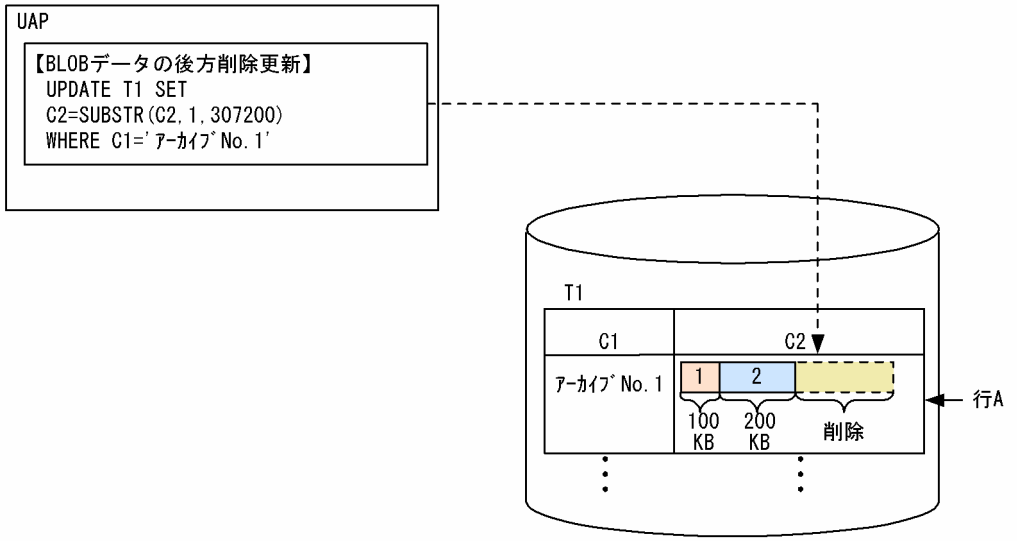
〔説明〕

スカラ関数 SUBSTR を使用して，行 A の C2 列の中からファイル 2 のデータ列の開始位置（100×1024 ＋ 1 ＝ 102401 バイト目）から，ファイル 2 のデータ列の長さ分（200×1024 ＝ 204800 バイト）だけ抽出します。

(c)　BLOB データの後方削除更新

「BLOB データの追加更新」で格納した行 A の BLOB データ（C2 列）の後方部分を削除し，ファイル 1 とファイル 2 だけを残します。BLOB データの後方削除更新の例を次の図に示します。

図 6–20　BLOB データの後方削除更新の例



[説明]

スカラ関数 SUBSTR を使用して，ファイル 1 のデータ列の開始位置 1 バイト目から，ファイル 1 とファイル 2 の長さ分（100×1024 ＋ 200×1024 ＝ 307200 バイト）だけを抽出したデータに置き換えて更新します。これによって，ファイル 1 とファイル 2 だけが残り，後方部分のデータは削除されます。

6.8.14　位置付け子機能

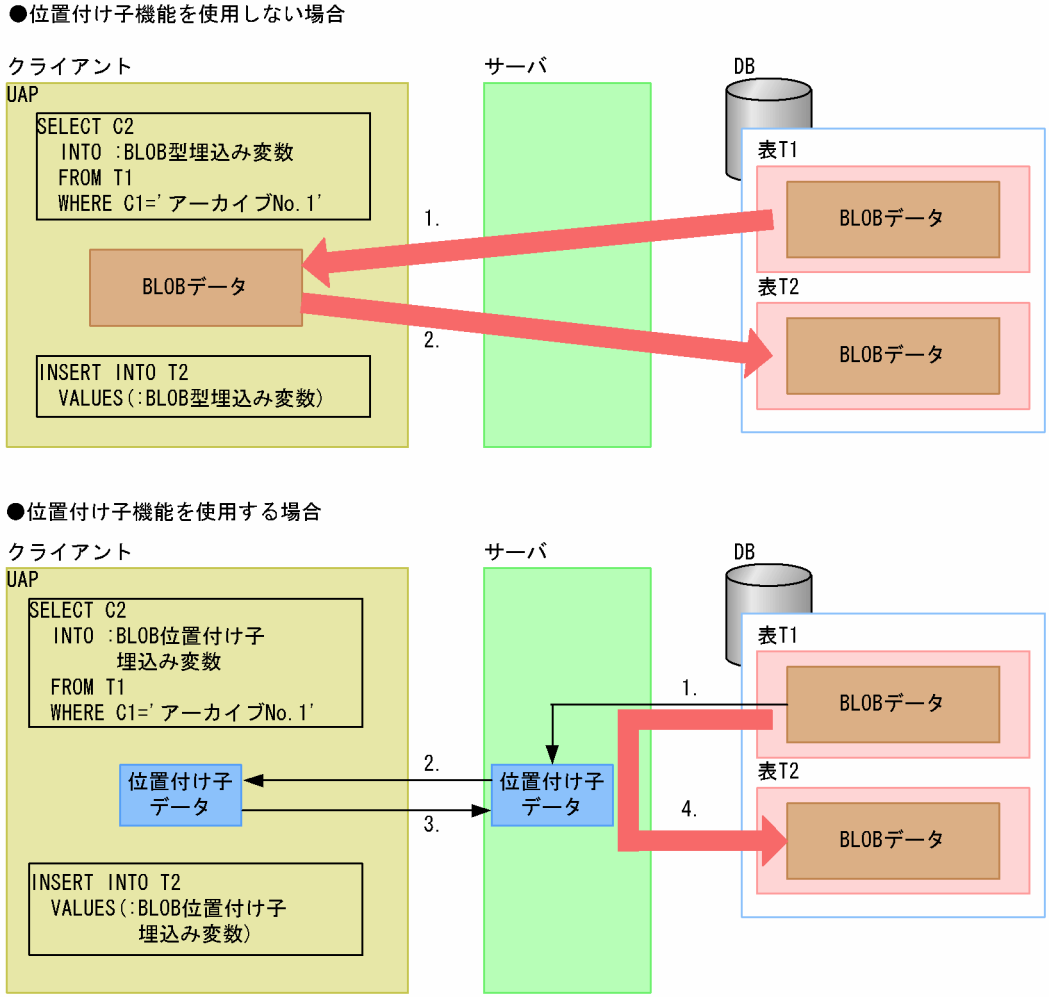
(1)　位置付け子機能とは

クライアントの UAP で，検索した BLOB データ又は BINARY データをそのデータ型の埋込み変数で受け取る場合，受け取ったデータを格納するためのメモリ領域をクライアント側で用意する必要があります。このため，長大なデータを検索する場合，クライアント側のメモリ資源を圧迫します。さらに，サーバからクライアントへのデータ転送量も大きくなります。しかし，必要なデータが一部だけであったり，受け取ったデータを変更しないでほかの SQL 文中に指定してサーバに送り返すだけであったりする場合，データをクライアントに転送することはむだなことになります。

位置付け子機能は，これを解決する機能です。位置付け子は，サーバ上のデータを識別する 4 バイトの値のデータであり，1 行 SELECT 文や FETCH 文の INTO 句などに位置付け子の埋込み変数を指定することで，検索結果としてデータの実体ではなく，そのデータを識別する位置付け子の値を受け取ります。また，データを識別する位置付け子の埋込み変数をほかの SQL 文中に指定すると，位置付け子が識別するデータを扱う処理ができます。

位置付け子機能の概要を次の図に示します。

図 6–21　位置付け子機能の概要



〔説明〕

位置付け子機能を使用しない場合

1. DB から検索した BLOB データを，サーバからクライアントに転送します。
2. クライアントからサーバに BLOB データを転送し，それを DB に格納します。

位置付け子機能を使用する場合

1. サーバが，DB から検索したデータを識別する位置付け子データを作成します。
2. 位置付け子データを，サーバからクライアントに転送します。
3. クライアントからサーバに位置付け子データを転送します。
4. 位置付け子データが識別するサーバ上の BLOB データを DB に格納します。

(2) 適用基準

BLOB データ又は BINARY データを検索する場合に適用します。

位置付け子機能を使用すると，クライアント側で実際のデータの大きさ分だけメモリを確保する必要がなくなり，更にサーバ，クライアント間のデータ転送を位置付け子で行えるため，データ転送量を少なくできます。

(3) 利点

位置付け子機能を使用すると，クライアント側のメモリ所要量を削減できます。また，サーバ，クライアント間のデータ転送量を少なくできます。

(4) 使用方法

位置付け子の値を受け取る場合，SQL 文中の BLOB 型又は BINARY 型のデータを受け取る埋込み変数を指定する箇所に，対応するデータ型の位置付け子の埋込み変数を指定します。また，位置付け子に割り当てられたデータを処理する場合は，SQL 文中に BLOB 型又は BINARY 型の埋込み変数を指定する代わりに，対応するデータ型の位置付け子の埋込み変数を指定します。

位置付け子機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

6.9　トランザクション制御

UAP で HiRDB のデータベースへアクセスする場合のトランザクションの制御について説明します。トランザクションとは，論理的な作業単位のことです。HiRDB のシステムの場合，トランザクション制御には次に示す意味があります。

HiRDB 独自のトランザクション制御：

HiRDB 仕様の範囲のトランザクションです。SQL 文で COMMIT 文又は ROLLBACK 文を発行して，表データの更新処理を有効にするか無効にするかを決定する処理のことです。

XA インタフェースに準拠したトランザクション制御：

XA インタフェースで OLTP と連携したときに UAP の処理で実行されるトランザクションです。XA インタフェースの関数（tx_commit 関数，tx_rollback 関数など）で OLTP 下の UAP の処理を有効にするか無効にするかを決定する処理のことです。

6.9.1　HiRDB への接続と切り離し

HiRDB への接続

UAP で，HiRDB のデータベースにアクセスする状態にすることを HiRDB への接続といいます。トランザクションを開始する前に，HiRDB に接続する必要があります。HiRDB への接続は，制御系 SQL の **CONNECT 文**で定義します。

HiRDB との切り離し

UAP での HiRDB のデータベースにアクセスを終了し，HiRDB への接続を切り離すことを HiRDB との切り離しといいます。HiRDB の切り離しをすると，トランザクションを終了させ，**同期点**を取得します。HiRDB との切り離しは，制御系 SQL の **DISCONNECT 文**で定義します。

6.9.2　複数接続機能

HiRDB では，一つの HiRDB クライアントの UAP から，同時に複数の HiRDB サーバに接続できます。この機能を複数接続機能といいます。複数接続機能を使用すると，HiRDB クライアント側の一つの UAP から一つの HiRDB のサーバに対して複数接続できます。また，一つの UAP から複数の HiRDB のサーバに接続することもできます。ただし，複数の接続はそれぞれ独立したトランザクションとして扱われます（HiRDB サーバから見れば，別々の UAP から接続された場合と同様に扱われます）。一つの UAP から複数接続できることから，実行する UAP の数を削減でき，全体としての UAP のメモリ所要量を削減できます。

複数接続機能を使用する場合には，専用の提供ライブラリをリンケージする必要があります。複数接続機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

複数接続機能を使用できる接続形態

複数接続機能は「HiRDB 独自のトランザクション制御」でも「XA インタフェースに準拠したトランザクション制御」でも使用できます。ただし，XA インタフェースで OLTP と連携する場合に複数接続機能を使用するときは，トランザクションマネジャに HiRDB を登録するときに指定する項目が異なります。登録する方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

複数接続機能の設定

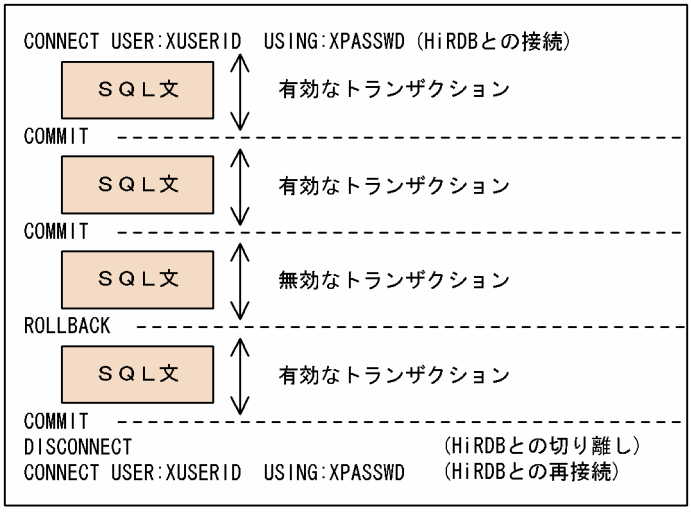
複数接続機能を使用する場合は，次に示す SQL を UAP に定義する必要があります。

- 接続ハンドルを割り当て（ALLOCATE CONNECTION HANDLE 文）
- 接続ハンドルの宣言（DECLARE CONNECTION HANDLE SET 文）
- 接続ハンドルの宣言の解除（DECLARE CONNECTION HANDLE UNSET 文）
- 接続ハンドルの解放（FREE CONNECTION HANDLE 文）

6.9.3　トランザクションの開始と終了

HiRDB のトランザクションは，最初の SQL が実行されたときに開始します。また，トランザクションは同期点の設定（コミット又はロールバック）によって終了します。HiRDB との接続中は，トランザクションの開始と終了を幾つでも実行できます。トランザクションの開始と終了の例を次の図に示します。

図 6–22　トランザクションの開始と終了の例



6.9.4　コミットとロールバック

トランザクションによるデータベースの更新内容が有効になることを**コミット**といいます。トランザクションによる更新内容が無効になることを**ロールバック**といいます。コミットとロールバックのタイミングには，大きく分けて次に示す 2 種類があります。

- SQL でタイミングを定義する場合

- HiRDB によって自動的に設定される場合

コミット及びロールバックの設定タイミングをそれぞれの種類ごとに説明します。

(1) コミットのタイミング

コミットのタイミングを次に示します。

SQL でタイミングを定義する場合

制御系 SQL の COMMIT 文を指定して，COMMIT 文が実行されるタイミングでコミットできます。

HiRDB で自動的にコミットされる場合

以下の SQL 文実行時に HiRDB によって自動的にコミットされます。

- 定義系 SQL
- PURGE TABLE 文
- DISCONNECT 文

(2) ロールバックのタイミング

ロールバックのタイミングを次に示します。

SQL でタイミングを定義する場合

制御系 SQL の ROLLBACK 文を指定して，ROLLBACK 文が実行されるタイミングで直前のコミット時点までロールバックできます。

HiRDB で自動的にロールバックされる場合

- SQL 実行時に処理が続行できなくなった場合，HiRDB によって，直前のコミット時点まで暗黙的にロールバックされます。
- 以下の SQL 文を実行しないで UAP が終了した場合， HiRDB によって，直前のコミット時点までロールバックされます。

  COMMIT 文

  ROLLBACK 文

  DISCONNECT 文

(3) HiRDB/パラレルサーバのコミットメント制御

HiRDB/パラレルサーバのコミットメント制御は次に示す二つの方式があります。

- 一相コミット
- 二相コミット

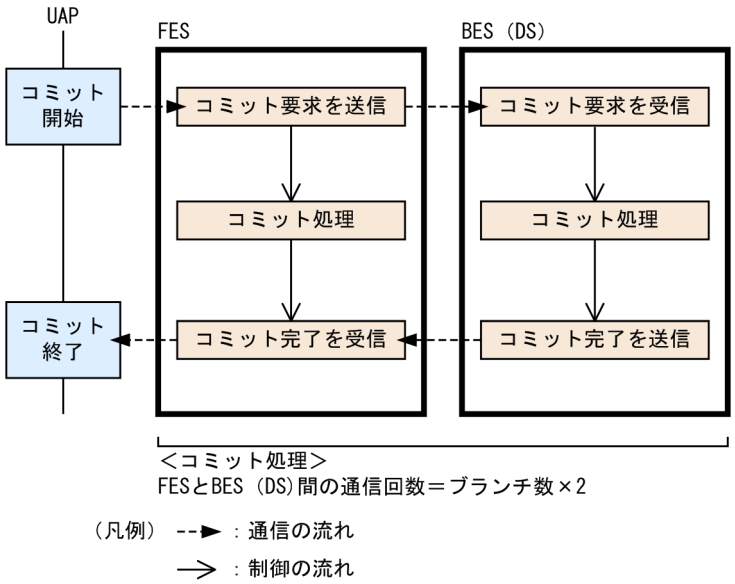
(a)　一相コミット

コミットメント制御を二相（プリペア処理とコミット処理）にしないで，コミット処理だけを行います。したがって，フロントエンドサーバとバックエンドサーバ（ディクショナリサーバ）間の同期点処理の通信回数が**ブランチ数×2**（二相コミットの場合は**ブランチ数×4**）になるため，トランザクションの処理性能が向上します。コミットメント制御に一相コミットを使用する場合は，**pd_trn_commit_optimize** オペランドに **ONEPHASE**（省略値）を指定してください。

なお，一相コミットが適用されるケースは，一つのトランザクション内の更新ブランチ数が一つのときだけです。それ以外の場合は二相コミットが適用されます。

一相コミットの処理方式を次の図に示します。

図 6–23　一相コミットの処理方式

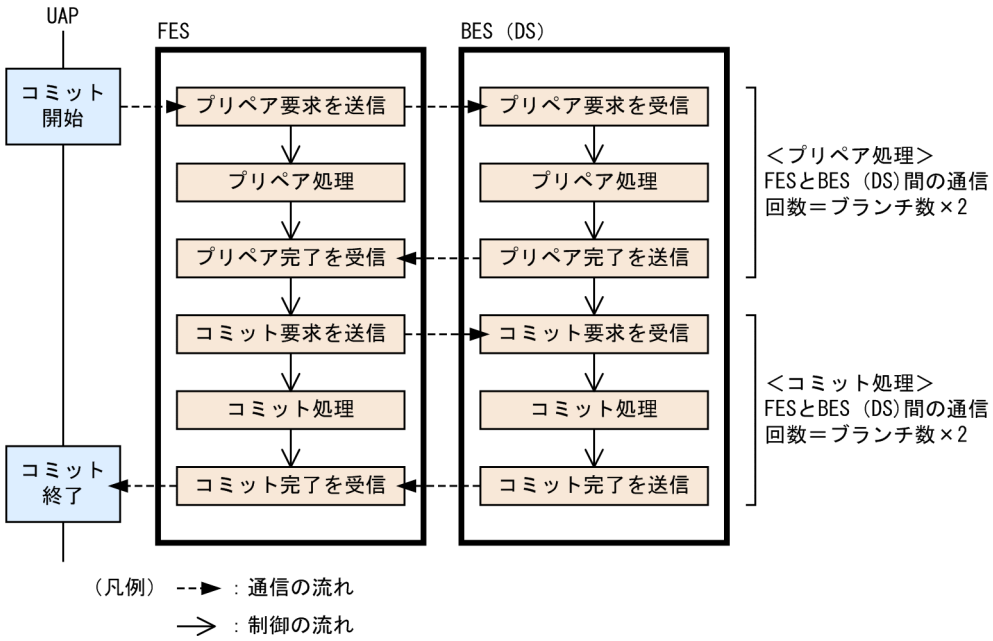


コミットメント制御で一相コミットを行うことを**一相最適化**といいます。

(b)　二相コミット

コミットメント制御を**プリペア処理**と**コミット処理**の二相に分けてトランザクションの同期点処理を行います。フロントエンドサーバとバックエンドサーバ（ディクショナリサーバ）間の同期点処理の通信回数が**ブランチ数×4** になります。二相コミットの処理方式を次の図に示します。

図 6–24　二相コミットの処理方式



コミット発行元，及びトランザクションの実行環境によって，HiRDB/パラレルサーバのコミットメント制御が決定されます。HiRDB/パラレルサーバのコミットメント制御を次の表に示します。

表 6–11　HiRDB/パラレルサーバのコミットメント制御

条件			HiRDB のコミットメント制御
コミット発行元	コミット発行元が指示したコミットメント制御	トランザクションの実行環境	
UAP	－	参照トランザクションの場合	一相コミット
		トランザクションによる更新サーバ数が一つで，かつ pd_trn_commit_optimize オペランドに ONEPHASE を指定（又は省略）している場合	
		上記以外	二相コミット
OLTP システム	一相コミット	参照トランザクションの場合	一相コミット
		トランザクションによる更新サーバ数が一つで，かつ pd_trn_commit_optimize オペランドに ONEPHASE を指定（又は省略）している場合	
		上記以外	二相コミット
	二相コミット	参照トランザクションの場合	一相コミット
		上記以外	二相コミット

（凡例）－：該当しません。

6.9.5　OLTP 環境での UAP のトランザクション管理

OLTP 環境の UAP で XA インタフェースに準拠したトランザクション処理をするときは，**X/Open に準拠した API** でトランザクションのコミットとロールバックを UAP から実行します。

トランザクションの移行

UAP が HiRDB をアクセスしたときと異なるプロセスでトランザクションのコミット処理を実行することを**トランザクションの移行**といいます。ここでの UAP とは，HiRDB XA ライブラリを使用して HiRDB に接続する UAP のことです。

6.9.6　自動再接続機能

サーバプロセスダウン，系切り替え，ネットワーク障害などの要因で HiRDB サーバとの接続が切断された場合に，自動的に再接続する機能を自動再接続機能といいます。自動再接続機能を使用すると，ユーザは HiRDB サーバとの接続の切断を意識しないで，UAP の実行を継続できます。自動再接続機能を使用する場合はクライアント環境定義 PDAUTORECONNECT に YES を指定します。

適用基準

HiRDB サーバで次の処理を実行している場合，HiRDB クライアントはその処理が終わるまで待ち状態となります。

- システム構成変更コマンド（pdchgconf）を実行している場合
- 修正版 HiRDB の入れ替えを実行している場合
- トランザクションキューイング機能を使用して，計画系切り替えを実行している場合

待ち状態となっている間，PDCWAITTIME の時間で待ち時間が監視されます。PDCWAITTIME の時間を超えた場合，待ち状態は解除されて UAP に PDCWAITTIME オーバーのエラーを返却します。

タイミングによっては，上記の処理が実行中であることを検知できないため，通信処理エラーになることがあります。あらかじめ上記の処理を実行することが分かっている場合は，自動再接続機能の適用を検討してください。自動再接続機能を使用すると，上記の処理を実行している場合でも，UAP にエラーを返さないで処理を続行できます。

6.10　排他制御

複数のユーザが，一つの表のデータを同時に操作しようとする場合，データの不整合が発生するのを防ぐため，HiRDB は自動的に**排他制御**をして，データの整合性を保持しています。特に，HiRDB/パラレルサーバは，基本的に各サーバ間で資源の共有をしないため，サーバごとに独立した排他制御をしています。ただし，業務の内容によっては HiRDB が自動的に行っている排他制御を変更できます。

6.10.1　排他制御の単位

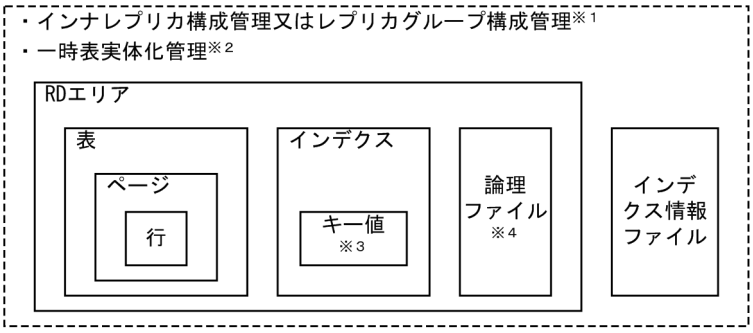
排他制御の単位とその包含関係について説明します。

(1)　排他資源とその包含関係

HiRDB は**排他資源**という単位で排他を掛けて，不正に参照されたり，更新されたりすることを防止しています。排他制御では排他資源の上位から下位へと順番に排他が掛けられます。排他を掛けていく途中で同一資源に対して，ほかのトランザクションと同時に実行できないトランザクションがある場合には，そのトランザクションは待ち状態になります。

また，排他資源には包含関係があるため，上位の資源で排他を掛けると，それより下位の資源には排他を掛ける必要がなくなります。排他資源とその包含関係を次の図に示します。例えば，表とページとの関係では，表が上位，ページが下位の排他資源です。

図 6–25　排他資源とその包含関係



注※1【UNIX 版限定】

インナレプリカ機能使用時の最上位排他資源は，インナレプリカ構成管理情報，又はレプリカグループ構成管理情報となります。

インナレプリカ構成管理情報の排他を取得できない場合に，レプリカグループ構成管理情報の排他を取得します。レプリカ RD エリアを定義していない RD エリアにアクセスする場合でも，排他が掛かります。これによって，業務実行中に通常 RD エリアにレプリカ RD エリアが定義されたり，インナレプリカグループ内の構成が変わったりすることを抑止します。

注※2

一時表の実体化時に，一時的に取得します。一時表に対する操作で取得される排他については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

注※3

インデクスキー値無排他を適用した場合，キー値には排他を掛けません。インデクスキー値無排他については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注※4

論理ファイルはプラグインが使用するファイルです。

(2)　最下位の排他資源の単位設定

HiRDB が自動的にする排他制御に対して，表ごとに最下位の排他資源の単位を設定できます。最下位の排他資源の単位，設定方法及びそれぞれの長所と短所を説明します。

(a)　行単位の排他制御

最下位の排他制御の単位を行にしたい場合には，定義系 SQL の **CREATE TABLE** で **LOCK ROW** を指定します。最下位の排他制御の単位が行の場合の排他制御を**行排他**といいます。ページ排他と比較して排他資源の単位が下位であるため，同時実行性が向上します。しかしその反面，排他制御による処理時間やメモリ使用量が増加します。

(b)　ページ単位の排他制御

最小の排他制御の単位をページにしたい場合には，定義系 SQL の **CREATE TABLE** で **LOCK PAGE** を指定します。最下位の排他制御の単位がページの場合の排他制御を**ページ排他**といいます。行排他と比較して排他資源の単位が上位であるため，排他制御による処理時間やメモリ使用量が減少します。しかし，その反面，同時実行性が低下します。

6.10.2　排他制御モード

HiRDB の排他制御では，表，ページなどのそれぞれの排他資源に対して，次に示す 5 種類の**排他制御モード**があります。

1. **共用モード**（**PR モード**：Protected Retrieve）
   ほかのトランザクションには参照だけを許すモードです。
2. **排他モード**（**EX モード**：Exclusive）
   一つのトランザクションだけが排他資源を占有し，ほかのトランザクションには参照，追加，更新及び削除を許さないモードです。
3. **意図共用モード**（**SR モード**：Shared Retrieve）
   参照だけが許されているモードです。ほかのトランザクションには排他資源の参照，追加，更新及び削除を許すモードです。
4. **意図排他モード**（**SU モード**：Shared Update）
   参照，追加，更新及び削除が許されているモードです。ほかのトランザクションにも排他資源の参照，追加，更新及び削除を許すモードです。

5. **共用意図排他モード（PU モード：**Protected Update）

参照，追加，更新及び削除が許されているモードです。ほかのトランザクションには参照だけを許すモードです。このモードは 1.～4.とは異なり，排他制御モードの遷移の結果として発生します。

6.10.3　HiRDB による自動的な排他制御

HiRDB が自動的にする排他制御について，「データの更新，追加，削除」及び「データの検索」のときの，表と行の排他制御について説明します。

(1)　データの更新，追加，削除時の HiRDB による排他制御

表に対する排他制御

ほかのトランザクションにもその表に対するデータの参照，追加，更新，削除を許します。つまり，**意図排他モード（SU モード）**が掛かります。

行に対する排他制御

そのトランザクションが資源を占有し，ほかのトランザクションからはその行に対する参照，追加，更新，削除を許しません。つまり，**排他モード（EX モード）**が掛かります。

(2)　データの検索時の HiRDB による排他制御

表に対する排他制御

ほかのトランザクションにもその表に対するデータの参照，追加，更新，削除を許します。つまり，**意図共有モード（SR モード）**が掛かります。

行に対する排他制御

そのトランザクションが資源を占有し，ほかのトランザクションには行の参照だけを許します。つまり，**共有モード（PR モード）**が掛かります。

6.10.4　ユーザの設定による排他制御の変更

HiRDB が自動的にしている排他制御をユーザが変更することもできます。「データの更新，追加，削除」及び「データの検索」のときの，表と行の排他制御について説明します。

(1)　データの更新，追加，削除時のユーザ設定による排他制御

LOCK TABLE 文の IN EXCLUSIVE MODE を指定すると，ユーザがデータの更新，追加，削除時の排他制御を変更できます。表に対して，ほかのトランザクションからのその表のデータに対する参照，追加，更新，削除は許しません。つまり，**排他モード（EX モード）**が掛かります。一方，行に対しては，行単位の排他はしないため，排他を掛ける処理のオーバヘッドが削減，排他制御用のバッファ不足を防止できます。し

かし，表単位には排他が掛かっているため，ほかのトランザクションが長時間待ち状態になる可能性があります。

(2)　データの検索時のユーザ設定による排他制御

次に示す SQL を指定すると，データの検索時の排他制御を変更できます。

SELECT 文の WITHOUT LOCK 指定

検索したデータの内容をトランザクション終了まで排他制御しません。検索し終わった行から排他制御が解除されるため，同時実行性が向上します。

LOCK TABLE 文の IN SHARE MODE 指定

そのトランザクションが資源を占有し，ほかのトランザクションからはその表のデータの参照だけが許されます。つまり，**共用モード（PR モード）**が掛かります。一方，行に対しては，行単位の排他はしないため，排他を掛ける処理のオーバヘッドが削減，排他制御用のバッファ不足を防止できます。しかし，表単位には排他が掛かっているため，ほかのトランザクションが長時間待ち状態になる可能性があります。

6.10.5　排他制御の期間

一つのトランザクションが，ある資源に対して排他を掛けると，コミット又はロールバックをするまで資源を占有します。例えば，ある排他資源（行又はページ）に対して，更新処理をすると，HiRDB による自動的な排他制御では，その資源への，ほかのトランザクションによる参照，追加，更新，削除を許さない排他モードになります。このため，更新中の行に対して，ほかのトランザクションは，コミット又はロールバックをするまですべて待ち状態になります。ただし，**LOCK 文**で **UNTIL DISCONNECT** を指定している場合には，DISCONNECT 文の実行時又はその表の削除後のコミット時まで，排他が掛かった状態になります。

6.10.6　デッドロック

二つのトランザクションが二つ以上の資源を異なる順序で確保したことで，互いに相手が確保した資源の解放を待つ状態になって処理が進まなくなることを**デッドロック**といいます。デッドロックは，一般に参照のトランザクションと更新及び削除のトランザクションとの間で多く発生します。したがって，UAP のアクセス順序を変えると，デッドロックの発生頻度を低減できます。また，HiRDB の運用時にデッドロックが発生した場合，HiRDB はデッドロック情報を出力します。デッドロック情報は，サーバ内の複数トランザクション間に発生したデッドロックの情報です。デッドロックを防止する方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

6.11　データベースの更新ログを取得しないときの運用

6.11.1　データベースの更新ログを取得しないときの運用方法

HiRDB は，UAP（又はユティリティ※）によって更新されたデータベースの履歴情報（システムログ中のデータベースの更新ログ）をシステムログファイルに取得しています。また，データベースの更新ログを取得しないこともできます。データベースの更新ログを取得しないと，その分の処理時間が短縮されます。したがって，UAP（又はユティリティ）の実行時間を短縮できます。

注※

次に示すユティリティのことです。

- データベース作成ユティリティ（pdload）
- データベース再編成ユティリティ（pdrorg）
- リバランスユティリティ（pdrbal）

(1)　データベースの更新ログ取得方式

UAP（又はユティリティ）を実行するときのデータベースの更新ログ取得方式には，次の表に示す 3 種類のモードがあります。

表 6–12　データベースの更新ログ取得方式

データベースの 更新ログ取得方式	説明
ログ取得モード	ロールバック及びロールフォワードに必要なデータベース更新ログを取得します。通常はログ取得モードで更新ログを取得します。
更新前ログ取得モード	ロールバックに必要なデータベース更新ログだけを取得します。
ログレスモード	データベース更新ログを取得しません。

(2)　データベースの更新ログ取得方式の指定方法

データベースの更新ログ取得方式の指定方法を次の表に示します。

表 6–13　データベースの更新ログ取得方式の指定方法

データベースの更新ログ 取得方式の指定方法	説明
UAP の場合	クライアント環境定義の **PDDBLOG** オペランドで，データベースの更新ログ取得方式を指定します。PDDBLOG オペランドでは，ログ取得モード又はログレスモードを指定できます。更新前ログ取得モードは指定できません。

データベースの更新ログ取得方式の指定方法	説明
データベース作成ユティリティ（pdload）の場合	データベース作成ユティリティ（pdload）の **-l** オプションで，データベースの更新ログ取得方式を指定します。
データベース再編成ユティリティ（pdrorg）の場合	データベース再編成ユティリティ（pdrorg）の **-l** オプションで，データベースの更新ログ取得方式を指定します。
リバランスユティリティ（pdrbal）の場合	リバランスユティリティ（pdrbal）の **-l** オプションで，データベースの更新ログ取得方式を指定します。
ユーザ LOB 用 RD エリアを使用している場合	ユーザ LOB 用 RD エリアに格納されているデータについては，CREATE TABLE の **RECOVERY** オペランドで，データベースの更新ログ取得方式を指定します。なお，ユーザ LOB 用 RD エリアに抽象データ型のデータが格納されている場合，更新前ログ取得モードは指定できません。指定しても無視されます。

(3) RECOVERY オペランドについての注意

RECOVERY オペランドで指定したデータベースの更新ログ取得方式は，PDDBLOG オペランド又は-l オプションの指定で変更される場合があります。「RECOVERY オペランドと PDDBLOG オペランド又は-l オプションの指定値」と「UAP（又はユティリティ）実行時に仮定される値」との関係を次の表に示します。

表 6–14 「RECOVERY オペランドと PDDBLOG オペランド又は-l オプションの指定値」と「UAP（又はユティリティ）実行時に仮定される値」との関係

PDDBLOG オペランド又は-l オプションの指定	RECOVERY オペランドの指定		
	ALL	PARTIAL	NO
ALL 又は a	ALL	PARTIAL	NO
p	PARTIAL	PARTIAL	NO
NO 又は n	NO	NO	NO

（凡例）

ALL 及び a：ログ取得モード

PARTIAL 及び p：更新前ログ取得モード

NO 及び n：ログレスモード

(4) データベースの更新ログ取得方式による運用方法の違い

データベースの更新ログ取得方式が異なると，次に示す運用方法が異なります。

- UAP が異常終了したときの HiRDB の処理とユーザの処置
- データベースを回復できる時点

(a) UAP が異常終了したときの HiRDB の処理とユーザの処置

UAP が異常終了したときの HiRDB の処理とユーザの処置を次の表に示します。

表 6–15 UAP が異常終了したときの HiRDB の処理とユーザの処置

データベースの更新ログ取得方式	HiRDB の処理	ユーザの処置
ログ取得モード	更新した RD エリアの状態を UAP 実行前の状態又は異常終了直前に取得した同期点までロールバックします。	UAP 実行前の状態にロールバックされた場合は，UAP を再実行してください。異常終了直前に取得した同期点までロールバックされた場合は，同期点以降の処理を実行してください。
更新前ログ取得モード		
ログレスモード	ロールバックしません。更新した RD エリアを**障害閉塞（ログレス閉塞）**します。RD エリアの内容は破壊されます。	UAP 実行前に取得したバックアップを入力情報として，データベース回復ユティリティ（pdrstr）で RD エリアを回復してください。その後，UAP を再実行してください。

(b) データベースを回復できる時点

データベース回復ユティリティ（pdrstr）を使用してデータベースを回復できる時点を次の表に示します。

表 6–16 データベースを回復できる時点

データベースの更新ログ取得方式	データベースを回復できる時点
ログ取得モード	バックアップ取得時点又はバックアップ取得時点以降の任意の同期点
更新前ログ取得モード	バックアップ取得時点
ログレスモード	

(5) バックアップについての注意（重要）

1. ログレスモード又は更新前ログ取得モードの UAP（又はユティリティ）の実行中に，更新可能モード（-M s 指定）でバックアップを取得しないでください。
2. ログレスモード又は更新前ログ取得モードの UAP（又はユティリティ）の実行後は，次に示すモードでバックアップを取得してください。
   - 参照・更新不可能モード（-M x 指定）
   - 参照可能モード（-M r 指定）

7 データベースの管理

この章では，HiRDB 管理者が実施するデータベースの管理方法について説明します。

7.1　データベースの回復

ディスク障害などでデータベースの回復が必要な場合は，**データベース回復ユティリティ（pdrstr）**でデータベースを回復します。ここでは，データベースの回復方法について簡単に説明します。データベースの回復方法の詳細については，マニュアル「HiRDB システム運用ガイド」を参照してください。

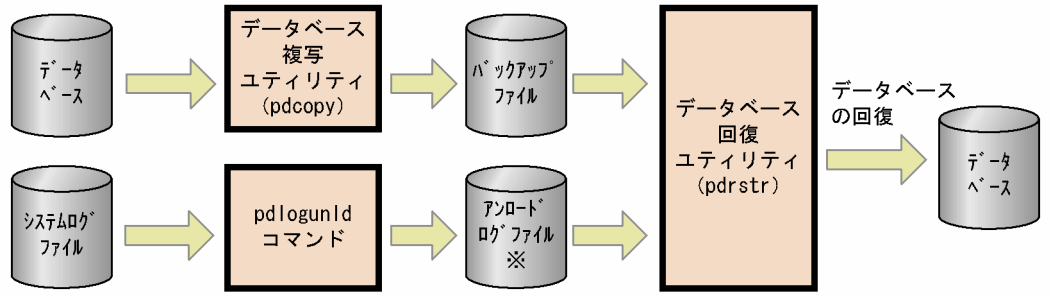
7.1.1　データベース回復の概要

次に示すファイルをデータベース回復ユティリティ（pdrstr）の入力情報にしてデータベースを回復します。

- バックアップファイル
- アンロードログファイル

データベース回復の概要を次の図に示します。

図 7–1　データベース回復の概要



注※

データベースの更新履歴情報を格納したシステムログファイルの内容（システムログ）をアンロードして作成したファイルを**アンロードログファイル**といいます。

〔説明〕

- データベース複写ユティリティ（pdcopy）でバックアップファイルを取得します。
- pdlogunld コマンドでアンロードログファイルを作成します。
- バックアップファイル及びアンロードログファイルを入力情報にして，データベース回復ユティリティ（pdrstr）でデータベースを回復します。

バックアップファイルの取得方法及びアンロードログファイルの作成方法については，「データベースの障害に備えた運用」を参照してください。

7.1.2　データベースを回復できる時点

データベースは次に示す時点（状態）に回復できます。

- バックアップ取得時点
- 障害発生直前の最新の同期点

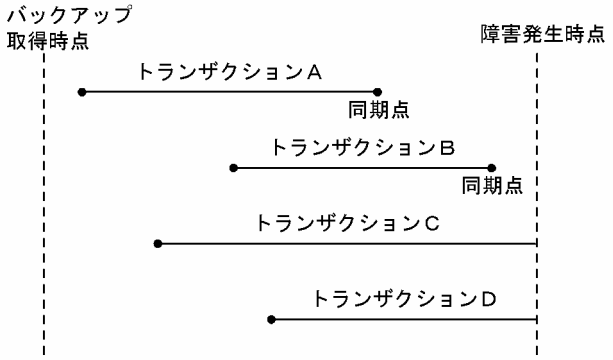
(1)　バックアップ取得時点に回復する場合

バックアップ取得時点に回復する場合は，アンロードログファイルは必要ありません。バックアップファイルだけが必要です。

(2)　障害発生直前の最新の同期点に回復する場合

トランザクションを決着した時点を**同期点**といいます。トランザクションによる更新処理を有効にする場合の同期点処理を**コミット**といい，無効にする場合の同期点処理を**ロールバック**といいます。障害発生時点で処理が完了しているトランザクションの同期点にデータベースを回復することを**障害発生直前の最新の同期点に回復する**といいます。障害発生時点で処理中のトランザクション（同期点に達していないトランザクション）は無効になるため，このトランザクションによる更新処理は回復の対象になりません。回復の対象になるトランザクションを次の図に示します。

図 7–2　回復の対象になるトランザクション（障害発生直前の最新の同期点に回復する場合）



〔説明〕

トランザクション A，B は処理を完了して同期点に達しているため，この同期点にデータベースを回復します。

トランザクション C，D は処理中で同期点に達していないため，このトランザクション処理は無効になります。したがって，回復の対象になりません。

障害発生直前の最新の同期点に回復する場合は，バックアップファイルのほかに，バックアップ取得以降に出力されたシステムログをアンロードしたアンロードログファイルが必要になります。

7.2　データベースの障害に備えた運用

ディスク障害などでデータベースに障害が発生した場合，HiRDB 管理者はデータベースの回復作業をする必要があります。HiRDB 管理者はデータベースの障害発生に備えて次に示すことを定期的にする必要があります。

- バックアップの取得
- システムログのアンロード（アンロードログファイルの作成）

また，バックアップ取得時のオプション機能として次に示す機能を紹介します。

- 差分バックアップ機能
- バックアップ閉塞
- 更新凍結コマンド（pddbfrz コマンド）
- NetBackup 連携機能

7.2.1　バックアップの取得

データベースの障害に備えて，定期的にデータベースのバックアップを取得する必要があります。バックアップは**データベース複写ユティリティ（pdcopy）**で取得します。

(1)　バックアップの取得単位

バックアップの取得単位はデータベース複写ユティリティ（pdcopy）のオプションで指定できます。バックアップの取得単位を次の表に示します。

表 7–1　バックアップの取得単位

バックアップの取得単位	説明	pdcopy のオプションの指定
システム単位	全 RD エリアを対象としてバックアップを取得します。マスタディレクトリ用 RD エリアなどのシステムが使用する RD エリアのバックアップも取得されます。	-a
ユニット単位※	ユニット下の RD エリアを対象としてバックアップを取得します。	-u ユニット名
サーバ単位※	サーバ下の RD エリアを対象としてバックアップを取得します。	-s サーバ名
RD エリア単位	RD エリアごとにバックアップを取得します。	-r RD エリア名

注※　HiRDB/パラレルサーバの場合に該当します。

(2) バックアップ取得モード

データベース複写ユティリティ（pdcopy）の**-M** オプションでバックアップ取得モードを選択します。バックアップ取得モードを次の表に示します。

表 7–2 バックアップ取得モード

バックアップ取得モード（-Mオプションの指定値）	モードの説明	バックアップ取得時点への RD エリアの回復方法の違い
参照・更新不可能モード（ｘ）	バックアップ取得中に，バックアップ対象 RD エリアを参照及び更新できません。バックアップを取得する前に，対象 RD エリアを pdhold -c コマンドで閉塞かつクローズ状態にする必要があります。	ここで取得したバックアップを使用して，データベースをバックアップ取得時点に回復できます。 また，システムログを使用すれば，バックアップ取得時点以降の任意の同期点に回復できます。
参照可能モード（ｒ）	バックアップ取得中に，バックアップ対象 RD エリアは参照だけできます。更新はできません。	
更新可能モード（ｓ）※1，※2	バックアップ取得中に，バックアップ対象 RD エリアを参照及び更新できます。	データベースをバックアップ取得時点には回復できません。バックアップ取得時点以降の任意の同期点への回復だけとなります。したがって，データベースを回復するには，バックアップ及びバックアップ取得直前のシンクポイントからのシステムログ※3が必要になります。

注※１【UNIX 版限定】

更新可能モードを指定する場合は，バックアップ取得対象 RD エリアがキャラクタ型スペシャルファイル上に作成されている必要があります。通常ファイル上に作成されている RD エリアは更新可能モードでバックアップを取得できません。

注※２

ログレスモード又は更新前ログ取得モードの UAP（ユティリティを含む）の実行中に，更新可能モードでバックアップを取得しないでください。

注※３

データベース複写ユティリティの処理結果出力ファイルに，RD エリアを回復するときに必要なシステムログファイルのラン ID 及び世代番号が出力されます。

(3) バックアップファイルを格納するサーバマシン

バックアップファイルは，HiRDB が稼働するサーバマシンであればどこにでも作成できます。バックアップを取得した RD エリアと同じサーバマシンに作成する必要はありません。CMT や DAT などのデバイスが，バックアップ対象 RD エリアと異なるサーバマシンにあってもかまいません。バックアップファイルを格納するサーバマシンは，データベース複写ユティリティ（pdcopy）のオプションで指定できます。

7.2.2　システムログのアンロード（アンロードログファイルの作成）

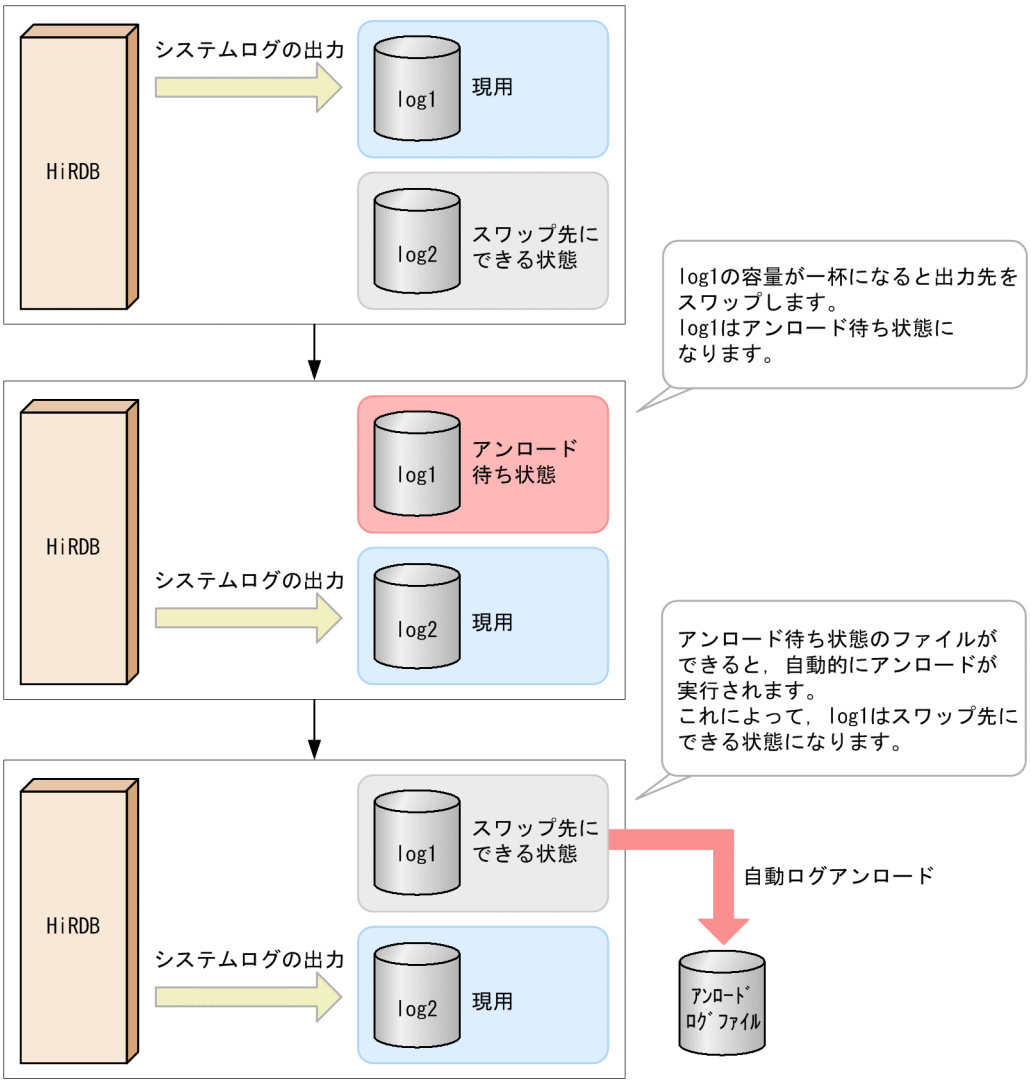
データベースの回復に必要なアンロードログファイルは，システムログをアンロードして作成します。システムログをアンロードしないで運用を続けると，システムログの出力先がなくなり，HiRDB は異常終了します。そのため，システムログのアンロードは自動で行うことをお勧めしています（**自動ログアンロード機能**）。自動ログアンロード機能の詳細は，マニュアル「HiRDB システム運用ガイド」の「自動ログアンロード機能の運用方法」を参照してください。

ここでは，システムログのアンロードが必要な理由や，アンロードログファイルの保管について説明します。

(1)　システムログファイルの状態について

システムログが出力されているファイルを**現用**ファイルといいます。この現用ファイルの容量一杯にシステムログが出力されると，システムログの出力先がほかのシステムログファイルに変更されます。これを**システムログファイルのスワップ**といいます。現用だったファイルは**アンロード待ち状態**のファイルとなります。自動ログアンロード機能を使用すると，システムログファイルがアンロード待ち状態になった時点で，自動的にシステムログをアンロードします。システムログファイルの状態変化を次の図に示します。

図 7–3　システムログファイルの状態変化



アンロード待ち状態のファイルにはシステムログを出力できません。**全システムログファイルがアンロード待ち状態になると，システムログが出力できなくなり，HiRDB が異常終了します。**そのため，自動ログアンロード機能の使用をお勧めしています。また，アンロードログファイル作成ディレクトリのディスク容量が満杯になると，自動ログアンロード機能が停止するため，不要なアンロードログファイルは定期的に削除して，自動ログアンロード機能が停止しないようにしてください。アンロードログファイルの保管については，「アンロードログファイルの保管について」を参照してください。

なお，アンロードは HiRDB 管理者が手動（**pdlogunld** コマンド）で実行することもできます。この場合，特に次の注意が必要です。

手動でアンロードを行う場合の注意事項

HiRDB を正常終了して，その後正常開始すると，正常開始時にシステムログファイルがスワップします。したがって，アンロード待ち状態のファイルができます。手動でアンロードを行う運用の場合，これを忘れないでアンロードしてください。**アンロードしないで何度も終了，開始を繰り返すと，HiRDB が開始できなくなります。**

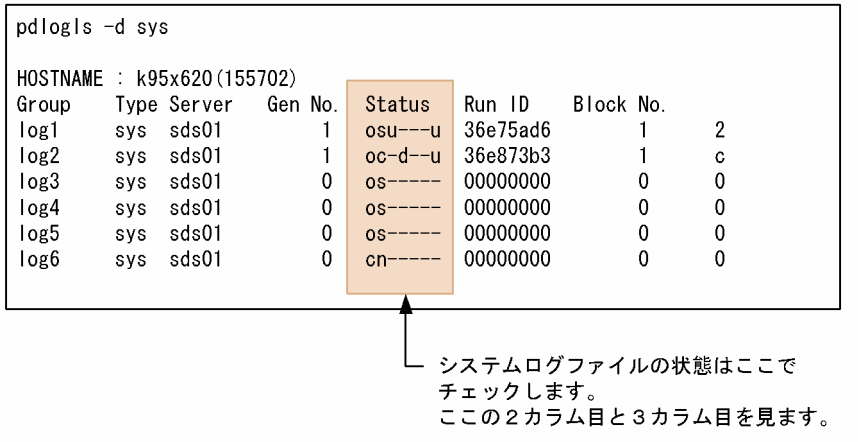
なお，サーバ定義で pd_log_rerun_swap=Y を指定すると，HiRDB の再開始時にもシステムログファイルがスワップするので注意してください。

(2)　システムログファイルの状態を知るには

手動でアンロードを行う場合は，まずシステムログファイルの状態を確認する必要があります。システムログファイルの状態は，**pdlogls コマンド**で確認できます。pdlogls コマンドの実行例を次に示します。

（例）

pdlogls コマンドでシステムログファイルの状態をチェックします。



〔説明〕

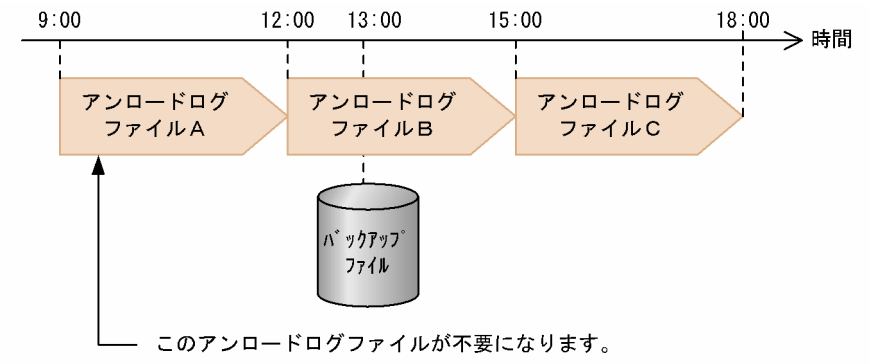
- 2 カラム目が「c」になっているファイルが，「現用」のファイルです。この場合，log2 が該当します。
- 3 カラム目が「u」又は「a」になっているファイルが，「アンロード待ち状態」のファイルです。この場合，log1 が該当します。手動でアンロードを行う運用の場合は，このファイルをアンロードしてください。

(3)　アンロードログファイルの保管について

システムログのアンロードを繰り返すと，アンロードログファイルが増えてディスク容量を圧迫する原因になります。HiRDB 管理者は不要なアンロードログファイルを削除するようにしてください。

バックアップを取得すると，バックアップ取得以前に出力されたシステムログをアンロードしたアンロードログファイルが不要になります。バックアップとアンロードログファイルの関係を次の図に示します。

図 7–4　バックアップとアンロードログファイルの関係



〔説明〕

- アンロードログファイルＡには，9：00～12：00 に出力されたシステムログがアンロードされています。
- アンロードログファイルＢには，12：00～15：00 に出力されたシステムログがアンロードされています。
- アンロードログファイルＣには，15：00～18：00 に出力されたシステムログがアンロードされています。
- バックアップを 13：00 ごろに取得したとすると，データベースの回復時にはバックアップとアンロードログファイル B 及び C が必要になります。アンロードログファイル A は不要になります。

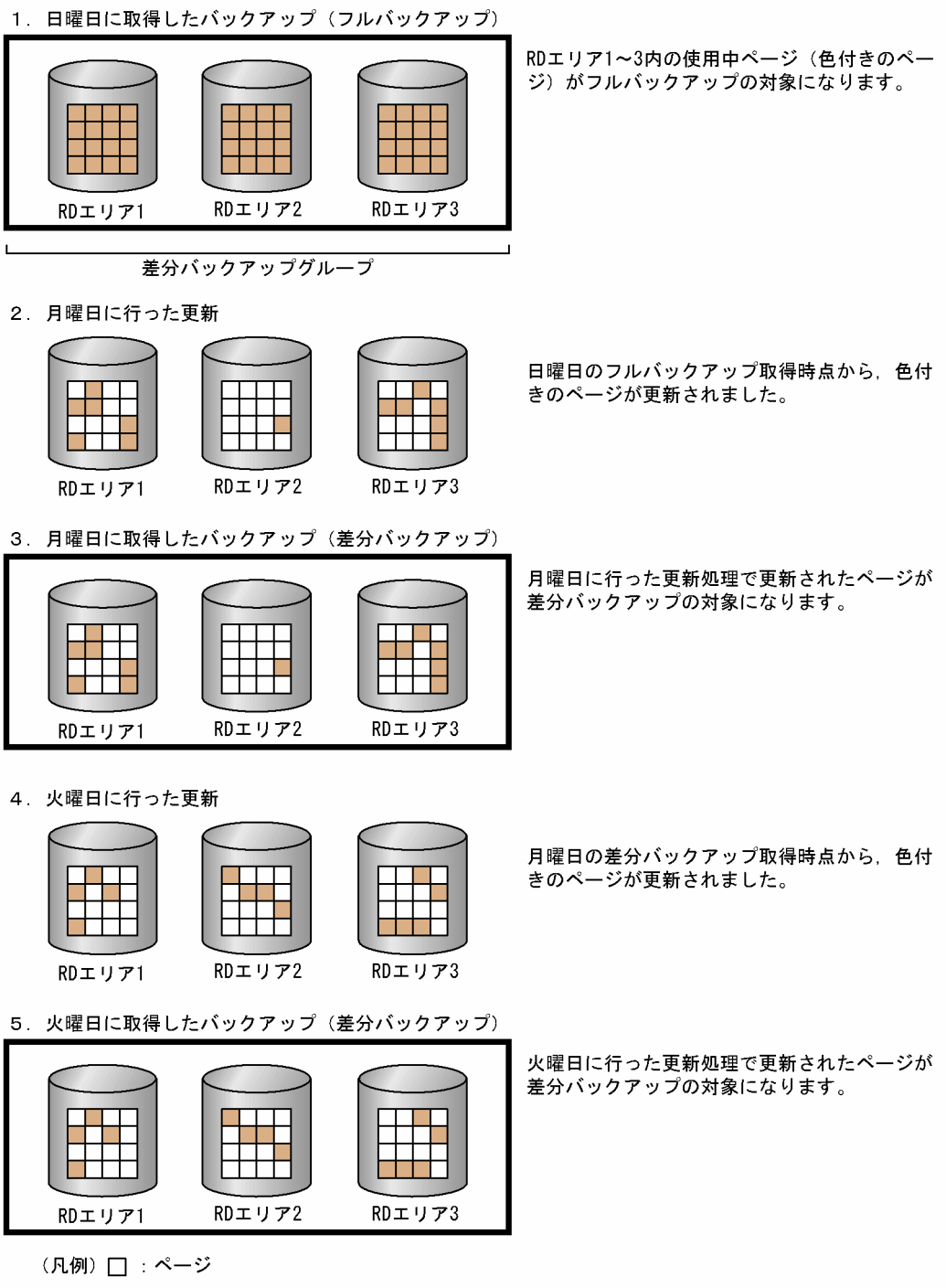
注意事項

アンロードログファイルをデータベースと異なるディスクに格納してください。同じディスクに格納すると，そのディスクに障害が発生した場合，データベースを回復できなくなります。

7.2.3　差分バックアップ機能

通常，バックアップは RD エリア単位に取得するため，更新したページも更新しないページもバックアップの取得対象になります。差分バックアップ機能を使用すると，前回のバックアップ取得時点から現在までに更新したページだけがバックアップの取得対象になります。このように，前回のバックアップ取得時点からの差分だけをバックアップとして取得するため，バックアップの取得処理時間を短縮できます。データベースの容量が多くてデータ更新量が少ない場合に，差分バックアップ機能の使用を検討してください。差分バックアップ機能の概要を次の図に示します。

図 7–5　差分バックアップ機能の概要



〔説明〕

1. 日曜日に RD エリア 1～3 のバックアップを取得します。このとき，RD エリア 1～3 内の使用中ページがバックアップの対象になります。このバックアップを**フルバックアップ**といい，グループ化した RD エリア群を**差分バックアップグループ**といいます。
2. 月曜日の業務で更新処理を行います。
3. 月曜日の業務終了後に RD エリア 1～3 のバックアップを取得します。このとき，RD エリア 1～3 内の更新ページがバックアップの対象になります。このバックアップを**差分バックアップ**といいます。

4. 火曜日の業務で更新処理を行います。
5. 火曜日の業務終了後に RD エリア 1～3 のバックアップを取得します。このとき，RD エリア 1～3 内の更新ページがバックアップの対象になります。

データベースの回復方法

火曜日の差分バックアップ取得時点にデータベースを回復する場合，データベース回復ユティリティの入力情報は日曜日に取得したフルバックアップ，月曜日に取得した差分バックアップ，火曜日に取得した差分バックアップになります。差分バックアップ機能の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

注意事項

LOB 用 RD エリアに対しては差分バックアップを取得できません。

参考

アンロードレスシステムログ運用の場合でも，差分バックアップ機能を使用できます。

7.2.4　バックアップ閉塞

pdcopy コマンド以外（ほかの製品の機能）でバックアップを取得する場合，RD エリアを**バックアップ閉塞**してください。RD エリアをバックアップ閉塞すると，HiRDB の稼働中にほかの製品のバックアップ機能でバックアップを取得できます。RD エリアをバックアップ閉塞するには，**pdhold** コマンドで**-b** オプションを指定します。

(1)　バックアップ閉塞の適用例

【UNIX 版の場合】

データベースに LVM（論理ボリューム・マネージャー）を使用している場合，バックアップ対象 RD エリアをバックアップ閉塞（-b u 又は-b wu）すると，pdcopy コマンドの更新可能モードでバックアップを取得できます。

【Windows 版の場合】

バックアップ対象 RD エリアをバックアップ閉塞（-b u 又は-b wu）すると，pdcopy コマンドの更新可能モードでバックアップを取得できます。

(2)　バックアップ閉塞の種類

バックアップ閉塞には次の表に示す四つの種類があります。

表 7–3　バックアップ閉塞の種類

バックアップ閉塞の種類	説明
参照可能バックアップ閉塞	バックアップ閉塞中にバックアップ閉塞 RD エリアの参照はできるが，更新は SQL エラー（–920）になります。
参照可能バックアップ閉塞（更新 WAIT モード）	バックアップ閉塞中にバックアップ閉塞 RD エリアを参照できます。更新はバックアップ閉塞が解除されるまで排他待ち状態になります。
更新可能バックアップ閉塞	バックアップ閉塞中にバックアップ取得対象 RD エリアの参照及び更新ができます。更新トランザクションが実行中でも，pdhold コマンドを待ち状態にしないで，RD エリアをすぐに更新可能バックアップ閉塞状態にします。
更新可能バックアップ閉塞（WAIT モード）	バックアップ閉塞中にバックアップ取得対象 RD エリアの参照及び更新ができます。更新トランザクションが実行中の場合は，更新トランザクションが終了するまで pdhold コマンドを待ち状態にします。

バックアップ閉塞を使用したバックアップの取得方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

参考

参照可能バックアップ閉塞，及び参照可能バックアップ閉塞（更新 WAIT モード）は**データベースの静止化**ともいいます。

7.2.5　ユーザ LOB 用 RD エリアのバックアップ取得時間の短縮（更新凍結コマンド）

更新凍結コマンド（pddbfrz コマンド）を使用すると，ユーザ LOB 用 RD エリアのバックアップ取得時間を短縮できることがあります。次に示す運用をする場合に更新凍結コマンドの使用を検討してください。

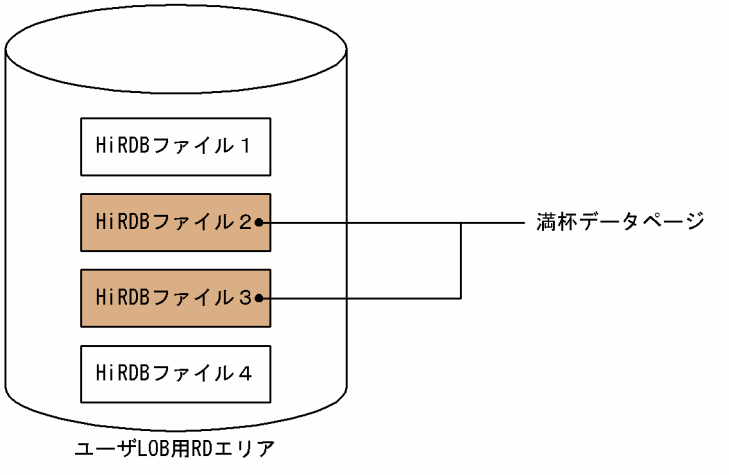
- データベース複写ユティリティ（pdcopy コマンド）以外の方法で，ユーザ LOB 用 RD エリアのバックアップを取得している
- バックアップの取得単位は HiRDB ファイル単位である
- 登録済みの LOB データの更新又は削除処理は通常発生しない運用である（LOB データの追加処理が通常の運用である）
- バックアップ取得対象のユーザ LOB 用 RD エリアは複数の HiRDB ファイルから構成されている

更新凍結コマンドを使用したバックアップの取得方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(1)　更新凍結コマンドとは

更新凍結コマンドを実行すると，ユーザ LOB 用 RD エリア中の満杯データページ（すべて割り当て済み）の HiRDB ファイルを**更新凍結状態**にします。更新凍結状態の HiRDB ファイル中のデータを更新及び削除できません。更新凍結コマンドの処理概要を次の図に示します。

図 7–6　更新凍結コマンドの処理概要



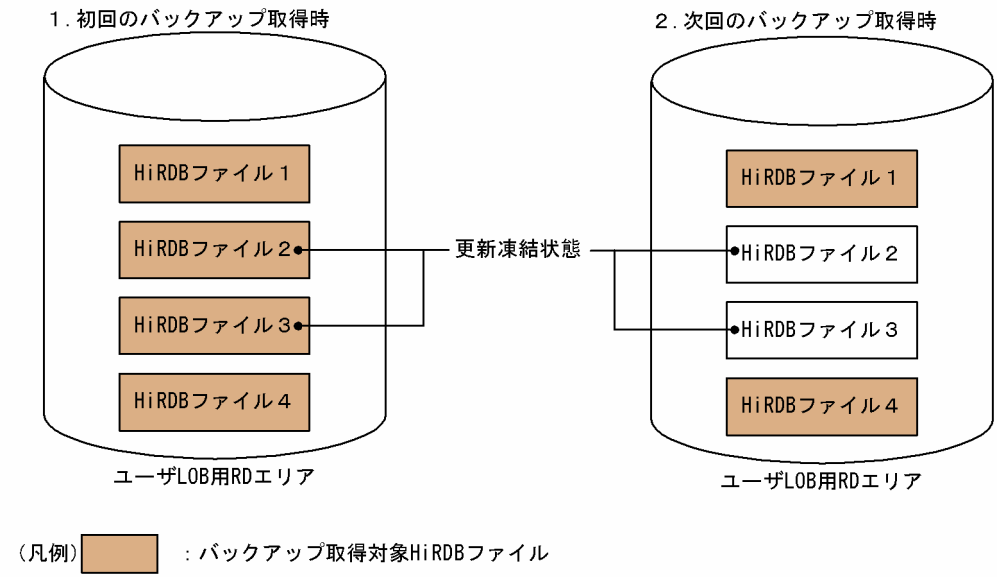
〔説明〕

- HiRDB ファイル 1～4 で構成されるユーザ LOB 用 RD エリアがあります。HiRDB ファイル 2～3 が満杯データページです。
- このユーザ LOB 用 RD エリアに更新凍結コマンドを実行すると，HiRDB ファイル 2～3 が**更新凍結状態**になります。更新凍結状態になった HiRDB ファイルは KFPH27024-I メッセージに表示されます。
- HiRDB ファイル 1 及び HiRDB ファイル 4 は**更新可能状態**になります。

(2)　更新凍結コマンドのバックアップ取得への利用方法

更新凍結コマンドのバックアップ取得への利用方法を次の図に示します。

図 7–7　更新凍結コマンドのバックアップ取得への利用方法



〔説明〕

バックアップ取得前に更新凍結コマンドを実行します。その結果，HiRDB ファイル 2～3 が更新凍結状態になりました。

1. 初回のバックアップ取得時には全 HiRDB ファイル（HiRDB ファイル 1～4）のバックアップを取得します。
2. HiRDB ファイル 2～3 は更新凍結状態のため，初回のバックアップ取得時と内容が変わっていません。したがって，次回のバックアップ取得時には HiRDB ファイル 2～3 のバックアップを取得する必要はありません。HiRDB ファイル 1 及び HiRDB ファイル 4 のバックアップだけを取得します。

備考

先頭 HiRDB ファイル（図「更新凍結コマンドのバックアップ取得への利用方法」の場合は HiRDB ファイル 1）には管理レコードがあるため，データ部が満杯になっても常に書き込みが発生します。このため，更新凍結状態になることはありません。したがって，先頭 HiRDB ファイルのバックアップは常に取得することになります。

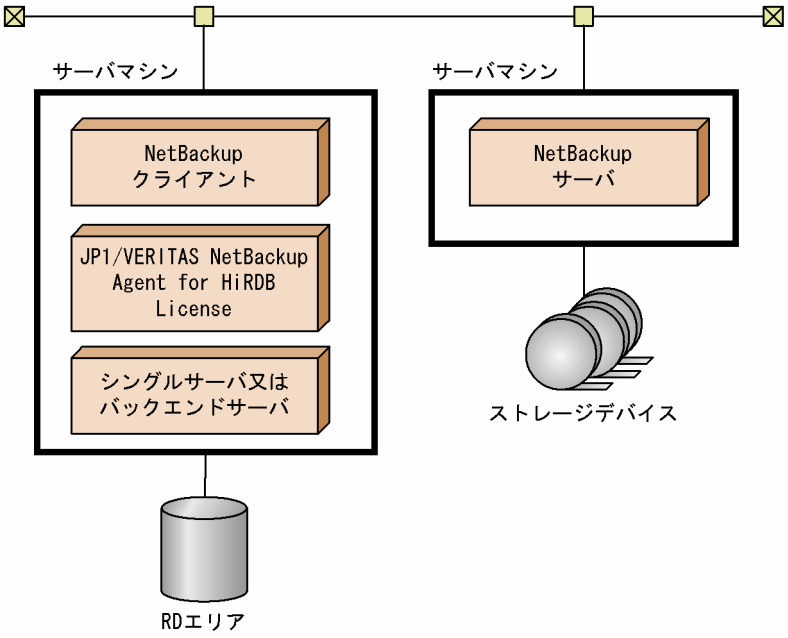
7.2.6　NetBackup 連携機能

NetBackup 連携機能を使用すると，データベース複写ユティリティ（pdcopy）又はデータベース回復ユティリティ（pdrstr）で使用するバックアップファイルを NetBackup サーバが管理する媒体上に作成できます。NetBackup 連携機能を使用する場合は **JP1/VERITAS NetBackup Agent for HiRDB License** が必要になります。

(1)　NetBackup 連携機能のシステム構成例

NetBackup 連携機能を使用すると，バックアップファイルを NetBackup サーバが管理する媒体上に作成できます。NetBackup 連携機能のシステム構成例を次の図に示します。

図 7–8　NetBackup 連携機能のシステム構成例



NetBackup 連携機能の詳細，及び前提製品については，マニュアル「JP1/VERITAS NetBackup v4.5 Agent for HiRDB License」を参照してください。

(2)　NetBackup 連携機能を使用したときの利点

NetBackup 連携機能を使用したときの利点を次に示します。

- **NetBackup でバックアップファイルを管理できる**

  バックアップファイルの出力先やバックアップ日時を NetBackup が管理するため，バックアップ取得時にバックアップファイル名を変える必要がありません。また，回復（リストア）するときも，ポリシー名とバックアップ日時を指定すると NetBackup が適切なバックアップファイルを選択してくれます。また，最新のバックアップファイルを使用する場合はバックアップ日時の指定も必要ありません。

- **HiRDB をインストールしていないサーバマシンのストレージデバイスにバックアップファイルを作成できる**

  HiRDB をインストールしていないサーバマシンのストレージデバイスにバックアップファイルを作成できます。NetBackup 連携機能を使用しない場合は，HiRDB をインストールしているサーバマシンのストレージデバイスだけにバックアップファイルを作成できます。

- **バックアップファイルに使用できる媒体が多い**

  NetBackup がサポートしている媒体をバックアップファイルにできます。NetBackup がサポートしている媒体については，NetBackup のマニュアルを参照してください。

7.3　表及びインデクスの再編成

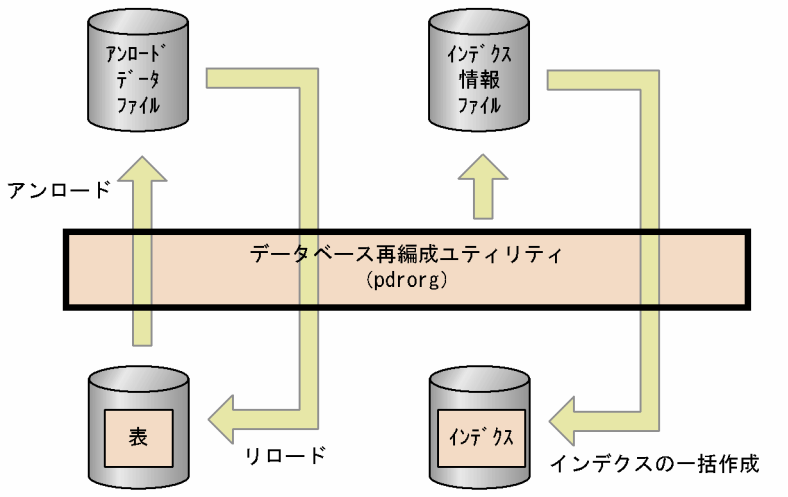
表の所有者又は DBA 権限保持者は，**データベース再編成ユティリティ（pdrorg）**で定期的に表及びインデクスの再編成を実行してください。

7.3.1　表の再編成

データを削除しても，そのデータを格納しているセグメント及びページは解放されません。このため，データの削除を繰り返すと，表中に無効領域が増えてデータの格納効率が悪くなり，データ量が増えていないのに RD エリアが容量不足になることがあります。

また，データの追加を繰り返すと，クラスタキーの値に近い場所のページにデータが格納されないため，データの入出力回数が増えます。このため，データを検索するときの性能が低下します。データベース再編成ユティリティで表の再編成を実行すると，データを再度格納し直すためこれらの現象を防げます。表の再編成を次の図に示します。

図 7–9　表の再編成



〔説明〕

- 表データをいったんアンロードファイルに吸い上げます。これを表データの**アンロード**といいます。その後，表にデータを格納し直します。これを表データの**リロード**といいます。これら全体の処理を**表の再編成**といいます。
- 表にインデクスが定義されていると，データをリロードするときにインデクス情報ファイルにインデクス情報が出力されます。その情報を基にして HiRDB がインデクスを一括作成します。これによって，インデクスも再編成されます。

(1)　表の再編成の実行単位

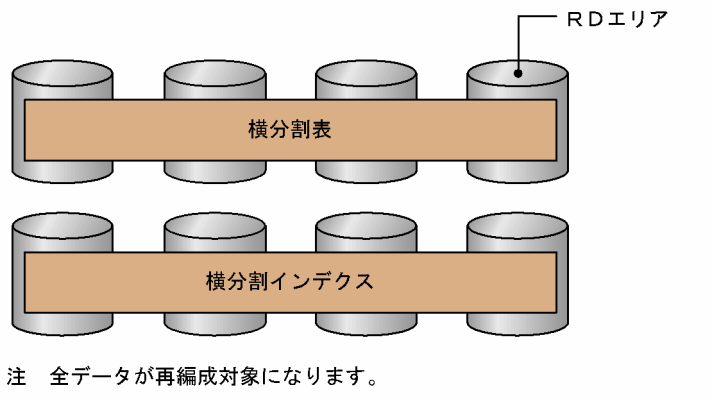
表の再編成は次に示す単位で実行できます。

- 表単位の再編成
- RD エリア単位の再編成
- スキーマ単位の再編成

(a)　表単位の再編成

再編成処理を表単位に行います。通常はこの方法を実施してください。データベース状態解析ユティリティの結果から，表全体を再編成する必要がある場合に表単位の再編成を実行します。表単位の再編成を次の図に示します。

図 7–10　表単位の再編成



注　全データが再編成対象になります。

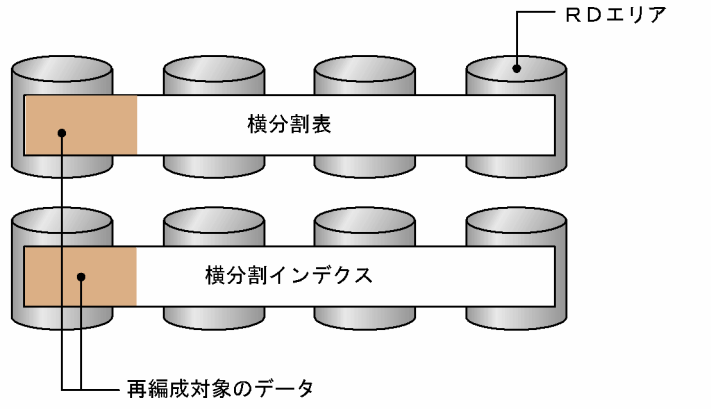
〔説明〕

データベース再編成ユティリティの-t オプションで再編成対象の表を指定します。

(b)　RD エリア単位の再編成

再編成処理を RD エリア単位に行います。この方法は表を横分割しているときだけ有効です。データベース状態解析ユティリティの結果から，横分割表のある部分だけを再編成すればよい場合に RD エリア単位の再編成を実行します。この場合，表単位の再編成に比べて，処理時間を短縮できます。RD エリア単位の再編成を次の図に示します。

図 7–11　RD エリア単位の再編成



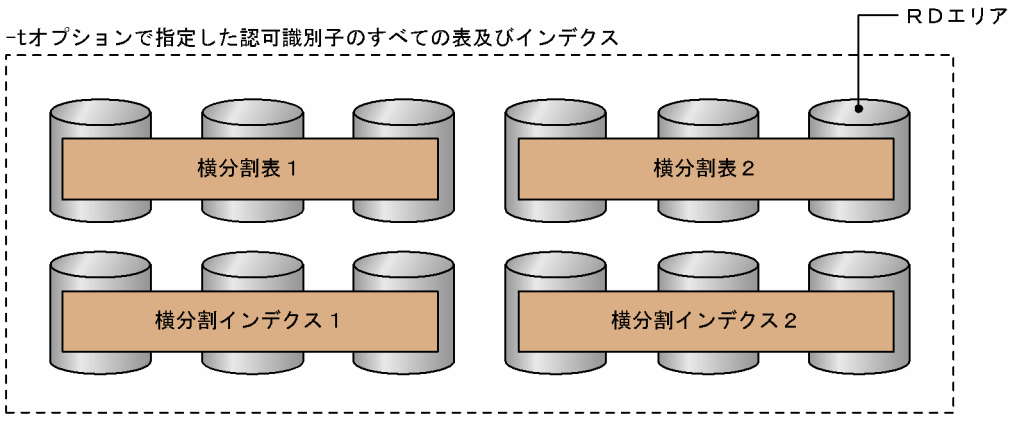
〔説明〕

データベース再編成ユティリティの-t オプションで再編成対象の表を指定して，かつ-r オプションで再編成対象の RD エリアを指定します。

(c)　スキーマ単位の再編成

スキーマ内のすべての表を一括して再編成します。所有する表の再編成を一括して行う場合にスキーマ単位の再編成を実行します。スキーマ単位の再編成を次の図に示します。

図 7–12　スキーマ単位の再編成



注　全データが再編成対象になります。

〔説明〕

データベース再編成ユティリティの-t オプションで再編成対象のスキーマの認可識別子を指定します。指定形式は**-t 認可識別子.all** です。

(2)　大量データを格納した表を再編成する場合

大量データを格納した表を再編成する場合，**同期点指定の再編成**を実施するかどうかを検討してください。

通常，表の再編成処理では全データの格納処理を完了するまでトランザクションを決着できません。このため，データベース再編成ユティリティ実行中はシンクポイントダンプを有効化できません。したがって，大量データの再編成処理中に HiRDB が異常終了すると，HiRDB の再開始処理に長い時間を必要とします。これを防ぐために，データ格納時（リロード処理時）に任意の件数で同期点を設定してトランザクションを決着できます。これを同期点指定の再編成といいます。

同期点指定の再編成をするには，データベース再編成ユティリティの option 文で**同期点行数**（何件データを格納したら同期点を取得するか）を指定してください。

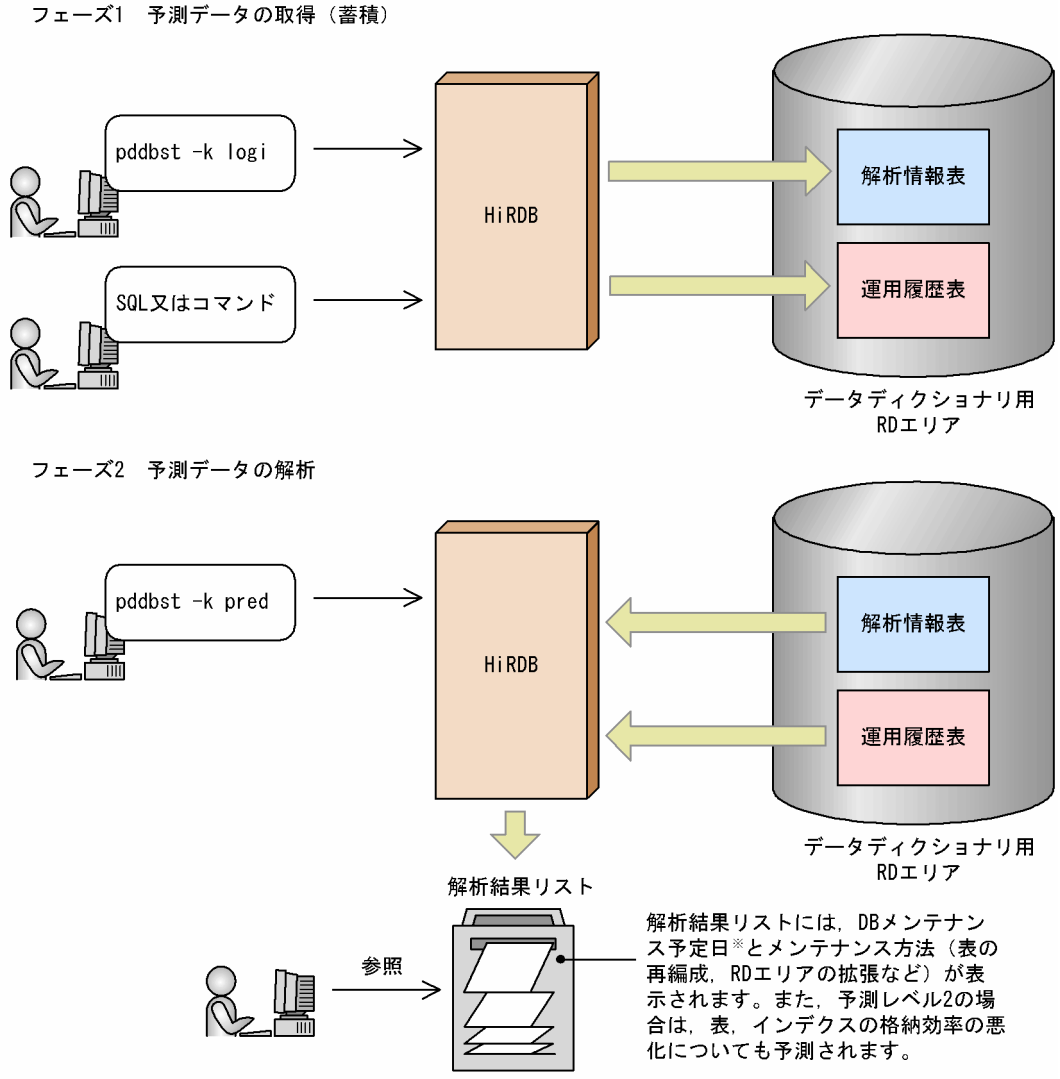
なお，データベース作成ユティリティでも同期点指定ができます。これを**同期点指定のデータロード**といいます。

(3)　再編成時期予測機能

表やインデクスの再編成をするかどうか，又は RD エリアを拡張するかどうかは，出力されたメッセージや，pddbst コマンドの実行結果から，再編成する表はどれか，いつ再編成する必要があるかなどユーザが総合的に判断する必要がありました。そのため，再編成する必要がない表を再編成したり，出力されたメッセージを見逃したため，再編成する必要がある表を再編成しなかったりする可能性がありました。

これらの運用を簡単にするために HiRDB が再編成時期の予測を行うようにしました。これを**再編成時期予測機能**といいます。再編成時期予測機能の概要を次の図に示します。

図 7–13　再編成時期予測機能の概要



注※　RD エリアのメンテナンスが必要になる予定日を **DB メンテナンス予定日**といいます。

再編成時期の予測は，次に示す二つのフェーズに分かれています。

- **フェーズ 1　再編成時期の予測データの取得**
  - pddbst コマンドを定期的に実行し，**解析情報表**にデータベースの解析結果を蓄積していきます。

- SQL 又はコマンドが実行されると，**運用履歴表**にデータベースの運用履歴情報が出力されます。

- **フェーズ 2　再編成時期の予測データの解析**

  解析情報表及び運用履歴表を入力情報にして，pddbst コマンドで再編成時期の予測データを解析します。ユーザは pddbst コマンドの実行結果を参照し，必要に応じて次に示すどれかの操作を行います。

  - pdrorg コマンドによる表又はインデクスの再編成
  - pdreclaim コマンドによる使用中空きページ及び空きセグメントの解放
  - pdmod コマンドによる RD エリアの拡張
  - pdmod コマンドによる RD エリアの自動増分
  - pdmod コマンドによる RD エリアの再初期化

また，再編成時期予測機能には次の二つのレベルがあります。pddbst でのデータベースの解析結果を蓄積する時間は，予測レベル 1 は比較的短時間で実行できますが，予測レベル 2 は長時間掛かってしまうことがあります。

- **予測レベル 1**

  RD エリアの容量不足を予測します。RD エリアの空き容量に余裕がある間，こまめに再編成を行わない運用をするユーザ向けです。

- **予測レベル 2**

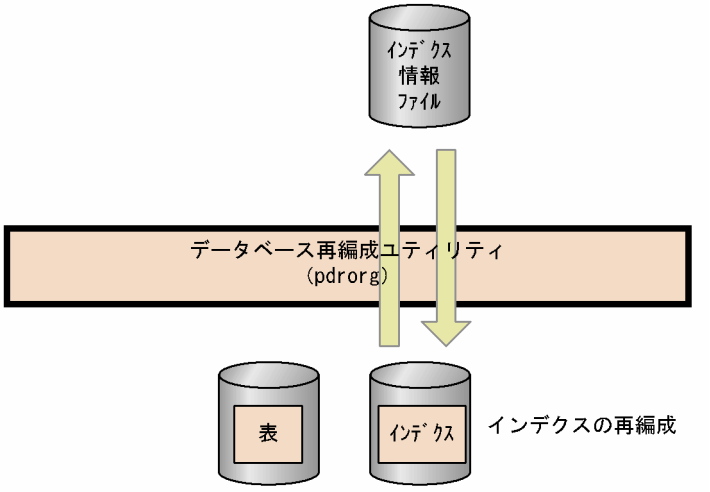
  RD エリアの容量不足のほかに，表及びインデクスの格納効率の悪化を予測します。RD エリアの空き容量に関係なく，表やインデクスへのアクセス効率が低下した場合は，すぐに再編成を行う運用をするユーザ向けです。

再編成時期予測機能については，マニュアル「HiRDB システム運用ガイド」，及び「HiRDB コマンドリファレンス」のデータベース状態解析ユティリティを参照してください。

7.3.2　インデクスの再編成

インデクスだけを再編成できます。インデクスの再編成を次の図に示します。

図 7–14　インデクスの再編成



〔説明〕

インデクスのキー情報を検索してインデクス情報ファイルを作成し，その情報を基にインデクスを再配置します。これを**インデクスの再編成**といいます。インデクスの再編成は，インデクス単位又はインデクス格納 RD エリア単位に実行できます。

(1)　適用範囲と適用基準

再編成の対象になるのは，通常のインデクスだけです。プラグインインデクスの再編成はできません。インデクスの再編成は，大量のデータの追加，削除，更新によって生じるインデクス格納ページの無効領域を解放する場合に実行します。

(2)　表の再編成との使い分け

- データの更新（UPDATE）が多い場合は，インデクスだけを再編成することをお勧めします。
- データの削除（DELETE）及び追加（INSERT）が多い場合は，表を再編成することをお勧めします。
- 表を再編成する時間の余裕がない場合，インデクスだけを再編成すればインデクス検索時間を短縮できます。

(3)　インデクスの再作成との違い

インデクスを再作成すると表データを検索しますが，インデクスを再編成しても表データを検索しません。そのため，インデクスの再編成は再作成に比べて処理時間が短くなり※，ソート処理も不要なため性能の面で優れています。

注※

次に示す条件を満たす場合に，処理時間が短くなります。

表格納 RD エリアの使用中ページ数 > インデクス格納 RD エリアの使用中ページ数

(4)　インデクスの再編成時の注意

同じ RD エリア内にある複数のインデクスを同時に再編成する場合は，インデクスを再編成する前に pdhold コマンドで RD エリアを閉塞状態にしてください。そして，インデクスの再編成の終了後に pdrels コマンドで RD エリアの閉塞状態を解除してください。

(5)　再編成の実行時間を短縮する方法

インデクスの再編成をするときに，データベースの更新ログを取得しなければ（ログレスモード又は更新前ログ取得モード），その分の処理時間が短縮されます。データベースの更新ログ取得方式は，データベース再編成ユティリティ（pdrorg）の -l オプションで指定します。

(6)　空きがない状態の RD エリア内のインデクスを再編成する場合

インデクスの再編成をするときは，ページ内の未使用領域の比率は CREATE TABLE 又は CREATE INDEX の PCTFREE オペランドの指定が適用されます。したがって，容量の空きがない状態の RD エリア内のインデクスを再編成すると，インデクスの再編成時に RD エリアの容量が不足することがあります。これを防ぐには，データベース再編成ユティリティ（pdrorg）の option 文で idxfree オペランドを指定して，CREATE TABLE 又は CREATE INDEX の PCTFREE オペランドで指定したページ内の未使用領域の比率を変更してください。

ただし，これは再編成時の暫定的な処置なので，通常の運用ではデータベース構成変更ユティリティ（pdmod）で RD エリアを拡張するようにしてください。

7.4　使用中空きページ及び使用中空きセグメントの再利用

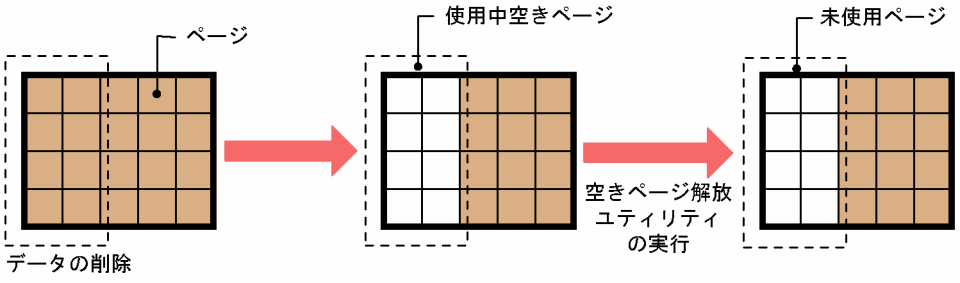
表及びインデクスの使用中空きページを未使用ページ化して再利用できます。同様に使用中空きセグメントを未使用セグメント化して再利用できます。なお，この節の説明を読む前にページ及びセグメントの状態について理解しておく必要があります。ページの状態については「ページの設計」を，セグメントの状態については「セグメントの設計」を参照してください。

7.4.1　使用中空きページの再利用

(1)　使用中空きページの解放

バッチジョブなどで表データを大量に削除すると，その表データを格納しているページ（データページ）の一部が使用中空きページになることがあります。また，インデクスを定義している場合は，インデクスのキー値を格納しているページ（インデクスページ）の一部が使用中空きページになります。**空きページ解放ユティリティ（pdreclaim）**を実行すると，この使用中空きページを未使用ページ化して再利用できます。これを**使用中空きページの解放**といいます。使用中空きページの解放を次の図に示します。

図 7–15　使用中空きページの解放



ポイント

- LOB 用 RD エリアに格納されているデータの使用中空きページは解放できません。
- プラグインインデクスの使用中空きページは解放できません。

使用中空きページの解放については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(2)　使用中空きページを解放したときの効果

(a)　表の使用中空きページを解放したときの効果

表の使用中空きページを解放したときの効果を次の表に示します。

表 7–4　表の使用中空きページを解放したときの効果

効果がある項目	説明	効果の度合い
表を再編成するサイクルを長くできる	使用中空きページを再利用できるためデータの格納効率が良くなります。このため，表を再編成するサイクルを長くできます。	○
大量データ検索時の性能が向上する	使用中空きページは使用中ページのため検索処理時のサーチ対象になりますが，未使用ページはサーチ対象になりません（サーチ処理がスキップされます）。その分，検索処理の性能が向上します。特に，大量データを検索するときに効果が出ます。	△
INSERT 及び UPDATE 時の性能が向上する	使用中ページにデータを格納するとき，データの格納に必要な連続した空き領域を確保できないと，HiRDB は**ページコンパクション**という処理を行います。ページコンパクションとは，データの格納に必要な連続した空き領域を確保するために行われるページ内のデータ詰め替え処理のことです。 使用中空きページを解放するときにページコンパクションも同時に行います。このため，INSERT 及び UPDATE 処理の延長で行われるページコンパクションが不要になり，その分処理性能が向上します。なお，ページコンパクションの処理対象ページは，満杯ページ及び使用中空きページを除いた使用中ページになります。	△
分岐行の INSERT 及び UPDATE 時のエラー発生を抑えられる	未使用ページがない状態で分岐行の INSERT 及び UPDATE を実行すると，エラー（KFPA11756-E メッセージ）になります。使用中空きページの解放で未使用ページが増えるため，このエラーの発生を抑えられます。	△

（凡例）

○：効果があります。

△：条件によって効果の度合いが変わります。

(b)　インデクスの使用中空きページを解放したときの効果

インデクスの使用中空きページを解放したときの効果を次の表に示します。

表 7–5　インデクスの使用中空きページを解放したときの効果

効果がある項目	説明	効果の度合い
インデクス格納 RD エリアの容量不足の発生を抑えられる	空きページ（使用中空きページ）があるのに領域不足になる場合は使用中空きページを解放してください。なお，キー値の更新又は削除が多い場合でもインデクス格納 RD エリアの容量不足の発生を抑えられます。	◎
インデクスを再編成するサイクルを長くできる	使用中空きページを再利用できるためデータの格納効率が良くなります。このため，インデクスを再編成するサイクルを長くできます。	○
インデクスを使用した大量データ検索時の性能が向上する	使用中空きページは使用中ページのため検索処理時のサーチ対象になりますが，未使用ページはサーチ対象になりません（サーチ処理がスキップされます）。その分，検索処理の性能が向上します。特に，大量データを検索するときに効果が出ます。	△

（凡例）

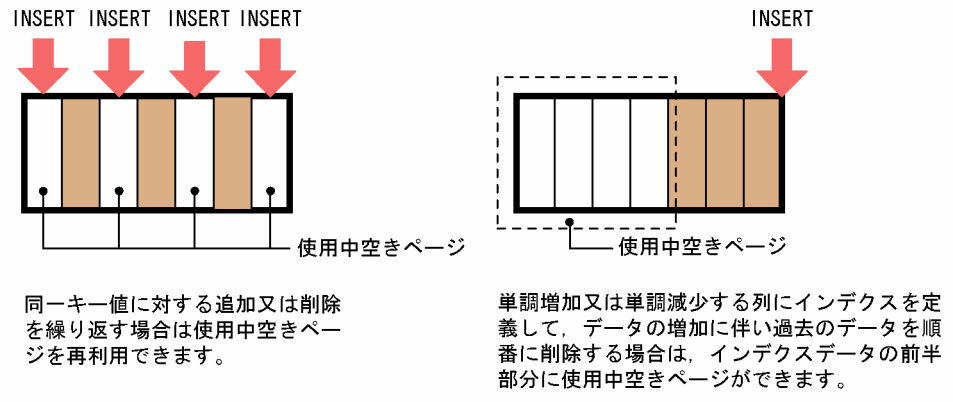
◎：特に効果があります。

○：効果があります。

△：条件によって効果の度合いが変わります。

特に，削除したキー値を再度登録しない場合にこの機能を適用すると効果があります。同一キー値の追加又は削除を繰り返す場合は使用中空きページを再利用するため，使用中空きページが大量に発生することはありません。しかし，単調増加又は単調減少する列（日付け，通番など）にインデクスを定義してデータの増加に伴い過去のデータを順番に削除する場合は，インデクスページの前半部分に再利用されない使用中空きページが大量に発生します。インデクスページに使用中空きページが作成される処理を次の図に示します。

図 7–16　インデクスページに使用中空きページが作成される処理



なお，使用中空きページの解放後は解放したページにキー値を格納していくため，データの格納効率が良くなります。

(3)　表又はインデクスの再編成との違い

性能面及びデータの格納効率という点から見ると，使用中空きページの解放より表又はインデクスの再編成の方が優れています。しかし，使用中空きページの解放の場合は，ユティリティの実行中に処理対象表又はインデクスをアクセスできます。再編成の場合は，ユティリティの実行中に処理対象表又はインデクスをアクセスできません。このため，使用中空きページの解放の場合は業務を中断する必要がありません。

再編成をするか，使用中空きページを解放するかはデータベース状態解析ユティリティの実行結果から判断してください。判断基準を次に示します。

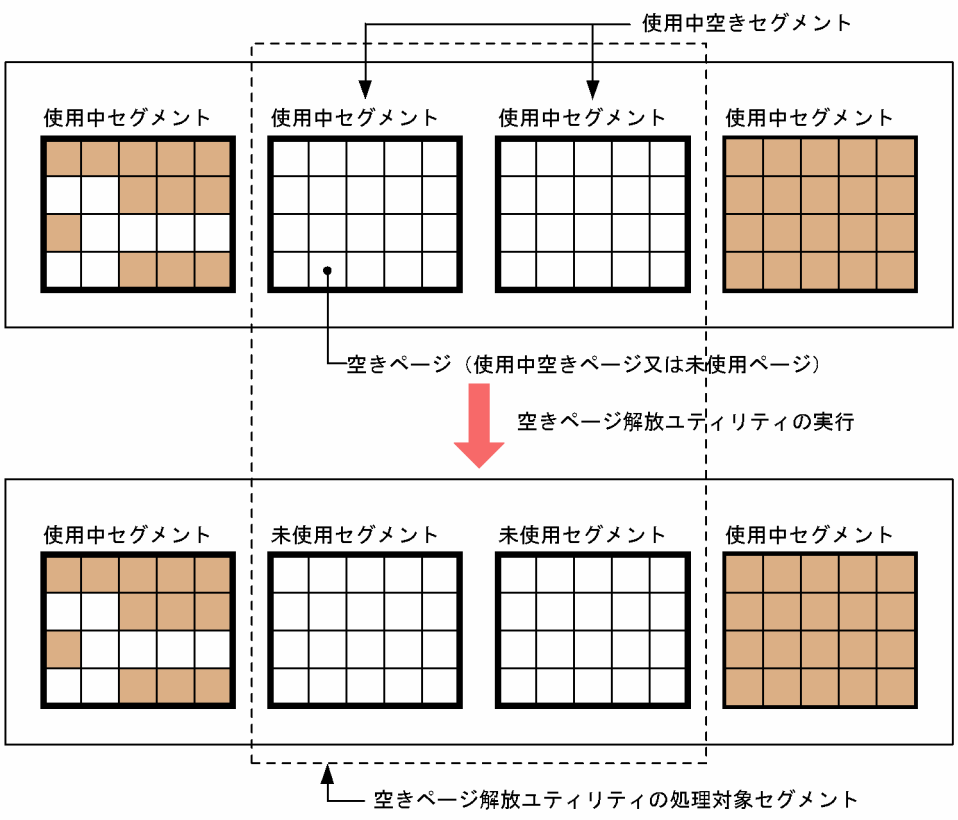
- 使用中空きページが大量にある場合は使用中空きページを解放してください。
- セグメント内の空きページ比率（CREATE TABLE の PCTFREE オペランドの値）と懸け離れたページ使用率の使用中ページが大量にある場合は再編成をしてください。

7.4.2　使用中空きセグメントの再利用

(1)　使用中空きセグメントの解放

空きページ解放ユティリティを実行すると，使用中空きセグメントを未使用セグメント化して再利用できます。これを**使用中空きセグメントの解放**といいます。使用中空きセグメントの解放を次の図に示します。

図 7–17　使用中空きセグメントの解放



使用中空きセグメントの解放については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(2)　使用中空きセグメントを解放したときの効果

一度使用されたセグメントは使用している表（又はインデクス）だけが使用でき，ほかの表は使用できません。使用中空きセグメントを解放して使用中空きセグメントを未使用セグメント化すると，その未使用セグメントをほかの表が使用できるようになります。

7.5　RD エリアの追加，拡張，及び移動

業務規模の拡大に比例して，データベースのデータ量も増加していきます。当初の見積もりよりデータベースが大きくなっても，HiRDB では後からデータベースを追加，拡張，又は移動できます。また，HiRDB/パラレルサーバの場合，後からサーバマシンを増やしてそのサーバマシンにデータベースを追加又は移動することもできます。

HiRDB ではデータベースのデータ量の増加に対応して次に示す機能を提供しています。

- RD エリアの追加
- RD エリアの拡張
- RD エリアの自動増分
- RD エリアの移動（HiRDB/パラレルサーバ限定）

7.5.1　RD エリアの追加

データベース構成変更ユティリティ（pdmod コマンド）の **create rdarea 文**で，RD エリアを追加できます。新規の表を作成する場合などに RD エリアを追加してください。

なお，追加した RD エリアを使用するには，グローバルバッファを割り当てる必要があります。したがって，定義されているグローバルバッファを **pdbufls** コマンドで調べる必要があります。

7.5.2　RD エリアの拡張

表にデータを追加していくと，RD エリアの残容量が少なくなります。残りの容量が少なくなったら，データベース構成変更ユティリティ（pdmod コマンド）の **expand rdarea 文**で RD エリアを拡張できます。

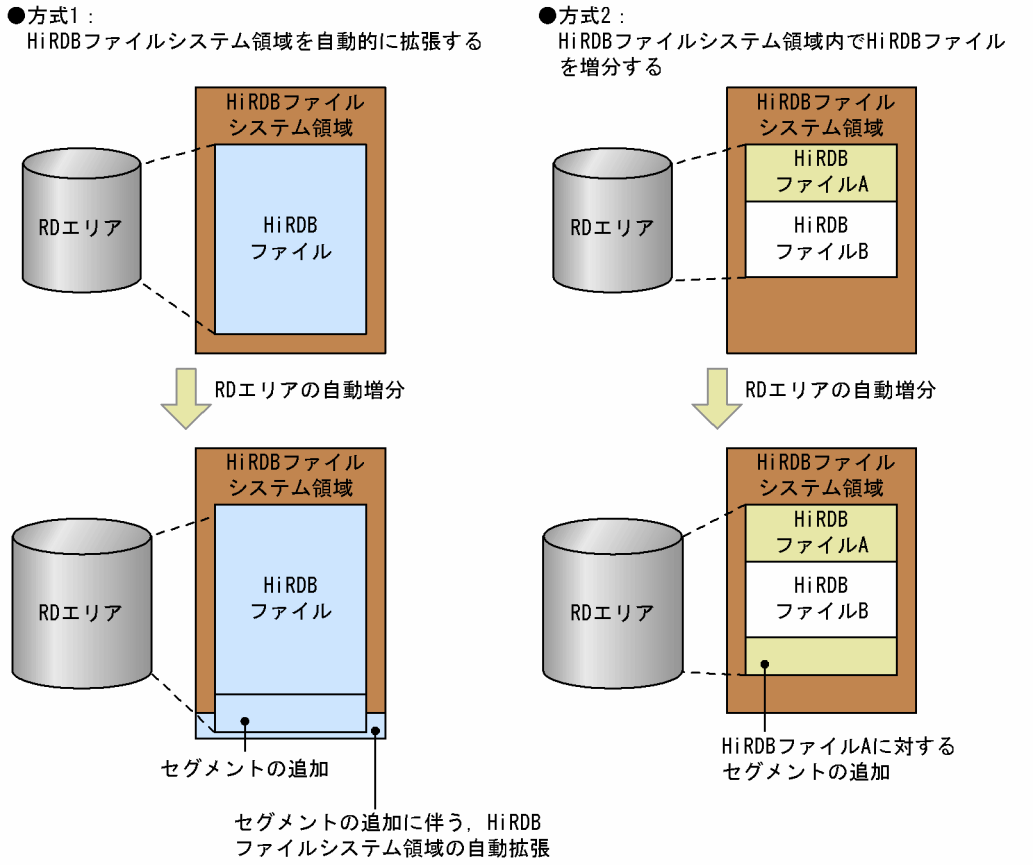
7.5.3　RD エリアの自動増分

RD エリアの容量不足が近付いたときに，その RD エリアの HiRDB ファイルにセグメントを追加して RD エリアの容量を自動的に拡張します。これを **RD エリアの自動増分**といいます。RD エリアの自動増分には次の二つの方式があります。

1. HiRDB ファイルシステム領域を自動的に拡張する方式
2. HiRDB ファイルシステム領域内で HiRDB ファイルを増分する方式

RD エリアの自動増分を次の図に示します。

図 7–18　RD エリアの自動増分



〔説明〕

方式 1 では，セグメントの追加で，HiRDB ファイルシステム領域サイズの上限を超える場合，HiRDB ファイルシステム領域を必要な分だけ自動的に拡張します。この方式では，一つの HiRDB ファイルシステム領域には，HiRDB ファイルを一つだけ作成できます。

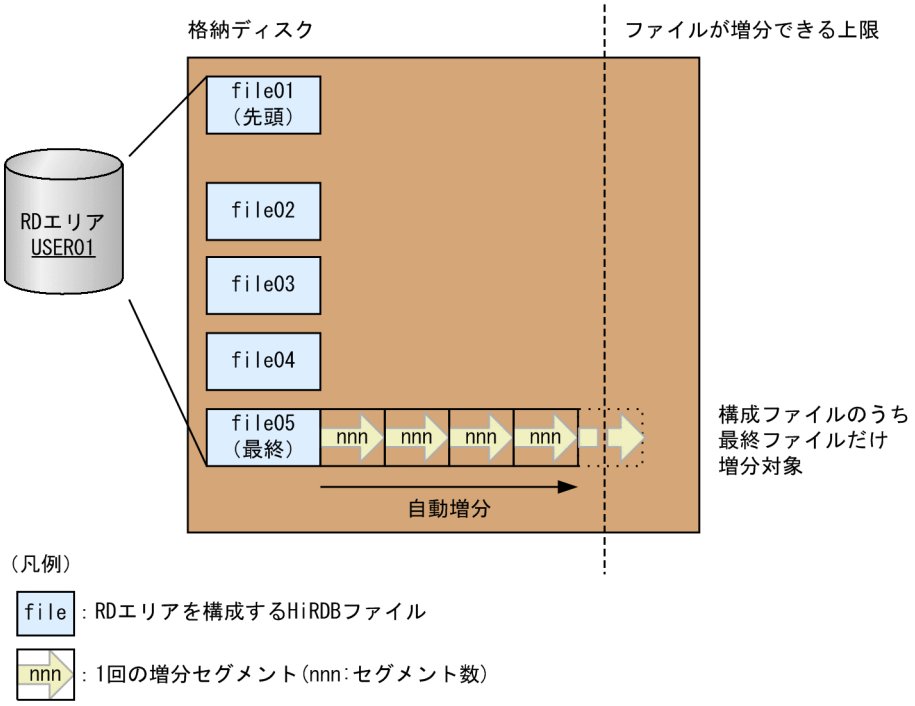
方式 2 では，HiRDB ファイルシステム領域サイズ内で HiRDB ファイルを拡張又は追加し，HiRDB ファイルシステム領域の上限まで自動増分します。

RD エリア構成ファイルが複数ある場合，自動増分の対象となるファイルをシステム共通定義 **pd_rdarea_extension_file** で選択できます。**pd_rdarea_extension_file** オペランドについては，マニュアル「HiRDB システム定義」を参照してください。

1. **pd_rdarea_extension_file=last** の場合

   自動増分の対象ファイルは，RD エリアを構成する HiRDB ファイルのうち，最終ファイルが対象となります。

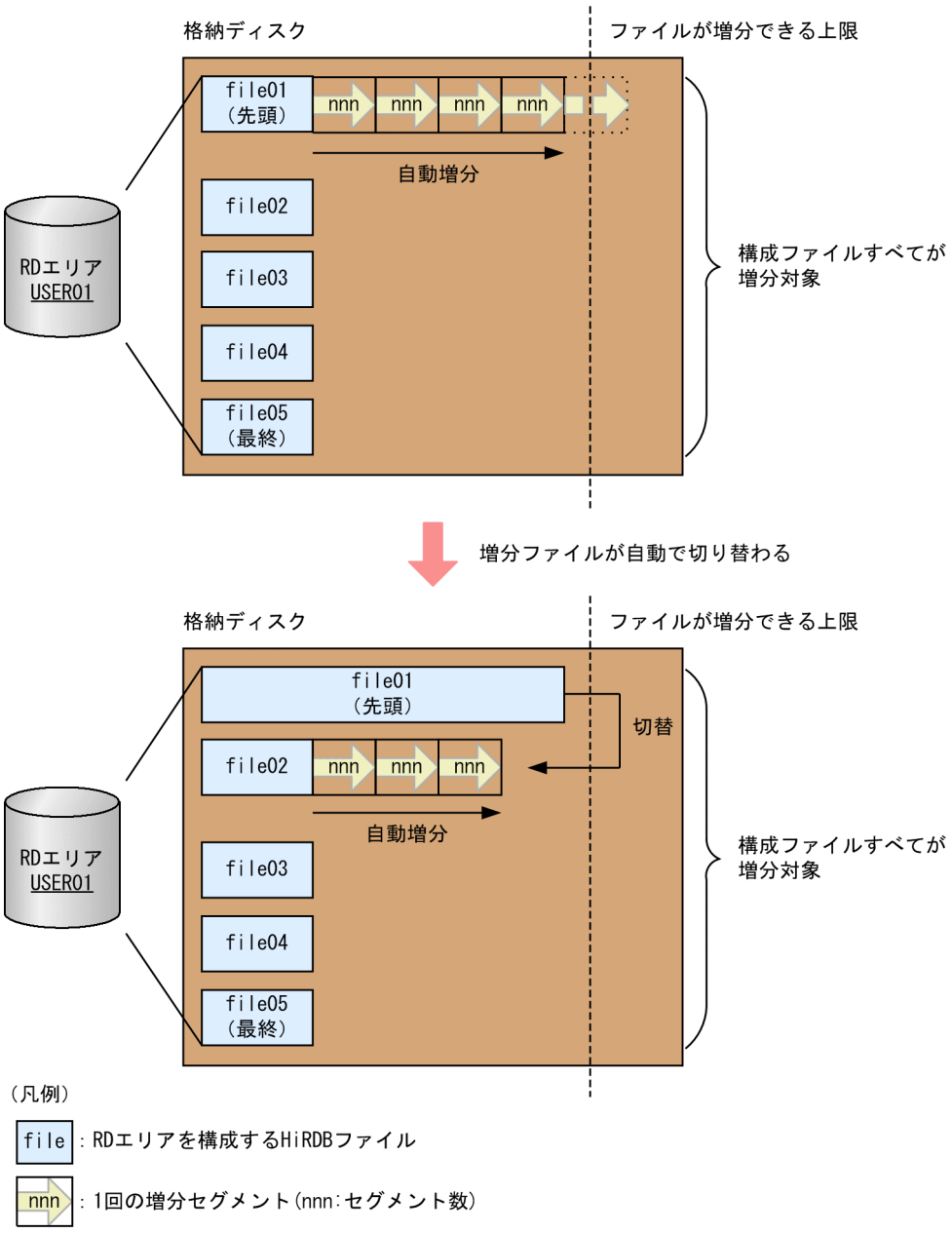
図 7–19　pd_rdarea_extension_file=last の方式



2. **pd_rdarea_extension_file=all** の場合

自動増分の対象ファイルは，RD エリアを構成する HiRDB ファイルすべてが対象となります。

図 7–20　pd_rdarea_extension_file=all の方式



RD エリアの自動増分の詳細については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(1)　自動増分を適用できる RD エリア

RD エリアの自動増分を適用できる RD エリアを次に示します。

表 7–6　自動増分機能を使用できる RD エリア種別と自動増分の対象 HiRDB ファイル

#	RD エリア種別		自動増分適用可否	自動増分の対象 HiRDB ファイル	
				all※	last※
1	ユーザ用 RD エリア	一時表用 RD エリア以外	○	構成ファイルすべて	最終ファイル

#	RD エリア種別		自動増分適用可否	自動増分の対象 HiRDB ファイル	
				all※	last※
2		一時表用 RD エリア		最終ファイル	
3	データディクショナリ用 RD エリア				
4	レジストリ用 RD エリア				
5	ユーザ LOB 用 RD エリア				
6	データディクショナリ LOB 用 RD エリア				
7	レジストリ LOB 用 RD エリア				
8	マスタディレクトリ用 RD エリア		×	−	−
9	データディレクトリ用 RD エリア				
10	リスト用 RD エリア				

（凡例）

○：自動増分を指定できる

×：自動増分を指定できない

−：自動増分の対象ファイルなし

注※

システム共通定義 pd_rdarea_extension_file オペランドの指定値です。

(2)　自動増分の契機

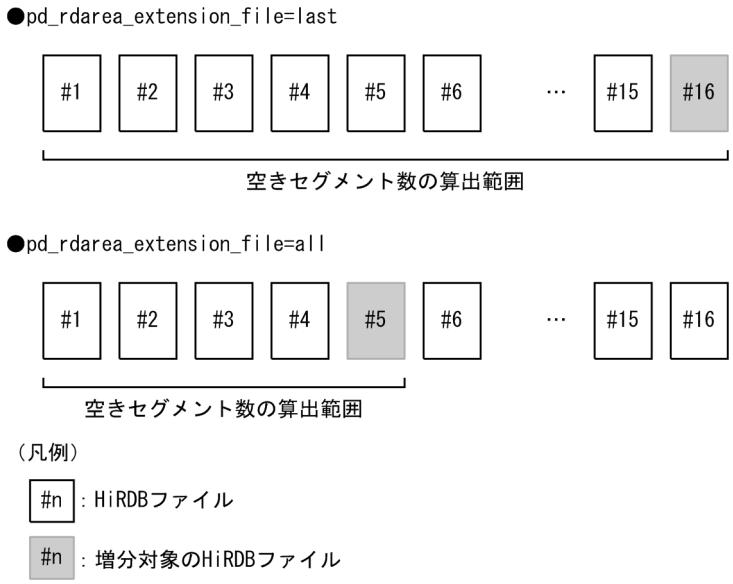
HiRDB は，次に示す RD エリアの自動増分契機になると，RD エリアを自動増分します。

- RD エリア内の空きセグメント数の合計が 1 回の自動増分で増分するセグメント数以下になったとき
  例えば，1 回の自動増分セグメント数を 50 セグメントとしている場合，空きセグメント数が 50 セグメント以下になったときに自動増分します。
- RD エリア内に空きセグメントがなく，新しいセグメントを確保できないとき

自動増分契機は，**pd_rdarea_extension_timing** オペランドで指定できます。詳細については，マニュアル「HiRDB システム定義」を参照してください。

自動増分契機を判断するために空きセグメント数を算出する範囲を次の図に示します。

図 7–21　空きセグメント数を算出する範囲



(3)　RD エリアの自動増分方式の選択基準

RD エリアの自動増分を適用する場合，基本的には方式 1 をお勧めします。方式 1 では，HiRDB ファイルの最大サイズ（64 ギガバイト）まで自動的に HiRDB ファイルシステム領域を拡張するため，領域の見積もりやデータベースの拡張が容易になります。

ただし，方式 1 には次の制限事項があるため，適用できないシステムの場合は，方式 2 を選択してください。

- 一つの RD エリアに一つの HiRDB ファイルシステム領域を割り当てるため，RD エリア数と同じ数の HiRDB ファイルシステム領域を作成する必要がある。
- ディスクに空きがあれば，HiRDB ファイルシステム領域が 64 ギガバイトまで自動的に拡張するため，ディスク使用量の上限を設定できない。

(4)　自動増分の対象ファイルの選択基準

次に示す運用をする場合，**pd_rdarea_extension_file** に all を指定してください。

- RD エリア作成時に複数の HiRDB ファイル構成とし，すべての構成ファイルを自動増分させたい。

all を指定した場合，最大 1 テラバイト（64 ギガバイト[構成ファイルの上限サイズ]×16[最大構成ファイル数]）まで RD エリアを自動的に拡張できます。

ただし，all 指定には次の制限事項があるため，適用できないシステムの場合は，last を指定してください。

- すべての構成ファイル配置先に，構成ファイルが自動増分によって拡張できるディスク容量をあらかじめ用意する必要がある。

(5)　自動増分の設定方法

自動増分の設定手順を次に示します。詳細は，マニュアル「HiRDB システム運用ガイド」の「RD エリアの自動増分」を参照してください。

(a)　方式 1 の場合

この方式の場合，RD エリアの自動増分で HiRDB ファイルシステム領域サイズの上限を超える場合，HiRDB ファイルの最大サイズ（64 ギガバイト）まで自動的に HiRDB ファイルシステム領域を拡張します。

(b)　方式 2 の場合

この方式の場合，HiRDB ファイルシステム領域の上限まで自動増分します。自動増分で，HiRDB ファイルシステム領域に空きがないと，自動増分できません。この場合，RD エリアを拡張するか，RD エリア内の表及びインデクスを再編成してください。また，エクステント数が上限値である 24 を超えた場合，HiRDB ファイルシステム領域のエクステントを統合して一つにするか，又は RD エリアを拡張してください。エクステント，及び HiRDB ファイルシステム領域を統合する手順については，マニュアル「HiRDB システム運用ガイド」を参照してください。

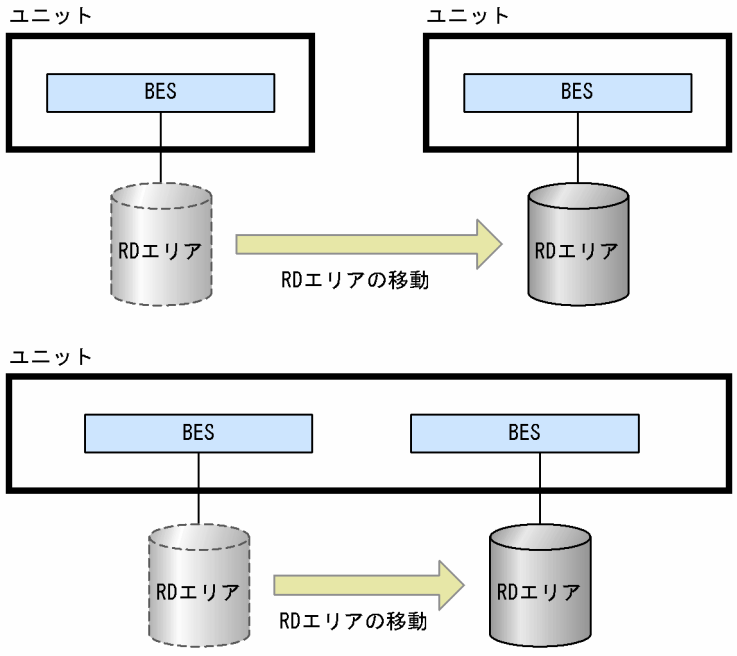
7.5.4　RD エリアの移動（HiRDB/パラレルサーバ限定）

データベース構成変更ユティリティ（pdmod コマンド）の **move rdarea 文**で，RD エリアをほかのバックエンドサーバに移動できます。RD エリアの移動機能は HiRDB/パラレルサーバ限定の機能です。移動できる RD エリアを次に示します。

- ユーザ用 RD エリア
- ユーザ LOB 用 RD エリア

RD エリアの移動を次の図に示します。

図 7–22　RD エリアの移動



RD エリアの移動については，マニュアル「HiRDB システム運用ガイド」を参照してください。

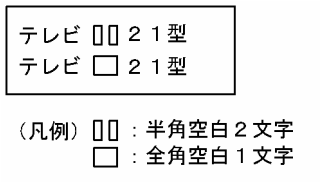
7.6　空白変換機能

7.6.1　空白変換レベルの概要

データを比較するとき，全角の空白 1 文字と半角の空白 2 文字は異なるデータとして認識されます。したがって，全角の空白 1 文字と半角の空白 2 文字が表データ中に混在していると，検索結果が不正になることがあります。

（例）

次に示すデータは異なるデータと認識されます。



空白変換機能を使用すると，表データ中に混在している全角の空白と半角の空白を統一できます。

なお，ここでいう全角空白とは次に示すコードのことです。半角空白 2 文字とは，X'2020'のことです。

- シフト JIS 漢字コードの場合：X'8140'
- EUC 日本語漢字コードの場合：X'A1A1'【UNIX 版限定】
- EUC 中国語漢字コードの場合：X'A1A1'
- 中国語漢字コード（GB18030）の場合※：X'A1A1'
- Unicode（UTF-8）又は Unicode（IVS 対応 UTF-8）の場合※：X'E38080'

注※

文字コードが Unicode（UTF-8），Unicode（IVS 対応 UTF-8），又は中国語漢字コード（GB18030）の場合は，NCHAR 及び NVARCHAR を使用できません。

(1)　空白変換レベル

空白変換機能には三つのレベルがあり，次の表に示すように空白を変換します。

表 7–7　空白変換レベル

レベル	説明
レベル 0	空白変換をしません。
レベル 1	操作系 SQL での定数，埋込み変数又は？パラメタのデータの空白及びユティリティで格納するデータの空白を次のように変換します。 • 文字列定数を各国文字列定数とみなした場合，半角空白 2 文字を全角空白 1 文字に変換します。なお，半角空白が 1 文字単独で現れる場合は変換しません。

レベル	説明
	• 文字列定数を混在文字列定数とみなした場合，全角空白 1 文字を半角空白 2 文字に変換します。 • 各国文字列型の列へのデータの格納時及び各国文字列型の値式との比較時は，埋込み変数又は？パラメタの半角空白 2 文字を全角空白 1 文字に変換します。なお，半角空白が 1 文字単独で現れる場合は変換しません。 • 混在文字列型の列へのデータの格納時及び混在文字列型の値式との比較時は，埋込み変数又は？パラメタの全角空白 1 文字を半角空白 2 文字に変換します。
レベル 3	空白変換レベル 1 の処理に加えて次に示す処理が加わります。 • 各国文字列型の値式のデータを検索するときに，全角空白 1 文字を半角空白 2 文字に変換します。

レベル 1 の処理方式，及びレベル 3 の処理方式を次の図に示します。

図 7–23　レベル 1 の処理方式

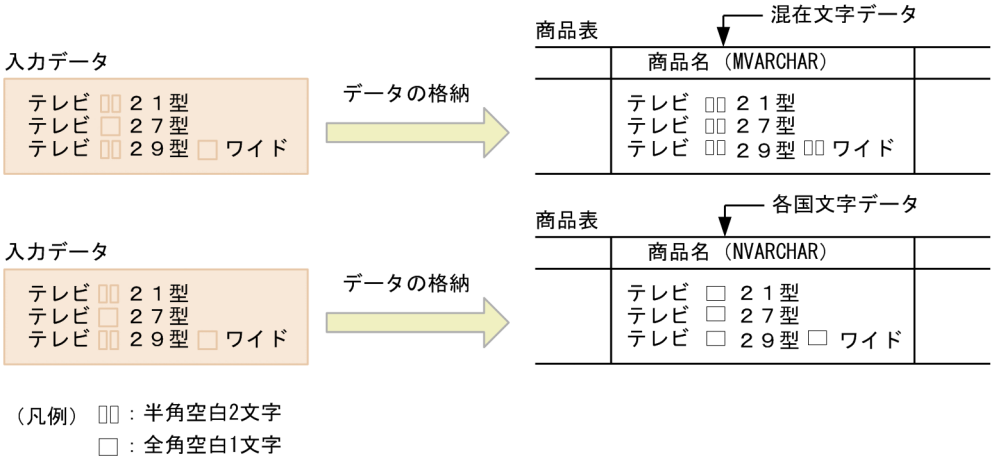
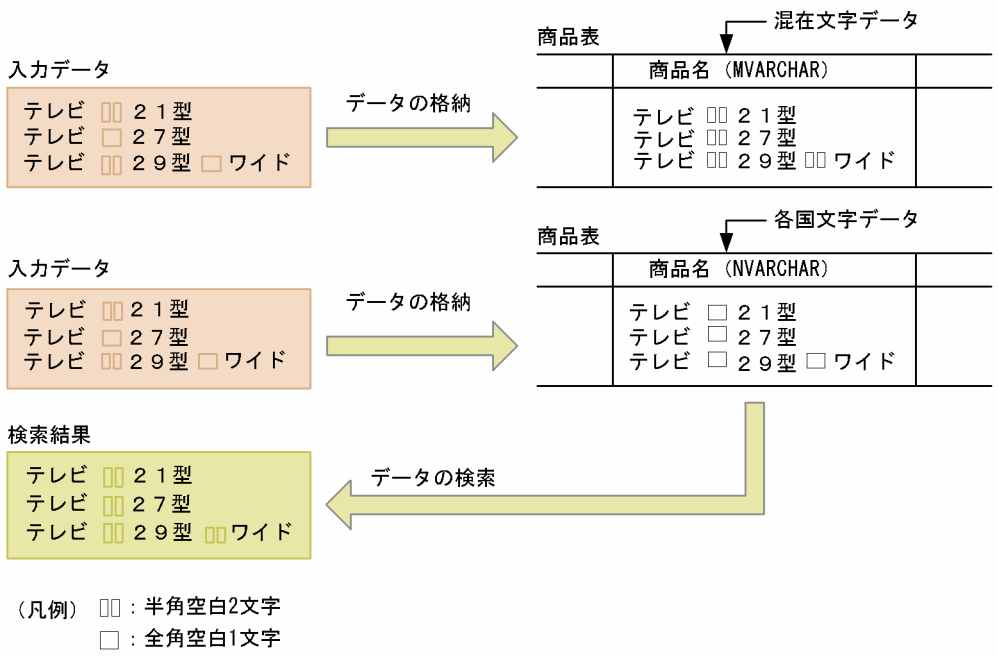


図 7–24　レベル 3 の処理方式



(2)　空白変換レベルの設定方法

空白変換レベルは次に示すオペランドで指定できます。

- システム共通定義の **pd_space_level** オペランド
- クライアント環境定義の **PDSPACELVL** オペランド
- データベース作成ユティリティ（pdload）の option 文の **spacelvl** オペランド
- データベース再編成ユティリティ（pdrorg）の option 文の **spacelvl** オペランド

空白変換機能の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

7.7　DECIMAL 型の符号正規化機能

DECIMAL 型，日間隔型及び時間隔型のデータ形式は，値の整数部分と符号部分で構成される符号付きパック形式です。通常は HiRDB では符号付きパック形式データの符号部分として X'C'（正），X'D'（負），X'F'（正）を有効な値として，UAP 又はユティリティから入力される符号をそのままデータベースに格納※しています。また，＋ 0（符号部 X'C'又は X'F'）と−0（符号部 X'D'）は異なる値としています。

DECIMAL 型の符号正規化機能を使用すると，DECIMAL 型，日間隔型及び時間隔型の符号付きパック形式の符号部を変換できます。

注※

SQL 実行中の型変換や演算などで符号が変換される場合があります。また，複数列インデクスを使用した場合に符号が変換される場合があります。

7.7.1　符号付きパック形式の符号部の仕様

HiRDB では符号付きパック形式の符号部の仕様が次の表に示すようになっています。

表 7–8　符号付きパック形式の符号部の仕様

符号部	意味
X'C'	正の値を示しています。
X'D'	負の値を示しています。
X'F'	正の値を示しています。

7.7.2　符号付きパック形式の符号部の変換規則

DECIMAL 型の符号正規化機能を使用すると，データを入力したときに符号付きパック形式の符号部を次の表に示す規則に従って変換します。この符号部を変換することを，**符号部を正規化する**といいます。符号部を正規化すると＋ 0 と−0 を同じ値として処理できます。

表 7–9　符号付きパック形式の符号部の変換規則（0 データ以外の場合）

埋込み変数のデータの符号部	正規化しない場合	正規化する場合
X'A'	エラー	X'C'に変換
X'B'	エラー	X'D'に変換
X'C'	無変換	無変換
X'D'	無変換	無変換
X'E'	エラー	X'C'に変換

埋込み変数のデータの符号部	正規化しない場合	正規化する場合
X'F'	無変換	X'C'に変換
X'0'～X'9'	エラー	エラー

表 7–10　符号付きパック形式の符号部の変換規則（0 データの場合）

0 データの符号部	正規化しない場合	正規化する場合
X'A'	エラー	X'C'に変換
X'B'	エラー	
X'C'	無変換	
X'D'	無変換	
X'E'	エラー	
X'F'	無変換	

7.7.3　適用基準

符号部に仕様差がある UAP を使用する場合に，DECIMAL 型の符号正規化機能を使用するとよいケースがあります。この場合，符号変換規則をよく確認してから DECIMAL 型の符号正規化機能を使用してください。

例えば，XDM/RD E2 の UAP を HiRDB に移行した場合に，DECIMAL 型の符号正規化機能を使用するとよいケースがあります。XDM/RD E2 と HiRDB は DECIMAL 型の符号部に仕様差があります。

7.7.4　環境設定

DECIMAL 型の符号正規化機能を使用するには，システム共通定義で **pd_dec_sign_normalize=Y** を指定します。

DECIMAL 型の符号正規化機能は，なるべく HiRDB の新規導入時に指定してください。既に HiRDB を運用しているときに符号部を正規化するには，DECIMAL 型を定義した表のデータをデータロードし直す必要があります。

DECIMAL 型の符号正規化機能の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

8 障害対策に関する機能

この章では，障害対策に関する機能について説明します。

8.1　系切り替え機能

クラスタソフトウェア製品と連携すると，システムの信頼性向上，稼働率向上を目的とした**系切り替え機能**が使用できるようになります。ここでは，系切り替え機能の概要について説明します。系切り替え機能の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

8.1.1　系切り替え機能とは

業務処理中の HiRDB に障害が発生した場合，待機用の HiRDB に業務処理を自動的に切り替えて運用できます。これを**系切り替え機能**といいます。業務処理が中断するのは障害発生時から待機用の HiRDB に処理が切り替わるまでです。障害発生時のシステム停止時間をなるべく短くしたい場合に系切り替え機能を使用します。

系切り替え機能には，スタンバイ型系切り替え機能とスタンバイレス型系切り替え機能があります。また，系切り替え機能は，監視対象とする障害によってモニタモード又はサーバモードで運用します。系切り替え機能と運用方法の組み合わせを，次の表に示します。

表 8–1　系切り替え機能の種類と運用方法の組み合わせ

系切り替え機能の種類		運用方法	
		モニタモード	サーバモード
スタンバイ型系切り替え機能	通常の系切り替え	○	○
	ユーザサーバホットスタンバイ	×	○
	高速系切り替え	×	○
スタンバイレス型系切り替え機能※	1：1 スタンバイレス型系切り替え	×	○
	影響分散スタンバイレス型系切り替え	×	○

（凡例）

○：運用できます。

×：運用できません。

注※

スタンバイレス型系切り替え機能を使用する場合は，HiRDB Advanced High Availability が必要です。

(1)　スタンバイ型系切り替え機能とは

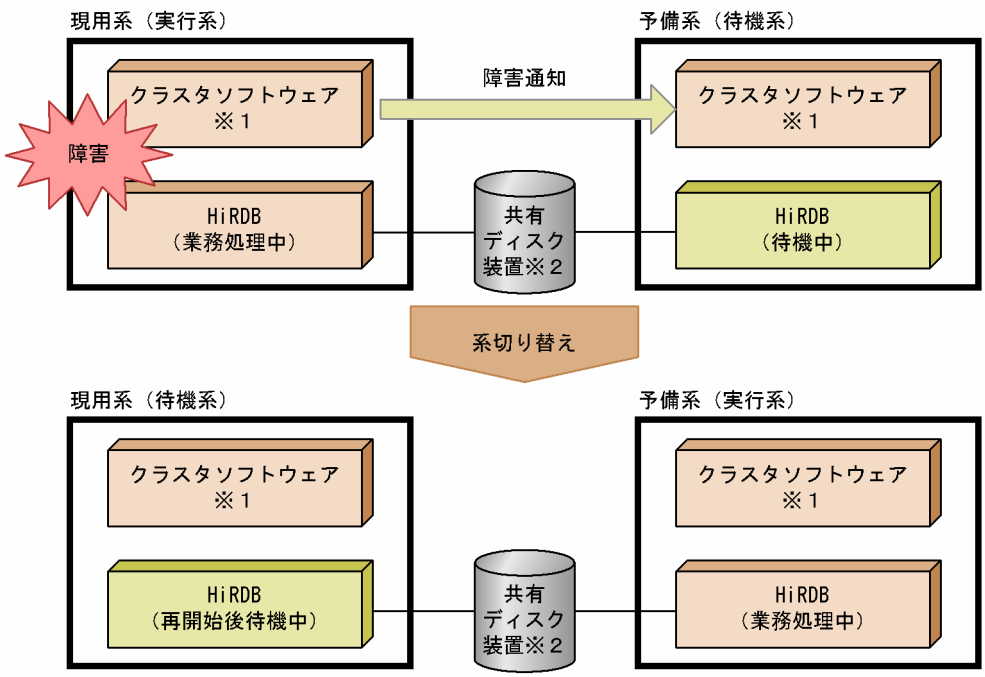
業務処理中の HiRDB のほかに待機用の HiRDB を準備して，業務処理中の HiRDB に障害が発生した場合，待機用の HiRDB に業務処理を自動的に切り替えます。これを**スタンバイ型系切り替え機能**といいます。

スタンバイ型系切り替え機能は複数のサーバマシンを使用したクラスタシステムの構成で実現します。HiRDB/シングルサーバの場合はシステム単位で系を切り替えます。ただし，ユティリティ専用ユニット

（UNIX 版限定）は系切り替えできません。HiRDB/パラレルサーバの場合はユニット単位で系を切り替えます。なお，ユーザサーバホットスタンバイ，又は高速系切り替え機能を使用すると，系切り替えに掛かる時間が短縮できます。ユーザサーバホットスタンバイ，高速系切り替え機能については，マニュアル「HiRDB システム運用ガイド」の「系の切り替え時間の短縮（ユーザサーバホットスタンバイ，高速系切り替え機能）」を参照してください。

なお，業務処理中の系を**実行系**，待機中の系を**待機系**といい，系の切り替えが発生するたびに実行系と待機系が入れ替わります。また，システム構築時や環境設定時に二つの系を区別するため，最初に実行系として起動する系を**現用系**，待機系として起動する系を**予備系**といいます。系が切り替わると実行系と待機系は変わりますが，現用系と予備系は変わりません。系切り替え機能（スタンバイ型系切り替え機能）の概要を次の図に示します。

図 8–1　系切り替え機能（スタンバイ型系切り替え機能）の概要



注※1

系切り替えを実行する製品をこのマニュアルではクラスタソフトウェアといいます。HiRDB がサポートしているクラスタソフトウェアについては，「HiRDB がサポートしているクラスタソフトウェア」を参照してください。

注※2

共有ディスク装置については，「共有ディスク装置」を参照してください。

〔説明〕

業務処理中の実行系に障害が発生すると，待機系に障害の発生が通知されて系が切り替わり，待機系が実行系になって業務処理を続行します。

(2)　スタンバイレス型系切り替え機能とは

業務処理中の HiRDB に障害が発生した場合，ほかのユニットに系を切り替えて稼働中のバックエンドサーバに処理を代行させます。これを**スタンバイレス型系切り替え機能**といいます。スタンバイレス型系切り替え機能ではスタンバイ型系切り替え機能とは異なり，待機系ユニットを準備する必要がありません。

スタンバイレス型系切り替え機能には，次の二つの機能があります。

- 1：1 スタンバイレス型系切り替え機能
- 影響分散スタンバイレス型系切り替え機能

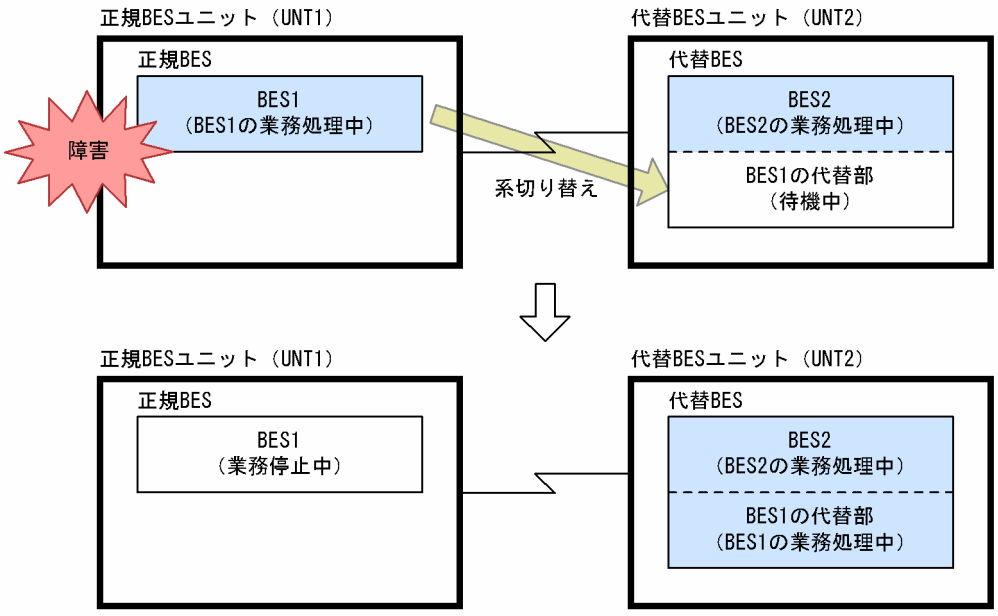
スタンバイレス型系切り替え機能は HiRDB/パラレルサーバのバックエンドサーバユニットに対して適用できます。ユニット内にバックエンドサーバ以外のサーバがある場合はそのユニットにスタンバイレス型系切り替え機能を適用できません。

(a)　1：1 スタンバイレス型系切り替え機能

1：1 スタンバイレス型系切り替え機能では，障害が発生したユニットを 1：1 に切り替えて別のバックエンドサーバに処理を代行させることができます。

なお，障害発生時に処理を代行してもらうバックエンドサーバを**正規 BES** といい，処理を代行するバックエンドサーバを**代替 BES** といいます。また，正規 BES のユニットを**正規 BES ユニット**といい，代替 BES のユニットを**代替 BES ユニット**といいます。1：1 スタンバイレス型系切り替え機能の概要を次の図に示します。

図 8–2　1：1 スタンバイレス型系切り替え機能の概要



〔説明〕

- 通常は BES1 及び BES2 の両方で処理を行います。

- 正規 BES ユニット（UNT1）に障害が発生した場合，系を切り替えて代替 BES で処理を代行します。処理を代行する部分を**代替部**といい，代替部で処理を行っているときを**代替中**といいます。
- 障害対策後に正規 BES ユニットを開始して，代替 BES で代行していた処理を正規 BES に切り替えて正常状態に戻します。これを**系の切り戻し**といいます。

備考

スタンバイ型系切り替え機能にある現用系などの概念と比較すると，1：1 スタンバイレス型系切り替え機能では次のようになります。

- 現用系が正規 BES ユニット，予備系が代替 BES ユニットと考えてください。
- 正常時は正規 BES ユニットが実行系で，代替部が待機系と考えてください。代替中は代替部が実行系で，正規 BES ユニットが待機系と考えてください。

前提条件

1：1 スタンバイレス型系切り替え機能を使用する場合は次に示す前提条件をすべて満たす必要があります。

- HiRDB Advanced High Availability を導入している
- Hitachi HA Toolkit Extension を導入している（クラスタソフトウェアが HA モニタの場合は必要ありません）
- 系切り替え機能をサーバモードで運用している

スタンバイ型系切り替え機能と比較して優れている点

1：1 スタンバイレス型系切り替え機能はスタンバイ型系切り替え機能に比べて次に示す点が優れています。

- 待機系ユニットを準備する必要がないため，システムリソースを効率的に使用できます。ただし，系が切り替わると処理を代行するバックエンドサーバではその分の負荷が大きくなるため，処理性能に影響を与えることがあります。
- サーバプロセスをあらかじめ起動しておくため，系の切り替え時間を高速系切り替え機能使用時と同じくらいに短縮できます。高速系切り替え機能については，「系の切り替え時間を短縮する機能（ユーザサーバホットスタンバイ，高速系切り替え機能）」を参照してください。

(b)　影響分散スタンバイレス型系切り替え機能

障害発生時に障害ユニット内のバックエンドサーバへの処理要求を，複数の稼働中ユニットに分散して実行させる機能を影響分散スタンバイレス型系切り替え機能といいます。影響分散スタンバイレス型系切り替え機能では，待機用サーバマシン，又は待機ユニットを準備する必要はなく，システムリソースを効率的に利用できます。障害発生後，障害ノードのサーバ処理を代行するユニットでは処理負荷が増えるため，トランザクション処理性能に影響を及ぼすことがあります。ただし，複数のユニットが障害ユニット内サーバへの処理要求を分担して実行することで，ユニット当たりの負荷上昇を抑え，システムの性能劣化を軽減します。

また，影響分散スタンバイレス型系切り替え機能では，バックエンドサーバを分散して切り替えられます。切り替え先を複数のユニットに分散させることもできます。さらに，障害発生によって切り替えた先のユ

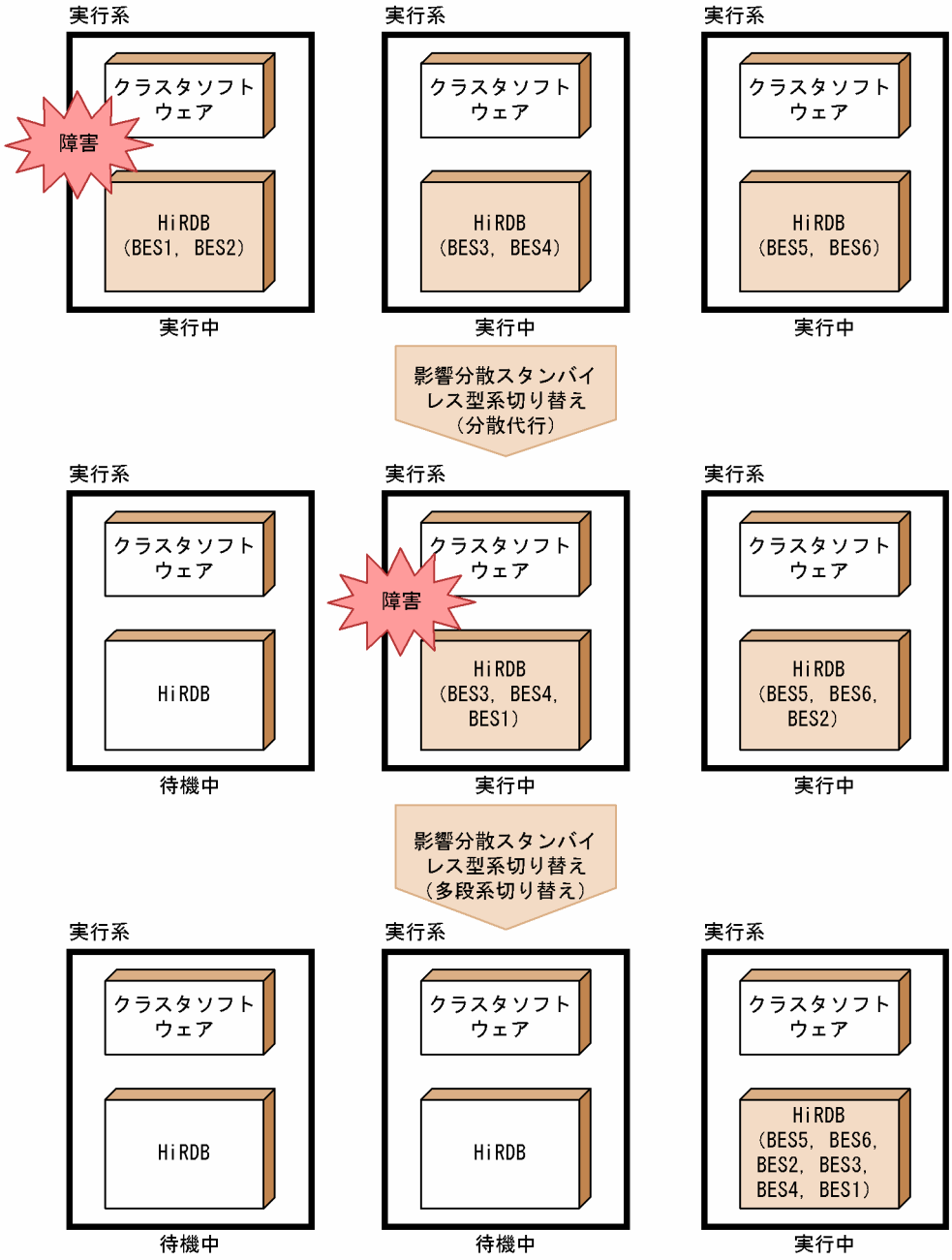
ニットで更に障害が発生しても，別の稼働中ユニットに更に切り替わることで処理を継続できます（以降，多段系切り替えといいます）。なお，1：1 スタンバイレス型系切り替えの場合は多段系切り替えができないため，切り替え先で障害が発生すると障害ユニットの代行処理は継続できなくなります。

影響分散スタンバイレス型系切り替え機能は，通常時からシステムリソースを有効に利用することを重視し，しかも，障害発生時の性能劣化を最小限に抑える必要があるシステムに対して適用してください。

なお，影響分散スタンバイレス型系切り替え機能では，障害発生時に処理を代行させるバックエンドサーバを**ホスト BES** といい，処理を代行するバックエンドサーバを**ゲスト BES** といいます。ホスト BES のユニットを**正規ユニット**といい，ゲスト BES のユニットを**受け入れユニット**といいます。受け入れユニットのすべては，**HA グループ**として定義しておく必要があります。また，ゲスト BES に対応付けられるバックエンドサーバ用のリソースを**ゲスト用領域**といいます。

影響分散スタンバイレス型系切り替え機能の概要（分散代行，多段系切り替え）を次の図に示します。

図 8–3　影響分散スタンバイレス型系切り替え機能の概要（分散代行，多段系切り替え）



前提条件

影響分散スタンバイレス型系切り替え機能を使用する場合は次に示す前提条件をすべて満たす必要があります。

- HiRDB Advanced High Availability を導入していることが必要です。
- 影響分散スタンバイレス型系切り替え機能は，バックエンドサーバだけから構成されるバックエンドサーバ専用ユニットだけを対象としています。
- 影響分散スタンバイレス型系切り替え機能を適用するユニットは，一つ以上の現用系のバックエンドサーバから構成される必要があります。受け入れ専用のユニットには適用できません。

(3)　HiRDB がサポートしているクラスタソフトウェア

HiRDB がサポートしているクラスタソフトウェアを次の表に示します。

表 8–2　HiRDB がサポートしているクラスタソフトウェア

HiRDB がサポートしているクラスタソフトウェア	OS の種類		
	AIX	Linux	Windows
HA モニタ	○	○	×
PowerHA	○	×	×
LifeKeeper	×	○	×
Microsoft Failover Cluster（MSFC）	×	×	○
CLUSTERPRO	×	○	○

（凡例）

○：サポートしています。

×：サポートしていません。

注

- HA モニタについては，マニュアル「高信頼化システム監視機能 HA モニタ」を参照してください。そのほかのクラスタソフトウェアについては，各製品のマニュアルを参照してください。
- クラスタソフトウェアの種類によってサポートする機能が異なります。各クラスタソフトウェアがサポートする機能については，表「モニタモードとサーバモードが監視対象とする障害」及び表「クラスタソフトウェアと，モニタモード及びサーバモードでの運用可否」を参照してください。

(4)　共有ディスク装置

現用系と予備系とで共有する外付けハードディスク（UNIX 版の場合はキャラクタ型スペシャルファイル）が必要になります。このハードディスクを**共有ディスク装置**といいます。共有ディスク装置は系切り替えが発生したときに，実行系から待機系に情報を引き継ぐために使用されます。共有ディスク装置には次に示す HiRDB ファイルシステム領域を作成します。

- RD エリア用の HiRDB ファイルシステム領域
- システムファイル用の HiRDB ファイルシステム領域
- バックアップファイル用の HiRDB ファイルシステム領域
- アンロードログファイル用の HiRDB ファイルシステム領域
- 監査証跡ファイル用の HiRDB ファイルシステム領域（セキュリティ監査機能を使用する場合）

8.1.2　系切り替え機能の運用方法

系切り替え機能の運用方法には**モニタモード**と**サーバモード**があります。モニタモードの場合は系障害だけを監視対象とし，サーバモードの場合は系障害及びサーバ障害を監視対象とします。また，サーバモードではモニタモードに比べて系の切り替え時間を短縮できます。モニタモードとサーバモードが監視対象とする障害を次の表に示します。

表 8–3　モニタモードとサーバモードが監視対象とする障害

系切り替え機能の運用方法	監視対象とする障害	
	系障害※1	サーバ障害※2
モニタモード	○	×
サーバモード	○	○

（凡例）

○：監視します。

×：監視しません。

注※1

ここでは次に示す障害を系障害として想定していますが，系障害の条件はクラスタソフトウェアによって異なります。クラスタソフトウェアのマニュアルなどで確認してください。

- ハードウェアの障害
- OS の障害
- 電源断
- クラスタソフトウェアの障害
- 系のスローダウン

注※2

ここでは次に示す障害をサーバ障害として想定していますが，サーバ障害の条件はクラスタソフトウェアによって異なります。クラスタソフトウェアのマニュアルなどで確認してください。

- HiRDB（HiRDB/パラレルサーバの場合はユニット）の異常終了
- HiRDB（HiRDB/パラレルサーバの場合はユニット）のスローダウン
- データベースのパス障害

(1)　モニタモード，及びサーバモードで運用できるクラスタソフトウェア

系切り替えを実行する製品をこのマニュアルでは**クラスタソフトウェア**といいます。HiRDB がサポートしているクラスタソフトウェアと，モニタモード及びサーバモードでの運用可否を次の表に示します。

表 8–4　クラスタソフトウェアと，モニタモード及びサーバモードでの運用可否

クラスタソフトウェア	モニタモード	サーバモード
HA モニタ	○	○
PowerHA	○	×
LifeKeeper	○	×
Microsoft Failover Cluster（MSFC）	○	○
CLUSTERPRO	○	○※

（凡例）

○：このモードで運用できます。

×：このモードで運用できません。

注※　Windows 版の場合のみサーバモードで運用できます。

(2)　サーバモードで運用する場合に必要な製品

系切り替え機能をサーバモードで運用する場合は，次の表に示す製品が必要になります。

表 8–5　サーバモードで運用する場合に必要な製品

機能	HiRDB Advanced High Availability	Hitachi HA Toolkit Extension
通常の系切り替え	–	○※
ユーザサーバホットスタンバイ	–	○※
高速系切り替え機能	–	○※
1：1 スタンバイレス型系切り替え機能	○	○※
影響分散スタンバイレス型系切り替え機能	○	○※

（凡例）

○：この機能を使用する場合に必要な製品です。

–：必要ありません。

注※　クラスタソフトウェアが HA モニタ（UNIX 版限定）の場合，Hitachi HA Toolkit Extension は必要ありません。

8.1.3　系切り替え機能の形態

系切り替え機能には次に示す 3 種類の形態があります。

- **自動系切り替え**
  実行系に障害が発生した場合，自動的に系を切り替えます。
- **計画系切り替え**
  【UNIX 版の場合】
  実行系でクラスタソフトウェアのコマンドを実行して意図的に系を切り替えます。
  【Windows 版の場合】
  クラスタアドミニストレータで HiRDB のグループを移動して意図的に系を切り替えます。
- **連動系切り替え**
  OLTP などほかの製品と HiRDB が連携している場合の系切り替えです。実行系に障害が発生した場合，OLTP 及び HiRDB（ユニット）をまとめて系切り替えします。連動系切り替えは，自動系切り替え又は計画系切り替えのどちらでもできます。連動系切り替えができるかどうかについては，クラスタソフトウェアのマニュアルで確認してください。

8.1.4　システム構成例

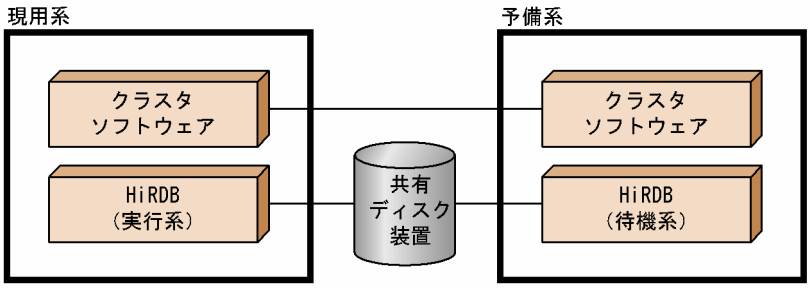
系切り替え機能使用時のシステム構成例について説明します。

(1)　スタンバイ型系切り替え機能のシステム構成例

(a)　1：1 系切り替え構成

実行系と待機系が 1：1 に対応している構成です。系が切り替わってもレスポンスタイムを保証したい場合にこの構成を適用します。ただし，待機系のサーバマシンのリソースは使用できません（サーバマシン 2 台のうち 1 台のリソースが使用できません）。1：1 系切り替え構成を次の図に示します。

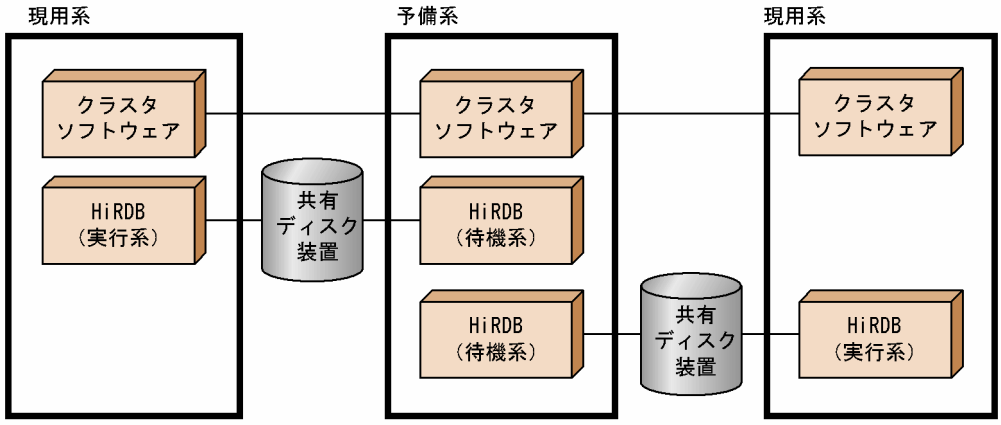
図 8–4　1：1 系切り替え構成



(b)　2：1 系切り替え構成

実行系と待機系が 2：1 に対応している構成です。予備系をマルチ HiRDB の構成にします。系が切り替わってもレスポンスタイムを保証したい業務にこの構成を適用します（二つの実行系が同時に切り替わった場合は，レスポンスタイムが低下します）。ただし，待機系のサーバマシンのリソースが使用できません（サーバマシン 3 台のうち 1 台のリソースが使用できません）。2：1 系切り替え構成を次の図に示します。

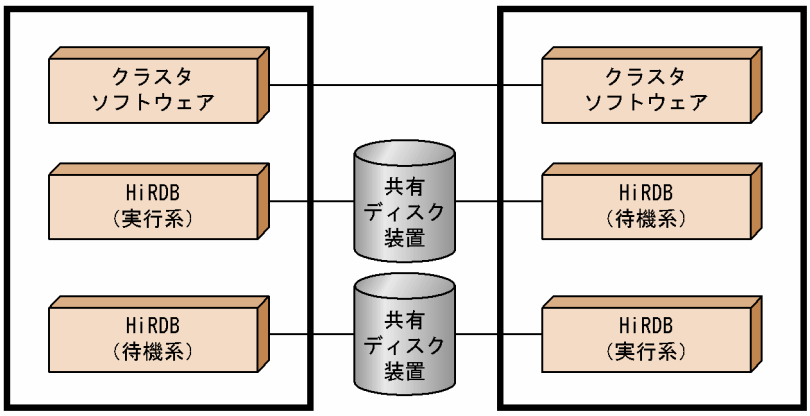
図 8–5　2：1 系切り替え構成



(c)　相互系切り替え構成

実行系として動作しながら，同じサーバマシンに互いの待機系を持つ構成です。それぞれのサーバマシンで，実行系及び待機系の二つの HiRDB をマルチ HiRDB の構成にします。サーバマシンのリソースを有効に利用したい場合にこの構成を適用します。ただし，系が切り替わるとレスポンスタイムが低下します。相互系切り替え構成を次の図に示します。

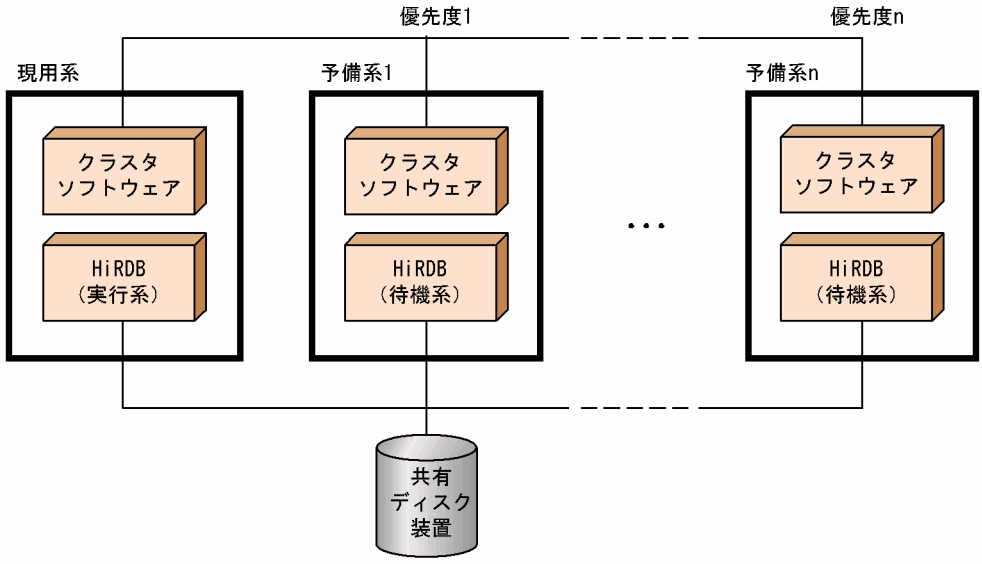
図 8–6　相互系切り替え構成



(d)　マルチスタンバイ構成

一つの実行系に複数の待機系を持つシステム構成です。実行系の障害が回復するまでの間に発生する，待機系の障害（多点障害）に備えたい場合にこの構成を適用します。待機系には優先度を付け，実行系に障害が発生した時は，一番優先度の高い待機系に切り替えます。マルチスタンバイ構成を次の図に示します。

図 8–7　マルチスタンバイ構成



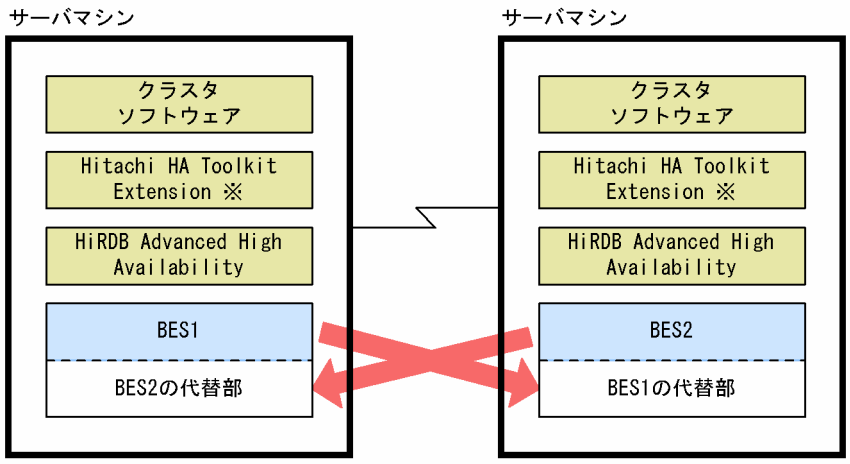
(2)　1：1 スタンバイレス型系切り替え機能のシステム構成例

1：1 スタンバイレス型系切り替えの代表的なシステム構成例を説明します。

(a)　相互代替構成

1：1 スタンバイレス型系切り替えで二つのバックエンドサーバがお互いに代替 BES になる構成です。相互代替構成のシステム構成例を次の図に示します。

図 8–8　相互代替構成のシステム構成例



注※　クラスタソフトウェアが HA モニタ（UNIX 版限定）の場合，Hitachi HA Toolkit Extension は必要ありません。

〔説明〕

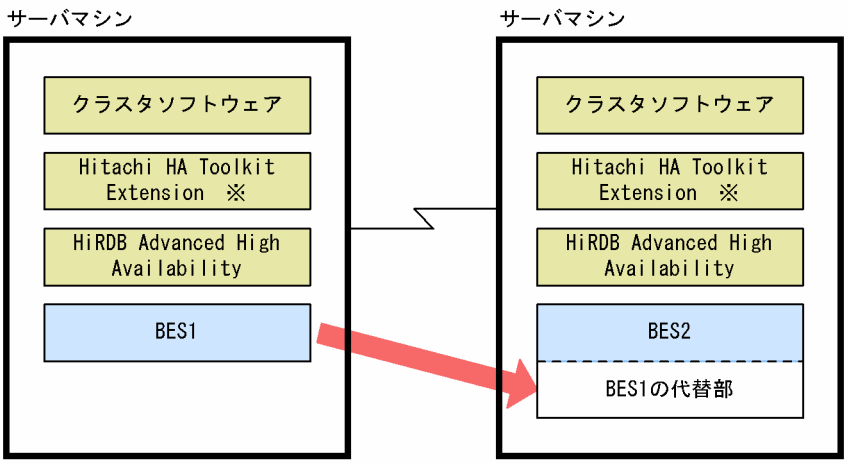
- BES1 は BES2 の代替 BES です。BES2 に障害が発生した場合は，BES2 の代替部が BES2 の処理を代行します。

- BES2 は BES1 の代替 BES です。BES1 に障害が発生した場合は，BES1 の代替部が BES1 の処理を代行します。

(b)　片方向代替構成

1：1 スタンバイレス型系切り替えで片方向だけの代行関係を持つ構成です。片方向代替構成（2 ノード構成）のシステム構成例を次の図に示します。

図 8–9　片方向代替構成（2 ノード構成）のシステム構成例



注※　クラスタソフトウェアが HA モニタ（UNIX 版限定）の場合，Hitachi HA Toolkit Extension は必要ありません。

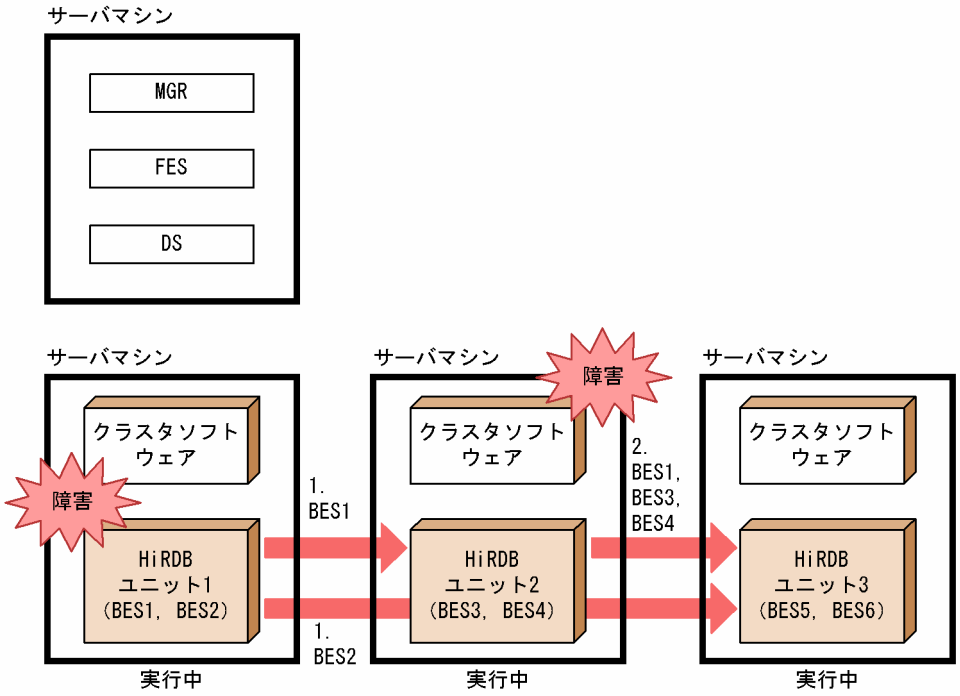
〔説明〕

BES2 は BES1 の代替 BES です。BES1 に障害が発生した場合は，BES1 の代替部が BES1 の処理を代行します。BES2 に障害が発生した場合，BES1 は処理を代行しません。

(3)　影響分散スタンバイレス型系切り替え機能のシステム構成例

影響分散スタンバイレス型系切り替え機能のシステム構成例を次の図に示します。正規ユニットでの障害発生時に，障害対象の現用系のバックエンドサーバへの処理を稼働中の複数のサーバマシンがバックエンドサーバ単位で分担して実行します。

図 8–10　影響分散スタンバイレス型系切り替え機能のシステム構成例



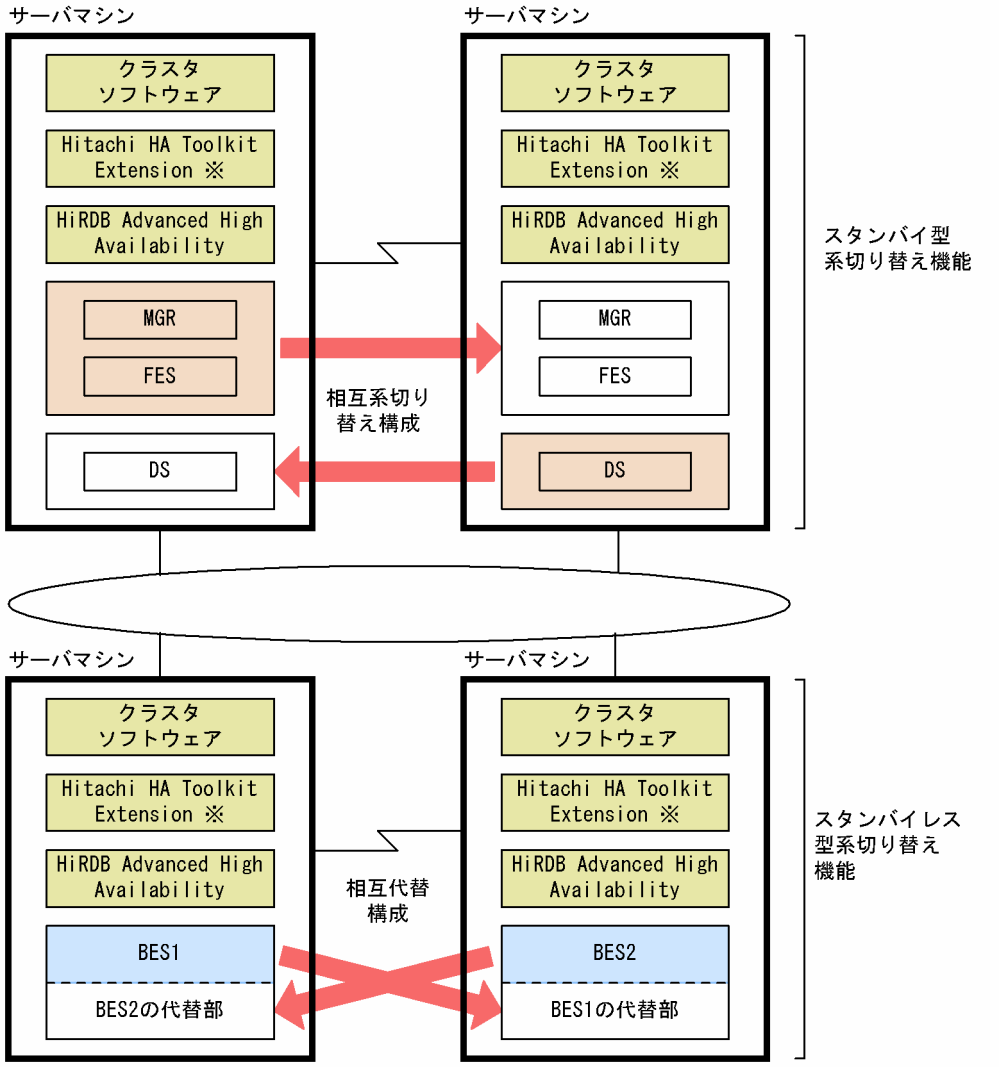
〔説明〕

1. ユニット 1 に障害が発生した場合は，ユニット 2 で BES1 がゲスト BES として処理を実行し，ユニット 3 で BES2 がゲスト BES として処理を実行します。
2. ユニット 1 の障害中に更にユニット 2 で障害が発生した場合は，ユニット 3 で BES1，BES2，BES3，BES4 がそれぞれゲスト BES として処理を実行します。

(4)　1：1 スタンバイレス型とスタンバイ型を混合したシステム構成例

1：1 スタンバイレス型とスタンバイ型を混合したシステム構成例を次の図に示します。

図 8–11　1：1 スタンバイレス型とスタンバイ型を混合したシステム構成例



注※　クラスタソフトウェアが HA モニタ（UNIX 版限定）の場合，Hitachi HA Toolkit Extension は必要ありません。

〔説明〕

- MGR（システムマネジャ），FES（フロントエンドサーバ），DS（ディクショナリサーバ）のユニットはスタンバイ型系切り替え機能（相互系切り替え構成）を使用します。
- BES（バックエンドサーバ）のユニットは 1：1 スタンバイレス型系切り替え機能（相互代替構成）を使用します。
- 全サーバマシンに HiRDB Advanced High Availability が必要になります。スタンバイレス型系切り替え機能を適用しないサーバマシン，及び系切り替え機能を適用しないサーバマシンにも HiRDB Advanced High Availability が必要です。

8.1.5　系の切り替え時間を短縮する機能（ユーザサーバホットスタンバイ，高速系切り替え機能）

系の切り替え時間を短縮する機能として次に示す機能があります。

- ユーザサーバホットスタンバイ
- 高速系切り替え機能

なお，これらの機能はサーバモードでの運用が前提条件になります。モニタモードで運用する場合はこれらの機能を使用できません。

(1)　ユーザサーバホットスタンバイ

系切り替えが発生したとき，待機系 HiRDB を開始するのに次に示す処理が実施されます。

- システムサーバの起動処理
- システムファイルの引き継ぎ処理
- サーバプロセスの起動処理
- ロールフォワード処理

この中のサーバプロセスの起動処理に掛かる時間は，系の切り替え時間の中で大きな割合を占めています。サーバプロセスの起動処理に掛かる時間はサーバプロセスの常駐数に比例するため，常駐数が多いと系の切り替え時間が長くなります。そこで，待機系 HiRDB のサーバプロセスをあらかじめ起動しておいて，系の切り替え時にサーバプロセスの起動処理をしないようにします。系の切り替え時にサーバプロセスの起動処理がない分，系の切り替え時間を短縮できます。これを**ユーザサーバホットスタンバイ**といいます。例えば，100MIPS ぐらいのサーバマシンでサーバプロセスを一つ起動するのに約１秒掛かります。その分，系の切り替え時間を短縮できます。

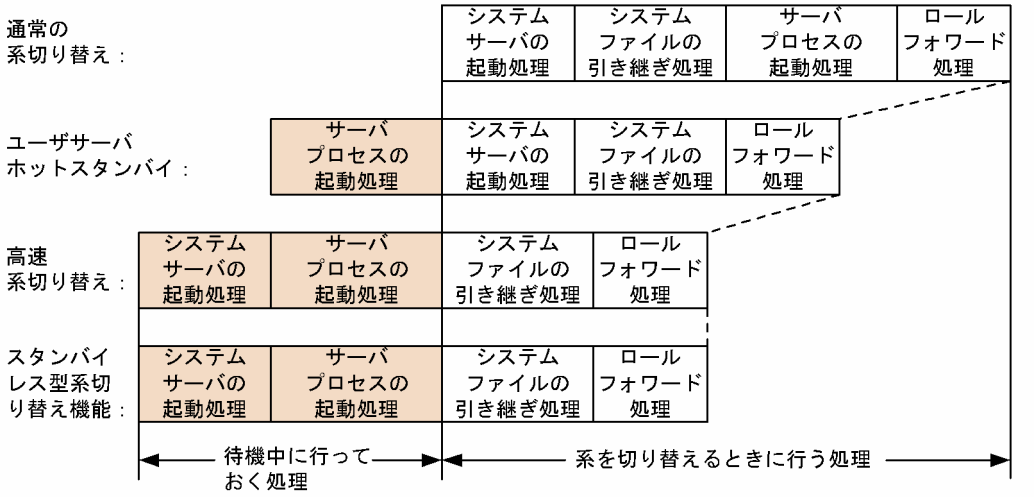
ユーザサーバホットスタンバイについては，マニュアル「HiRDB システム運用ガイド」を参照してください。

(2)　高速系切り替え機能

待機系 HiRDB のサーバプロセスをあらかじめ起動しておいて，系の切り替え時にサーバプロセスの起動処理をしません。これを**高速系切り替え機能**といいます。系の切り替え時にサーバプロセスの起動処理がない分，系の切り替え時間を短縮できます。高速系切り替え機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

なお，高速系切り替え機能の方がユーザサーバホットスタンバイより系の切り替え時間を短縮できます（高速系切り替え機能はユーザサーバホットスタンバイの機能を包括しています）。系の切り替え時間の比較を次の図に示します。

図 8–12　系の切り替え時間の比較



〔説明〕

あらかじめ網掛け部分の処理を実行して待機するため，系の切り替え時に網掛け部分の処理が不要になります。その分だけ系の切り替え時間が短縮されます。

8.1.6　系切り替え失敗時の自動再起動機能【UNIX 版限定】

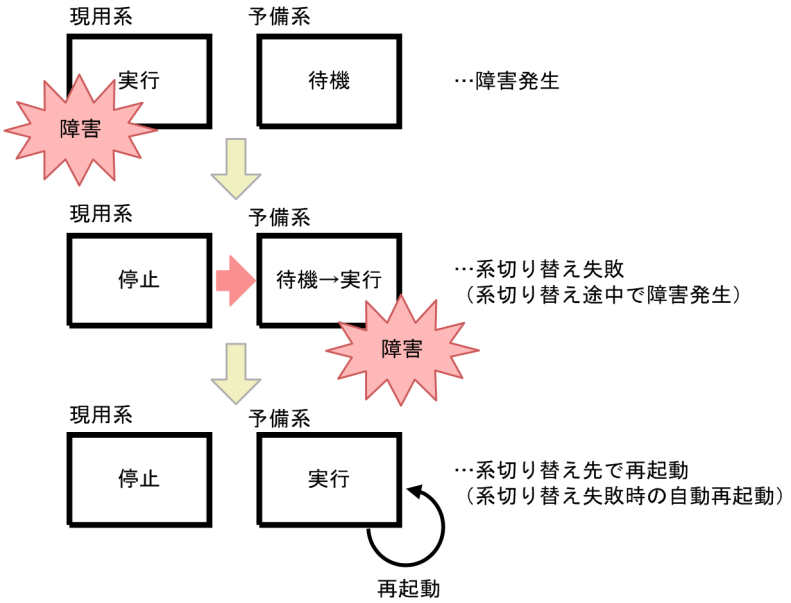
系切り替え先の HiRDB（HiRDB/パラレルサーバの場合はユニット）が系の切り替え中に異常終了した場合に，系切り替え先の HiRDB を再起動する機能です。これによって，系の切り替え中に障害が発生しても，自動で HiRDB を復旧できます。

この機能を使用するには，HA モニタの server 定義文の switch_error オペランドに retry を指定した構成で，システム定義の pd_ha_switch_error オペランドに retry を指定します。

詳細は，マニュアル「HiRDB システム定義」の pd_ha_switch_error オペランドを参照してください。

系切り替え失敗時の自動再起動の概要を次の図に示します。

図 8–13　系切り替え失敗時の自動再起動の概要



注意事項

- 系切り替え失敗時の自動再起動機能は，AIX 版及び Linux 版限定の機能です。
- この機能を使用するには，次に示す条件をすべて満たすようにシステムを構築してください。
  - ・クラスタソフトウェアに HA モニタを使用
  - ・HA モニタのバージョンが，この機能に対応したバージョン以降
    AIX の場合：01-25 以降
    Linux の場合：01-55-01 以降
  - ・ユーザサーバホットスタンバイ又は高速系切り替え機能を使用

8.1.7　系切り替えの実行時間監視機能【UNIX 版限定】

系切り替え処理を監視し，監視時間を過ぎても系切り替え処理が完了しなかった場合に，系切り替え先の HiRDB（HiRDB/パラレルサーバの場合はユニット）を異常終了させる機能です。

マルチスタンバイ構成又は系切り替え失敗時の自動再起動機能と組み合わせることで，系切り替え処理が停滞する障害が発生しても，自動で HiRDB を復旧できます。

この機能を使用するには，システム定義の pd_ha_observe_timeout オペランドに監視時間を指定します。

詳細は，マニュアル「HiRDB システム定義」の pd_ha_observe_timeout オペランドを参照してください。

系切り替えの実行時間監視の概要を次の図に示します。

- 図「系切り替えの実行時間監視を系切り替え失敗時の自動再起動機能と組み合わせて使用した例」

- 図「系切り替えの実行時間監視の概要」

図 8–14　系切り替えの実行時間監視を系切り替え失敗時の自動再起動機能と組み合わせて使用した例

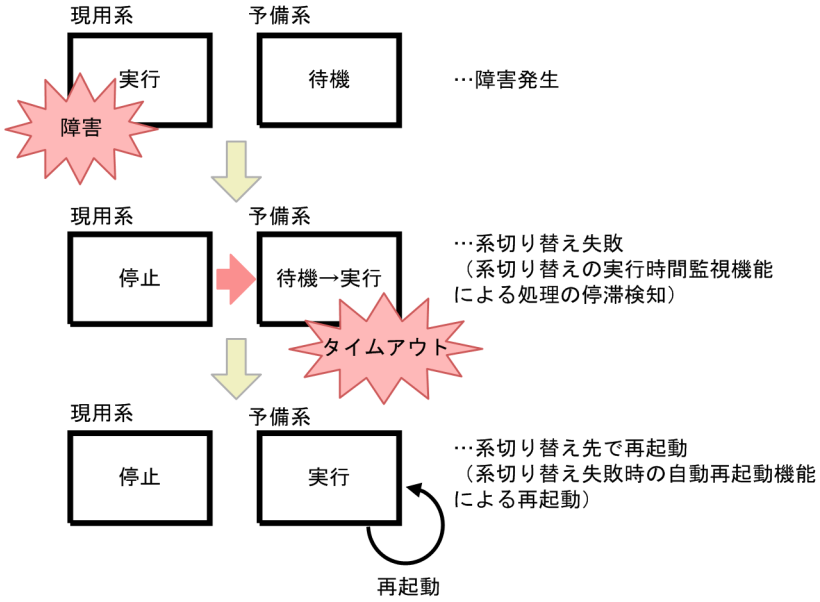
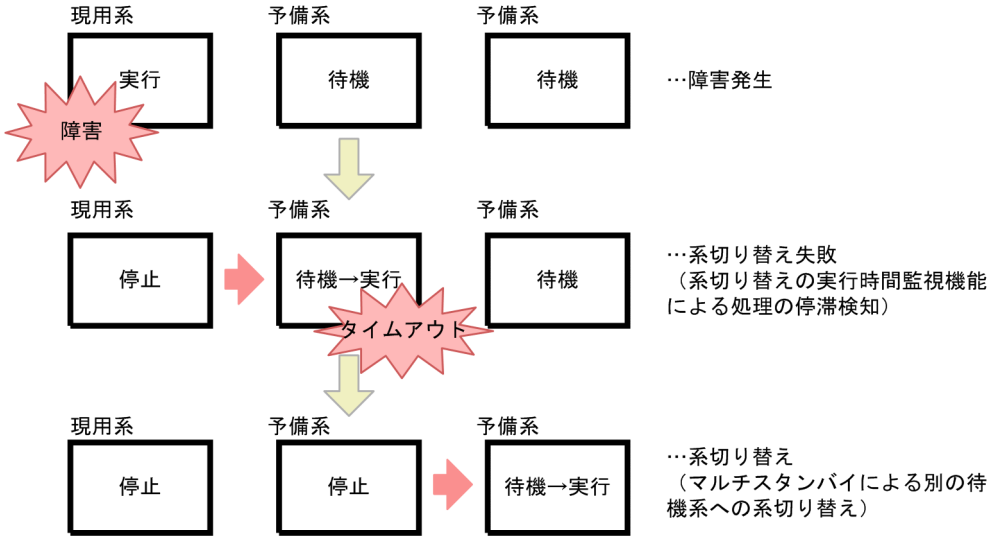


図 8–15　系切り替えの実行時間監視の概要



注意事項

- 系切り替えの実行時間監視機能は，ユーザサーバホットスタンバイ又は高速系切り替え機能と組み合わせた場合に使用できます。

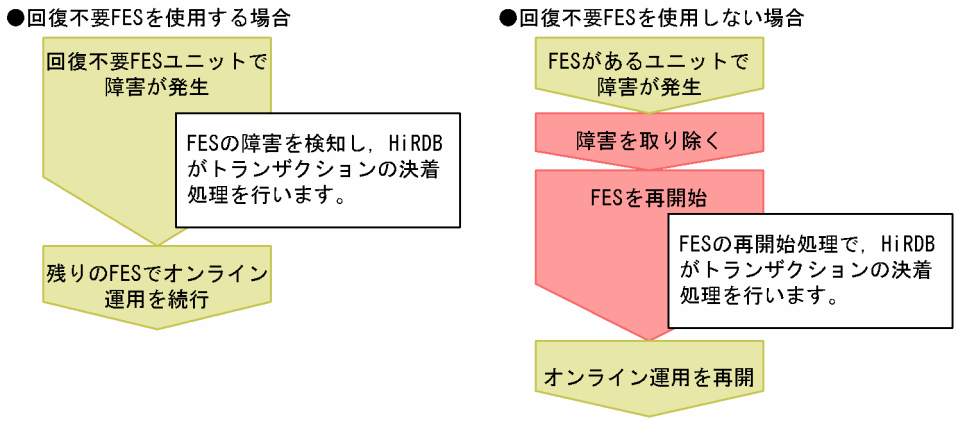
8.2　回復不要 FES

HiRDB Non Recover FES を導入すると，**回復不要 FES** を使用できます。ここでは，回復不要 FES について説明します。回復不要 FES の詳細は，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

8.2.1　回復不要 FES とは

フロントエンドサーバがあるユニットで障害が発生して異常終了すると，そのフロントエンドサーバから実行していたトランザクションは未決着状態になることがあります。未決着状態のトランザクションは，データベースの排他を確保しているため，一部のデータベースに対する参照又は更新が制限されます。通常，未決着状態のトランザクションの決着処理をするためには，フロントエンドサーバの障害を取り除いて再開始する必要がありますが，異常終了したフロントエンドサーバが**回復不要 FES** であれば，HiRDB が自動的に未決着状態になっていたトランザクションを決着します。これによって，ほかのフロントエンドサーバやバックエンドサーバを使用して，データベースの更新を再開できます。回復不要 FES があるユニットを**回復不要 FES ユニット**といいます。回復不要 FES を使用する場合としない場合の運用を次の図に示します。

図 8–16　回復不要 FES を使用する場合としない場合の運用



なお，回復不要 FES を使用するためには，**HiRDB Non Recover FES** が必要です。

適用基準

障害が発生したフロントエンドサーバを再開始しないで，残りのフロントエンドサーバでオンライン運用を続行できます。そのため，24 時間連続稼働が必要なシステムの場合に適用をお勧めします。

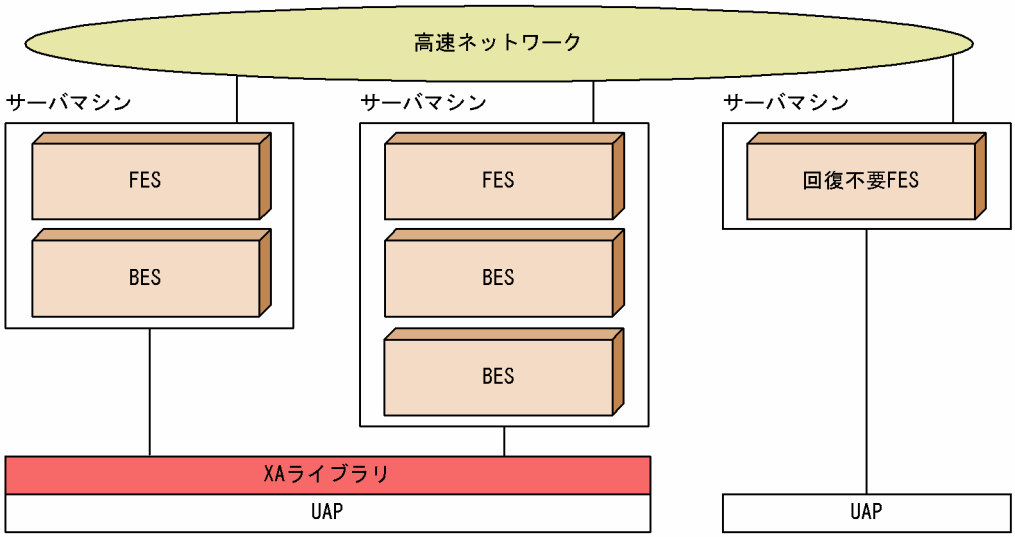
設定方法

回復不要 FES を使用するには，**pdstart** オペランドの**-k** オプションに **stls** を指定します。

8.2.2　回復不要 FES を使用したシステムの構成例

回復不要 FES を使用したシステムの構成例を次の図に示します。

図 8–17　回復不要 FES を使用したシステムの構成例



〔説明〕

- 回復不要 FES は，単独で構成されるユニットに配置してください。
- 回復不要 FES では，X/Open XA インタフェースを使用して接続する UAP は実行できません。クライアント環境定義の PDFESHOST 及び PDSERVICEGRP を指定して，回復不要 FES 以外のフロントエンドサーバに接続してください。
- 回復不要 FES，及び回復不要 FES ユニットが停止していても，**pdrplstart** コマンド，及び **pdrplstop** コマンドは実行できます。

8.3　リアルタイム SAN レプリケーション（ディザスタリカバリ）【UNIX 版限定】

ここでは，地震又は火災などの大規模災害からの早期復旧を想定した**ディザスタリカバリ**システム，**リアルタイム SAN レプリケーション**の概要について説明します。リアルタイム SAN レプリケーションの運用方法については，マニュアル「HiRDB ディザスタリカバリシステム　構築・運用ガイド」を参照してください。

注意事項

リアルタイム SAN レプリケーションは，AIX 版，及び Linux 版限定の機能です。

参考

この章の説明は，RAID Manager，TrueCopy，及び Universal Replicator についての知識があることを前提にしています。

8.3.1　リアルタイム SAN レプリケーションとは

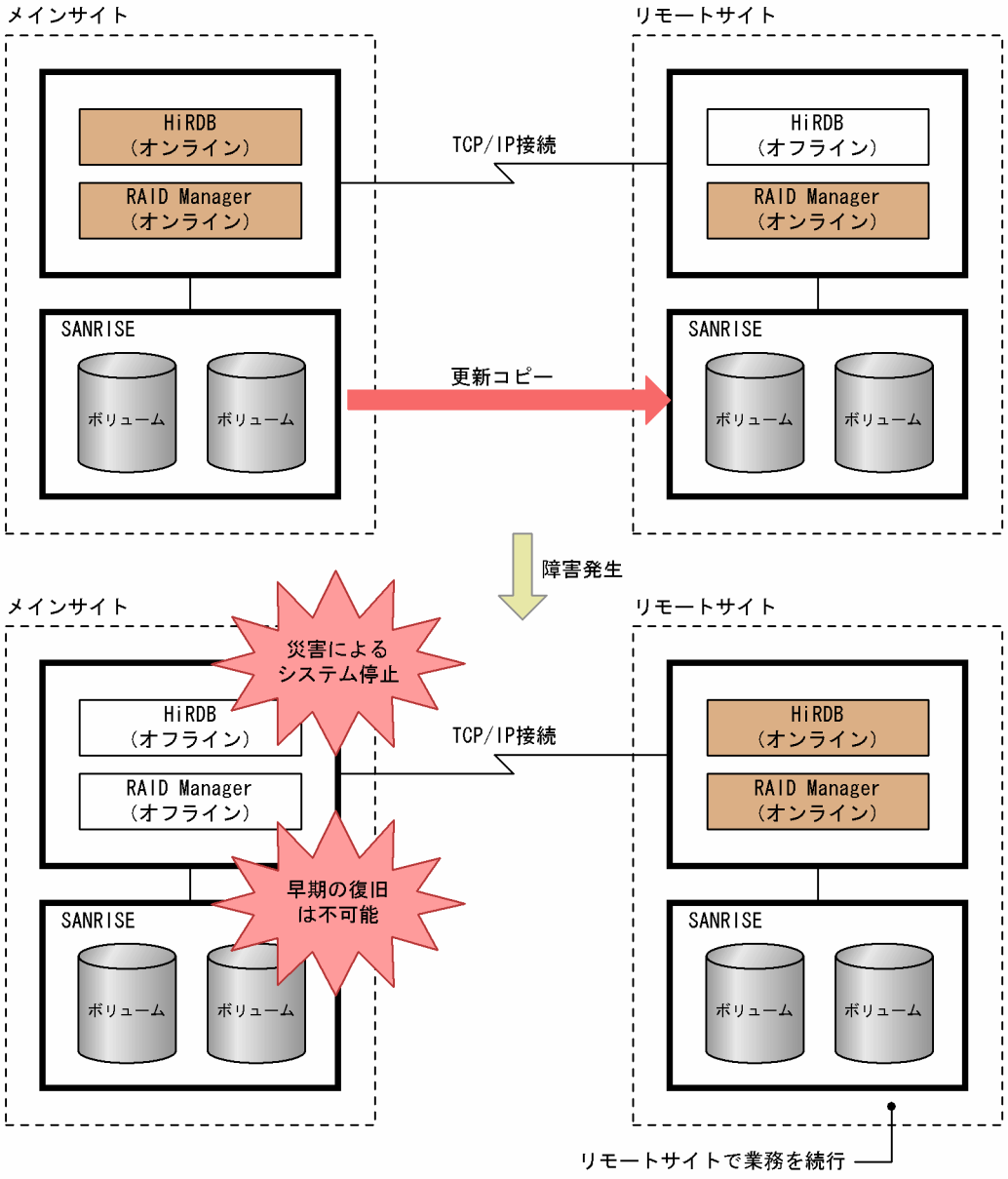
(1)　機能概要

通常使用しているシステムが地震，火災などの災害によって物理的に回復困難な状況になった場合，遠隔地に準備している予備のシステムを稼働して業務を続行できます。このシステム環境を**リアルタイム SAN レプリケーション**（RiSe）といいます。なお，通常使用しているシステムがあるサイトを**メインサイト**といい，遠隔地に準備している予備のシステムがあるサイトを**リモートサイト**といいます。

メインサイト及びリモートサイトのデータは，**日立ディスクアレイシステム**上に配置し，メインサイトのデータに更新が発生した場合，日立ディスクアレイシステムの **TrueCopy** 又は **Universal Replicator** を使用してリモートサイトに反映（**更新コピー**）します。

リアルタイム SAN レプリケーションの概要を次の図に示します。

図 8–18　リアルタイム SAN レプリケーションの概要



〔説明〕

- 通常はメインサイトの HiRDB を使用して業務を行います。メインサイトのファイルに更新が発生した場合，その更新内容をリモートサイトに**更新コピー**します。更新コピーによってメインサイトとリモートサイトで同じデータを持ちます。
- メインサイトで地震又は火災などの大規模災害が発生し，メインサイトのシステムが早期に復旧できない場合，リモートサイトで HiRDB を再開始して業務を続行します。

参考

- 更新コピーは TrueCopy 又は Universal Replicator が行います。TrueCopy 又は Universal Replicator は，ホストを経由しないで，日立ディスクアレイシステム間で直接データをコピーします。

- RAID Manager とは，日立ディスクアレイシステムの付加プログラムプロダクトで，TrueCopy 又は Universal Replicator を制御，運用するコマンドなどを提供しています。

(2)　更新コピーの対象になるファイル

更新コピーの対象になるファイルを次に示します。これらのファイルに更新が発生すると，リモートサイトの同じファイルに更新情報がコピーされます。

- データベースファイル（RD エリアを構成する HiRDB ファイル）
- システムログファイル
- シンクポイントダンプファイル
- ステータスファイル

(3)　同期コピーと非同期コピー

更新コピー処理には**同期コピー**と**非同期コピー**があります。同期コピーと非同期コピーの特徴を次の表に示します。

表 8–6　同期コピーと非同期コピーの特徴

項目	同期コピー	非同期コピー
処理方式	リモートサイトの更新処理が完了した後にメインサイトの更新処理を完了します（メインサイトの更新処理がリモートサイトの更新処理を待ち合わせます）。	リモートサイトの更新処理の完了を待たないでメインサイトの更新処理を完了します。
メインサイトとリモートサイトのデータの整合性	メインサイトとリモートサイトで常にデータが一致します。	データ欠損が発生することがあります。そのため，メインサイトとリモートサイトでデータが一致しないことがあります。
性能への影響	トランザクションの処理性能が低下します。低下の割合はサイト間の距離に比例します。※1	性能への影響はありません。※1※2

注※1　TrueCopy が保証する理論値に従った特徴です。

注※2　Universal Replicator が保証する理論値に従った特徴です。

8.3.2　前提プラットフォーム及び前提製品

(1)　前提プラットフォーム

前提プラットフォームを次に示します。メインサイトとリモートサイトで同じプラットフォームを使用してください。

- AIX
- Linux

(2) 前提製品

メインサイトとリモートサイトの両方に，日立ディスクアレイシステムの製品が必要となります。前提製品については，マニュアル「HiRDB ディザスタリカバリシステム　構築・運用ガイド」を参照してください。

8.3.3 リモートサイトへのデータ反映方式

ここでは，メインサイトからリモートサイトにデータを反映するときの処理方式について説明します。リアルタイム SAN レプリケーションではデータを反映するときの処理方式が四つあります。データ反映方式によってシステムの構築方法や，運用方法が異なるため，使用しているシステムに合わせて，HiRDB 管理者は次に示す四つの方式から一つを選択してください。

- 全同期方式
- 全非同期方式
- ハイブリッド方式
- ログ同期方式

8.3.4 全同期方式

全同期方式を適用すると，リモートサイトへの更新コピー処理が**同期コピー**で行われます。同期コピーの場合，リモートサイトの更新処理が完了した後にメインサイトの更新処理を完了します（メインサイトの更新処理がリモートサイトの更新処理を待ち合わせます）。したがって，全同期方式を適用した場合，メインサイトの更新内容が必ずリモートサイトに反映されます。そのため，メインサイトの HiRDB が災害などによって異常終了した場合，異常終了直前の状態からリモートサイトで HiRDB を再開始してサービスを続行できます。

ただし，メインサイトでファイルを更新するときに，リモートサイトのファイルに更新が反映されるまでの間，待ち合わせを行うため，メインサイトのトランザクション性能に影響を与えることがあります。

全同期方式の概要を次の図に示します。また，リモートサイトに更新コピーするときの処理方式（全同期方式の場合）を次の表に示します。

図 8–19　全同期方式の概要

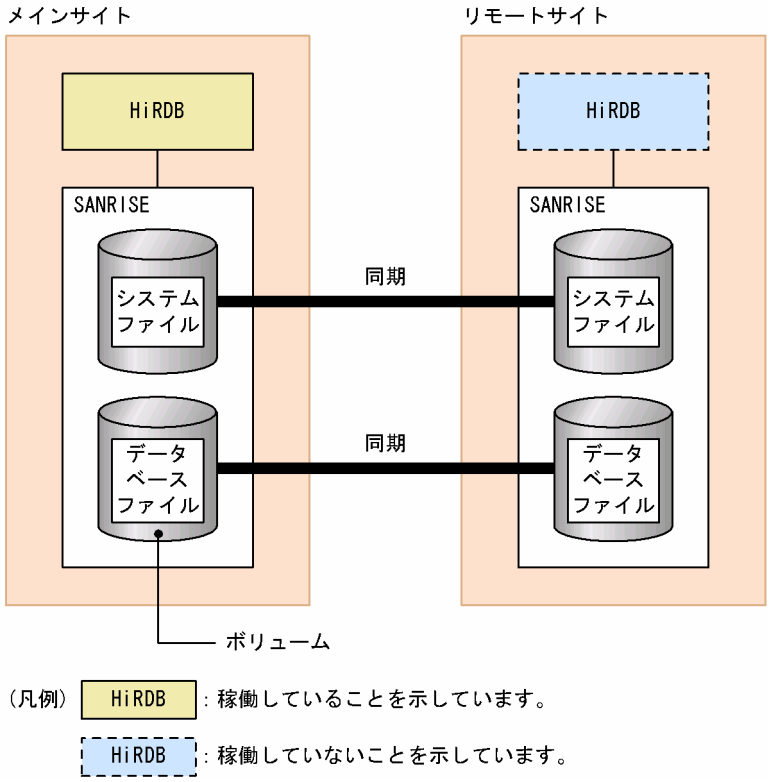


表 8–7　リモートサイトに更新コピーするときの処理方式（全同期方式の場合）

リモートサイトにコピーされるファイル		更新コピーするときの処理方式
データベースファイル		同期コピー
システムファイル	システムログファイル	
	シンクポイントダンプファイル	
	ステータスファイル	

8.3.5　全非同期方式

全非同期方式を適用すると，リモートサイトへの更新コピー処理が**非同期コピー**で行われます。非同期コピーの場合，リモートサイトの更新処理の完了を待たないでメインサイトの更新処理を完了するため，メインサイトのトランザクション性能に影響を与えません。

ただし，メインサイトのファイル（更新コピーの対象ファイル）の更新内容がリモートサイトに反映されないことがあります。そのため，メインサイトの HiRDB が災害などによって異常終了し，リモートサイトで HiRDB を再開始した場合，再開始したときの状態が異常終了直前の状態と異なることがあります。したがって，全非同期方式の場合，メインサイトから継続したサービスの開始を保証できません。

全非同期方式の概要を次の図に示します。また，リモートサイトに更新コピーするときの処理方式（全非同期方式の場合）を次の表に示します。

図 8–20　全非同期方式の概要

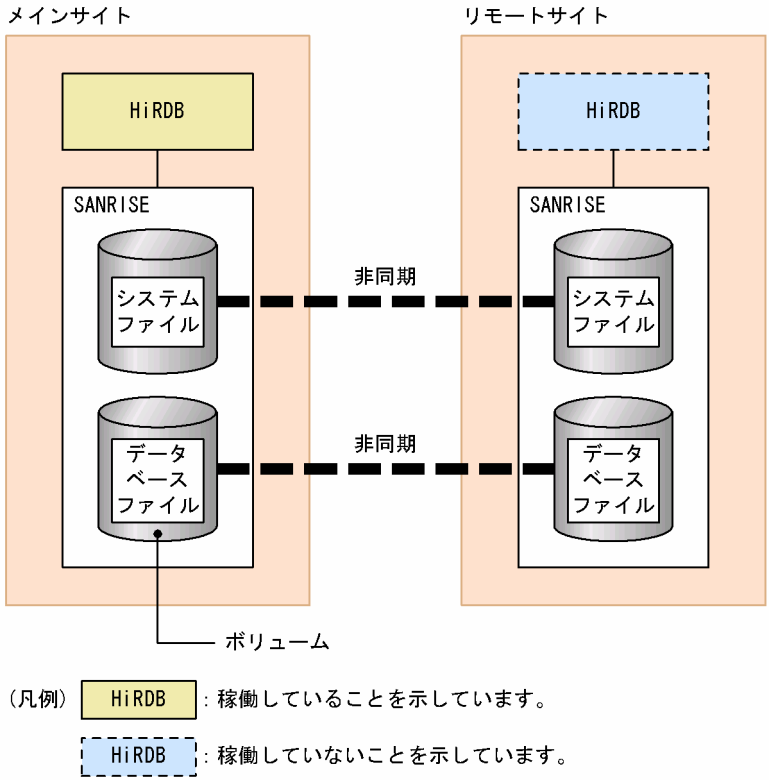


表 8–8　リモートサイトに更新コピーするときの処理方式（全非同期方式の場合）

リモートサイトにコピーされるファイル		更新コピーするときの処理方式
データベースファイル		非同期コピー
システムファイル	システムログファイル	
	シンクポイントダンプファイル	
	ステータスファイル	

8.3.6　ハイブリッド方式

ハイブリッド方式を適用すると，リモートサイトへの更新コピー処理が次に示すようになります。

- データベースファイルの更新コピー処理は**非同期コピー**で行います。
- システムファイルの更新コピー処理は**同期コピー**で行います。

システムログファイルなどのデータベースの回復に必要な情報はリモートサイトへの反映を保証するように同期コピーを行います。したがって，メインサイトの HiRDB が災害によって異常終了した場合，異常終了直前の状態でリモートサイトの HiRDB を再開始できます。ハイブリッド方式は，主に大規模システム向けの処理方式といえます。

また，回復可能なデータベースファイルについては非同期コピーとし，トランザクション性能への影響を全同期方式に比べて少なくします。

> **参考**
>
> ハイブリッド方式は，全同期方式と全非同期方式の両方の利点を持ちますが，ほかの方式に比べて運用が難しくなります。

ハイブリッド方式の概要を次の図に示します。また，リモートサイトに更新コピーするときの処理方式（ハイブリッド方式の場合）を次の表に示します。

図 8–21　ハイブリッド方式の概要

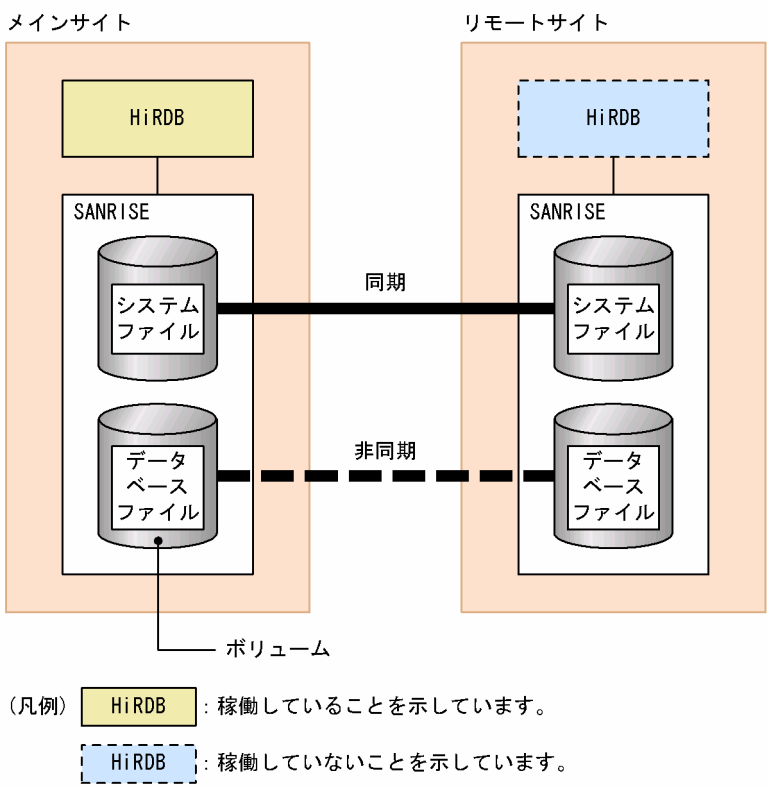


表 8–9　リモートサイトに更新コピーするときの処理方式（ハイブリッド方式の場合）

リモートサイトにコピーされるファイル		更新コピーするときの処理方式
データベースファイル		非同期コピー
システムファイル	システムログファイル	同期コピー
	シンクポイントダンプファイル	
	ステータスファイル	

8.3.7　ログ同期方式

ログ同期方式を適用すると，リモートサイトへの更新コピー処理が次に示すようになります。

- データベースファイルの更新コピー処理は，初期構築時又はシステムログ適用化時に一度だけ行います。
- システムファイルの更新コピー処理は**同期コピー**で行います。

ログ同期方式では，**業務サイト**でトランザクションを受け付け，**ログ適用サイト**で業務サイトからコピーされたシステムログを基に，データベースの更新処理を行います。通常，メインサイトが業務サイト，リモートサイトがログ適用サイトとなります。

システムログファイルなどのデータベースの回復に必要な情報は，リモートサイトへの反映を保証するために同期コピーを行います。したがって，メインサイトの HiRDB が災害によって異常終了した場合，異常終了直前の状態でリモートサイトの HiRDB を再開始できます。ログ同期方式は，主に小，中規模システム向けの処理方式といえます。

データベースファイルについては，初期構築時又はシステムログ適用化時にだけコピーするため，ほかの方式に比べて通信量を削減できます。そのため，トランザクション性能への影響は小さくなります。なお，リモートサイト側でシステムログを基にデータベースを更新するため，常にリモートサイトの HiRDB を稼働しておく必要があります。

注意事項

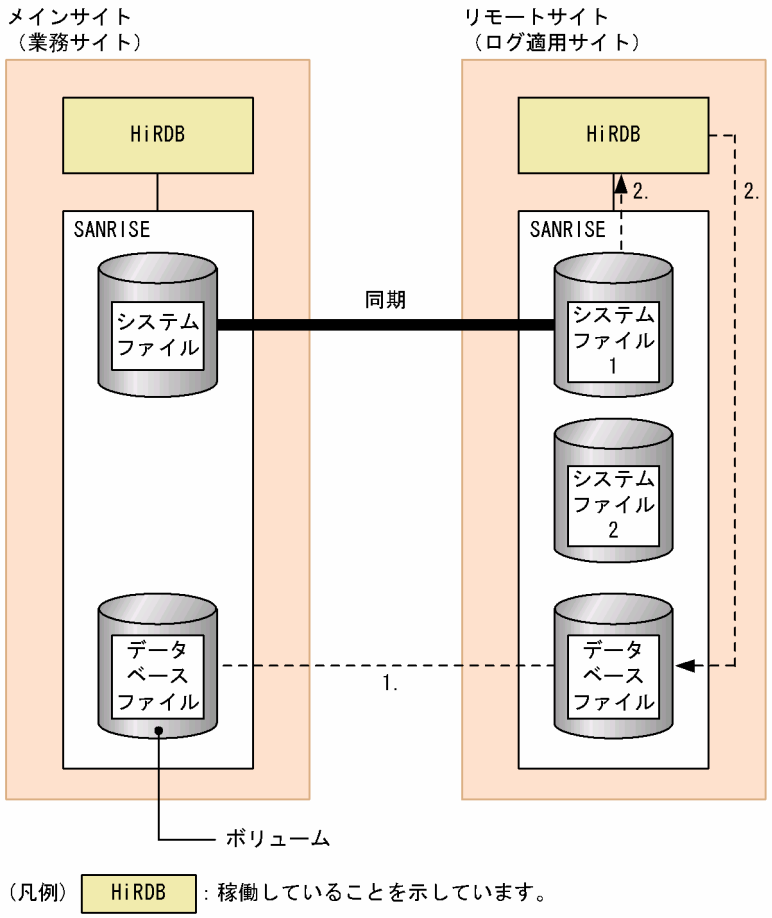
ログ同期方式を適用する場合，付加 PP の HiRDB Disaster Recovery Light Edition をインストールして，pdopsetup コマンドでセットアップする必要があります。

参考

ログ同期方式は，ほかの処理方式に比べて通信量を削減できますが，ハイブリッド方式より運用が難しくなります。

ログ同期方式の概要を次の図に示します。また，ログ適用サイトに更新コピーするときの処理方式（ログ同期方式の場合）を次の表に示します。

図 8–22　ログ同期方式の概要



〔説明〕

1. 初期構築時又はシステムログ適用化時だけデータベースファイルをコピーします。それ以外はコピーしません。
2. 業務サイトから同期コピーしたログを読み込んで，データベースへの更新処理を行います。これを**ログ適用**といいます。

表 8–10　ログ適用サイトに更新コピーするときの処理方式（ログ同期方式の場合）

ログ適用サイトにコピーされるファイル		更新コピーするときの処理方式
データベースファイル		初期構築時又はシステムログ適用化時だけコピー
システムファイル 1	システムログファイル	同期コピー
	正シンクポイントダンプファイル	
	正ステータスファイル	
システムファイル 2	副シンクポイントダンプファイル	コピーなし
	副ステータスファイル	

参考

正シンクポイントダンプファイル及び正ステータスファイルは，ログ適用処理で使用するファイルです（通常の HiRDB の運用に必要なシンクポイントダンプファイル及びステータスファイルのことです）。副シンクポイントダンプファイル及び副ステータスファイルは，ログ適用処理を行っているときのシンクポイントダンプ及びシステムステータス情報を取るためのファイルです。

注意事項

ログ同期方式の場合，全同期方式，全非同期方式，及びハイブリッド方式に比べて，システムファイル 2（副シンクポイントダンプファイル及び副ステータスファイル）が必要となります。したがって，副シンクポイントダンプファイル及び副ステータスファイルがあることを考慮して，ハードウェア構成を決めてください。

9 セキュリティ対策に関する機能

この章では，データベースのセキュリティ対策に関する機能について説明します。

9.1　機密保護機能

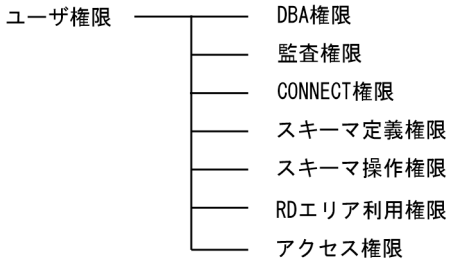
データベースを外部の人にアクセスされないように，HiRDB には**機密保護機能**があります。機密保護機能では**ユーザ権限**という概念を使用していて，必要な権限を持っていないとデータベースにアクセスできないようになっています。

ここでは，機密保護機能の概要について説明します。機密保護機能の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

9.1.1　ユーザ権限の種類

ここでは，HiRDB が設定しているユーザ権限の種類について説明します。HiRDB のユーザ権限の種類を次の図に示します。

図 9–1　HiRDB のユーザ権限の種類



これらの HiRDB のユーザ権限は，HiRDB 管理者，DBA 権限保持者，又はスキーマ所有者が，各ユーザに対して与えます。

HiRDB 管理者が与える権限：

自分自身の DBA 権限，監査権限，RD エリア利用権限

DBA 権限保持者が与える権限：

DBA 権限，スキーマ定義権限，RD エリア利用権限，CONNECT 権限

スキーマ所有者が与える権限：

アクセス権限，スキーマ操作権限

(1)　DBA 権限

DBA 権限，CONNECT 権限，及びスキーマ定義権限を与えたり，取り消したりするために必要な権限です。この権限を持っていると，次に示すことができます。

- ほかの人に DBA 権限，CONNECT 権限，及びスキーマ定義権限を与えられます。
- 与えた DBA 権限，CONNECT 権限，及びスキーマ定義権限を取り消せます。
- ほかの人のスキーマを定義できます。

スキーマを定義すると，スキーマ所有者は実表，ビュー表，インデクス，抽象データ型，ストアドプロシジャ，ストアドファンクション，及びトリガなどを定義できるようになります。

- ほかの人のスキーマ，実表，ビュー表，インデクス，抽象データ型，ストアドプロシジャ，ストアドファンクション，及びトリガなどを削除できます。
- CONNECT 関連セキュリティ機能に関する項目を定義できます。
- HiRDB に接続できます（CONNECT 権限を持っています）。

(2) 監査権限

監査人に必要な権限です。この権限を持っていると，監査証跡表，及び監査人に対する操作ができます。詳細は，マニュアル「HiRDB システム運用ガイド」の「ユーザ権限の種類」を参照してください。

セキュリティ監査機能を使用する場合に監査権限を設定します。セキュリティ監査機能については，「セキュリティ監査機能」を参照してください。

(3) CONNECT 権限

CONNECT 権限とは，HiRDB を利用するために必要な権限です。この権限を持っていると，データベースに接続（CONNECT）できるようになります。CONNECT 権限を持たないユーザが HiRDB を利用しようとするとエラーになります。

(4) スキーマ定義権限

スキーマ定義権限とは，スキーマを定義するために必要な権限です。この権限を持っていると，次に示すことができます。

- 自分のスキーマを定義できます。
  スキーマを定義すると，スキーマ所有者は実表，ビュー表，インデクス，抽象データ型，ストアドプロシジャ，ストアドファンクション，及びトリガなどを定義できるようになります。
- 自分のスキーマ，実表，ビュー表，インデクス，抽象データ型，ストアドプロシジャ，ストアドファンクション，及びトリガなどを削除できます。

(5) スキーマ操作権限

他ユーザが所有しているスキーマに対して，定義系 SQL 操作を行うために必要な権限です。

(6) RD エリア利用権限

RD エリア利用権限とは，RD エリアを使用するために必要な権限です。RD エリア利用権限がある RD エリアに表及びインデクスを定義できます。認可識別子を指定して RD エリア利用権限を与えた RD エリアを**私用 RD エリア**といい，PUBLIC を指定して RD エリア利用権限を与えた RD エリアを**公用 RD エリア**といいます。

(7)　アクセス権限

アクセス権限とは，表をアクセスするために必要な権限です。アクセス権限を持つユーザだけが表をアクセスできます。アクセス権限は，表単位に設定します。アクセス権限の種類を次の表に示します。

表 9–1　アクセス権限の種類

権限種別	内容
SELECT 権限	表の行データの検索（SELECT）を許可します。
INSERT 権限	表への行データの追加（INSERT）を許可します。
DELETE 権限	表の行データの削除（DELETE）を許可します。
UPDATE 権限	表の行データの更新（UPDATE）を許可します。

9.1.2　機密保護機能の運用方法

HiRDB 管理者，DBA 権限保持者，又はスキーマ所有者が，HiRDB のユーザに各種の権限を与えて，機密保護機能を運用します。権限の与え方については，マニュアル「HiRDB システム運用ガイド」を参照してください。

9.2　セキュリティ監査機能

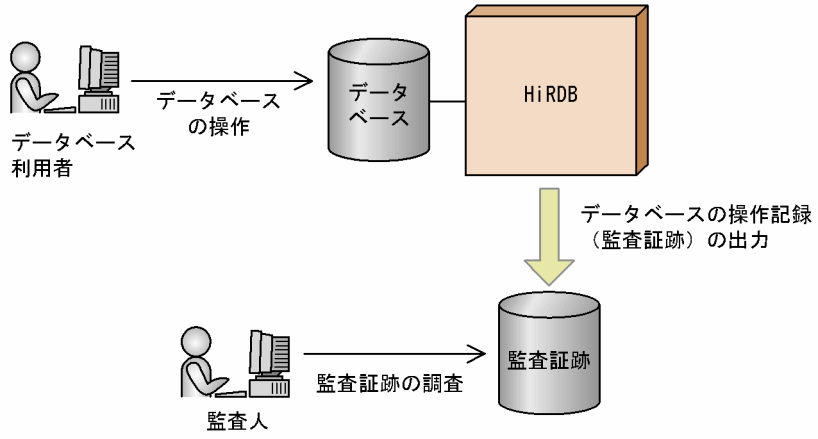
ここでは，セキュリティ監査機能の概要について説明します。セキュリティ監査機能の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

9.2.1　セキュリティ監査機能とは

(1)　機能概要

HiRDB のセキュリティは権限によって守られています。参照できる情報，更新できる情報，及び操作できるオブジェクト（表，インデクスなど）を権限によって制限しています。この権限の運用が適切に行われているかどうかをチェックするために，HiRDB ではデータベースに対する各種操作を記録できます。この機能を**セキュリティ監査機能**といい，出力される操作記録を**監査証跡**といいます。出力された監査証跡を調査して不正なアクセスが行われていないかをチェックできます。このチェックは**監査権限**を持つユーザ（これを**監査人**といいます）が行います。セキュリティ監査機能の概要を次の図に示します。

図 9–2　セキュリティ監査機能の概要



監査証跡には，だれがどのような権限を使用して何に対する操作を行ったかという情報が取得されます。どの操作に対して監査証跡を取得するかは，監査人が CREATE AUDIT 文で設定します。監査証跡の取得対象となる操作が実行されると，監査証跡が取得されます。

監査人は，監査証跡表を管理及び操作する主監査人と，監査証跡表を操作する副監査人の 2 種類があります。

主監査人はシステム内に１人，副監査人はシステム内に複数人登録できます。

参考

- セキュリティ監査機能はセキュリティを強化する機能ではありません。権限の運用が適切に行われているかどうかをチェックするための操作記録を出力する機能です。

- JP1/NETM/Audit と連携して，JP1/NETM/Audit で HiRDB の監査証跡を収集・一元管理できます。詳細は，「JP1/NETM/Audit」を参照してください。

(2) 監査証跡の取得契機

HiRDB が監査証跡を取得する契機を次に示します。

- コマンド又は SQL 文を実行するときの権限チェック時
- 各イベントの終了時

SQL の構文エラー時，及びコマンドの入力ミスによるエラー時は監査証跡を取得しません。

監査証跡の取得契機の詳細については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(3) 監査証跡の取得例

監査証跡の取得例を次に示します。

（例 1）表を検索した場合の監査証跡の取得例

表を検索した場合，表のアクセス権限（SELECT 権限）を使用するため，監査証跡が取得されます。

表の検索内容（SQL の指定）		監査証跡の内容				
		実行者	使用した権限	操作対象のオブジェクト種別	操作対象のオブジェクト名	操作種別
ユーザ（USR1）が次の SELECT 文を発行した場合 SELECT C1 FROM USR1.T1	権限	USR1	表のアクセス権限（SELECT 権限）	表	USR1.T1	表へのアクセス（SELECT）
	終了	USR1	−	表	USR1.T1	表へのアクセス（SELECT）
ユーザ（USR2）が次の SELECT 文を発行した場合 SELECT T1.C1,T2.C1 FROM USR1.T1 T1,USR2.T2 T2 WHERE T1.C1=T2.C1	権限	USR2	表のアクセス権限（SELECT 権限）	表	USR1.T1	表へのアクセス（SELECT）
		USR2	表のアクセス権限（SELECT 権限）	表	USR2.T2	表へのアクセス（SELECT）
	終了	USR2	−	表	USR1.T1	表へのアクセス（SELECT）
		USR2	−	表	USR2.T2	表へのアクセス（SELECT）

（凡例）

権限：権限チェック時に取得される監査証跡

終了：イベント終了時に取得される監査証跡

－：該当しません。

（例 2）表を定義又は削除した場合の監査証跡の取得例

表を定義又は削除した場合，スキーマ所有者の権限，表の所有者の権限，及び RD エリア利用権限を使用するため，監査証跡が取得されます。

表の検索内容（SQL の指定）		監査証跡の内容				
		実行者	使用した権限	操作対象のオブジェクト種別	操作対象のオブジェクト名	操作種別
ユーザ（USR1）が次の CREATE TABLE を発行した場合 CREATE TABLE T1(C1 INT) IN RDAREA1	権限	USR1	RD エリア利用権限	RD エリア	RDAREA1	定義作成
		USR1	所有者	スキーマ	USR1	定義作成
		USR1	所有者	表	USR1.T1	定義作成
	終了	USR1	－	表	USR1.T1	定義作成
ユーザ（USR2）が次の DROP TABLE を発行した場合 DROP TABLE T1	権限	USR2	所有者	表	USR2.T1	定義削除
	終了	USR2	－	表	USR2.T1	定義削除

（凡例）

権限：権限チェック時に取得される監査証跡

終了：イベント終了時に取得される監査証跡

－：該当しません。

(4)　監査証跡として取得する情報

監査証跡として取得する情報を次の表に示します。

表 9–2　監査証跡として取得する情報

取得する情報	説明
ユーザ識別子	監査対象のイベント実行者の認可識別子
イベント実行日	イベントを実行した年月日
イベント実行時刻	イベントを実行した時刻
イベント実行時刻	イベントを実行した時刻（単位：マイクロ秒）
イベントタイプ	イベントタイプ
イベントサブタイプ	イベントサブタイプ
イベント成否	イベントの実行結果（権限のチェックが成功したかどうかを出力します）
使用した権限	イベントを実行したときに使用した権限

取得する情報	説明
UAP 名称	クライアント環境定義の PDCLTAPNAME オペランドに指定した UAP 名称
サービス名	イベント発行元の UAP が要求したサービス名 OpenTP1 の SUP（サービス利用プログラム）が SPP（サービス提供プログラム）に要求したサービスの場合，又は TP1/Message Control が MHP（メッセージ処理プログラム）に要求したサービスの場合は，該当するサービス名称
IP アドレス	イベント発行元 UAP を実行したクライアントの IP アドレス※
プロセス番号	イベント発行元 UAP のプロセス ID※
スレッド番号	イベント発行元 UAP のスレッド ID※
ホスト名	イベント発行元 UAP の接続先ホスト名
ユニット識別子	イベント発行元 UAP の接続先ユニット識別子
サーバ名	イベント発行元 UAP の接続先フロントエンドサーバ名，又はシングルサーバ名
コネクト通番	イベント発行者のコネクト通番
SQL 通番	イベントの SQL 通番
オブジェクトの所有者名	イベントの権限チェックの対象になるオブジェクトの所有者名
オブジェクト名	イベントの権限チェックの対象になるオブジェクト名
オブジェクトの種別	イベントの権限チェックの対象になるオブジェクトの種別
付与，削除，又は変更した権限	イベントによって付与，削除，又は変更した権限
権限を付与，削除，又は変更されたユーザ識別子とイベント対象のユーザ識別子	イベントによって権限を付与，削除，又は変更されたユーザの識別子とイベント対象になった認可識別子
セキュリティ監査機能に関するオペランドの値	セキュリティ監査機能に関するオペランドの値（HiRDB 開始時の値）
監査証跡種別	権限チェックか，又はイベント終了かを示す種別
SQL コード又は終了コード	SQL，ユティリティ，コマンド終了時のコード
スワップ元の監査証跡ファイル名	スワップ発生時のスワップ元の監査証跡ファイル名
スワップ先の監査証跡ファイル名	スワップ発生時のスワップ先の監査証跡ファイル名
CONNECT 関連セキュリティ機能の設定変更種別	CONNECT 関連セキュリティ機能の設定変更種別（パスワードの変更時にも変更種別が設定されます）
CONNECT 関連セキュリティ機能に関するオペランドの値（変更前）	変更前の CONNECT 関連セキュリティ機能に関するオペランドの値
CONNECT 関連セキュリティ機能に関するオペランドの値（変更後）	変更後の CONNECT 関連セキュリティ機能に関するオペランドの値
監査証跡表オプション	イベントの操作対象が監査証跡表，監査証跡表を基表としたビュー表，又は監査証跡表を基表としたリストの場合のフラグ

取得する情報	説明
アクセス件数	イベントによってオブジェクト（実表，ビュー表，及びリスト）に対して検索，挿入，更新，及び削除をした行数
SQL 文	実行した SQL 文
SQL データ	実行した SQL のデータ
ユーザ付加情報 1	ユーザが任意に設定する付加情報
ユーザ付加情報 2	
ユーザ付加情報 3	
関連製品付加情報 1	Cosminexus が設定する付加情報

注

イベントによって取得する情報が異なります。イベントごとに取得する情報の一覧については，マニュアル「HiRDB システム運用ガイド」を参照してください。

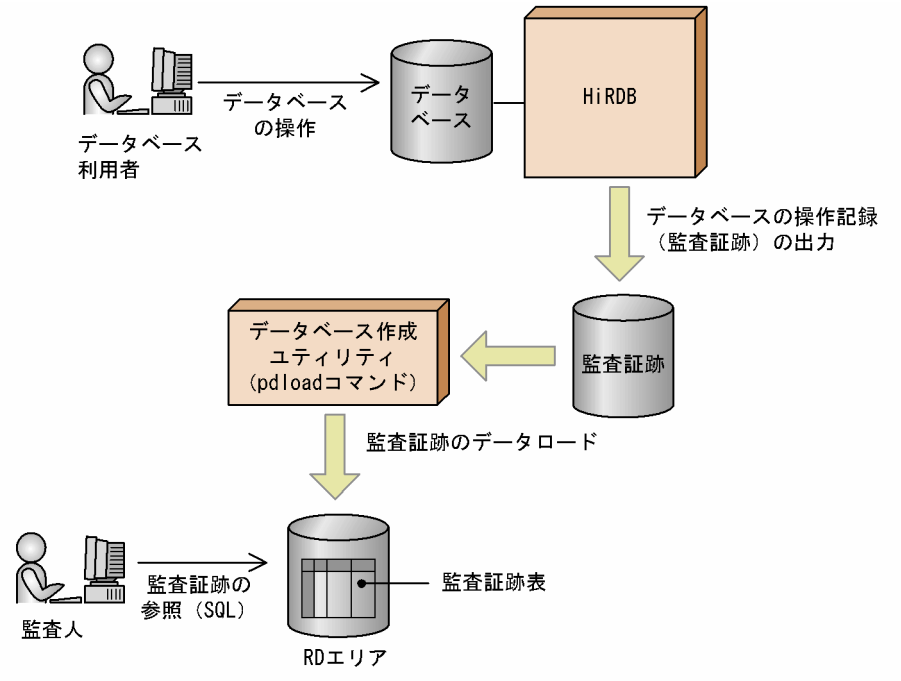
注※

Open/TP1 配下のアプリケーションを介している場合，又は Web サーバなどの製品を介している場合は，エンドユーザが実行しているアプリケーションの情報ではなく，HiRDB に接続しているアプリケーションの情報が取得されます。

(5)　監査証跡の参照

監査証跡は**監査証跡ファイル**に出力されます。監査証跡ファイル中のデータをデータベース作成ユティリティ（pdload コマンド）で，**監査証跡表**にデータロードした後に SQL で参照できます。なお，監査人はこの監査証跡表を参照できます（更新はできません）。監査人以外のユーザは，主監査人に参照権限を与えてもらえば監査証跡表を参照できます（更新はできません）。監査証跡の参照方法を次の図に示します。

図 9–3　監査証跡の参照方法



〔説明〕

1. 監査対象のイベントが実行された場合，監査証跡ファイルに監査証跡が出力されます。監査証跡ファイルは監査証跡ファイル用の HiRDB ファイルシステム領域内に作成されます。監査対象のイベントについては，「監査対象になるイベント」を参照してください。
2. 監査証跡ファイルに出力された監査証跡を入力情報にして，データベース作成ユティリティ（pdload コマンド）のデータロードで表にデータを登録します。なお，監査証跡表の自動データロード機能を適用すると，データベース作成ユティリティの実行は HiRDB が自動で行います。
3. 監査人は監査証跡表を利用して監査を実施します。

9.2.2　監査対象になるイベント

監査証跡の取得対象になる操作を**監査対象イベント**といいます。監査対象イベントを次の表に示します。

表 9–3　監査対象イベント

イベントの種類	説明及び監査対象になるイベント	選択可否
システム管理者セキュリティイベント	1. HiRDB 管理者又は DBA 権限保持者が行うセキュリティ対象イベントを監査対象にします。 2. CONNECT 関連セキュリティ機能の設定値の変更を監査対象にします。 3. システムが自動的に行うセキュリティ対象イベントを監査対象にします。 次に示すイベントが実行されたときに監査証跡を出力します。 • HiRDB の開始（pdstart コマンド）※1 • HiRDB の終了（pdstop コマンド）※1※2	選択不可（必ず監査証跡が出力されます）

イベントの種類	説明及び監査対象になるイベント	選択可否
	• 監査人の登録（pdmod コマンド） • 監査証跡表の作成（pdmod コマンド） • 監査証跡ファイルの削除（pdaudrm コマンド）※3 • 監査証跡の取得開始※5 • 監査証跡の取得終了※6 • 監査証跡ファイルの上書き開始 • 連続認証失敗アカウントロック状態への遷移 • 連続認証失敗アカウントロック状態の解除 次に示す場合が該当します。 ・アカウントロック期間経過後の CONNECT 時 ・DROP CONNECTION SECURITY の実行時 ・pdacunlck コマンドの実行時 • パスワード無効アカウントロック状態への遷移 • パスワード無効アカウントロック状態の解除 • CONNECT 関連セキュリティ機能の設定値の変更 ・連続認証失敗許容回数 ・アカウントロック期間 ・パスワードの文字列制限で設定する項目（事前チェックも含む） ・IP アドレスによる接続制限の登録又は削除 ・パスワードの有効期間の設定 • pdacunlck コマンドの実行	
監査人セキュリティイベント	監査人が行うセキュリティ対象イベントを監査対象にします。次に示すイベントが実行されたときに監査証跡を出力します。 • 監査証跡表へのデータロード（pdload コマンド） • 監査証跡ファイルのスワップ（pdaudswap コマンド） • 監査対象イベントの定義（CREATE AUDIT）※4 • 監査対象イベントの削除（DROP AUDIT）※4 • 監査人のパスワード変更（GRANT AUDIT）※4 • JP1/NETM/Audit 用監査ログ出力ファイルへの出力（pdaudput コマンド）	選択不可（必ず監査証跡が出力されます）
セッションセキュリティイベント	認可識別子とパスワードによるユーザ認証を監査対象にします。次に示すイベントが実行されたときに監査証跡を出力します。 • HiRDB への接続（CONNECT 文） • ユーザの変更（SET SESSION AUTHORIZATION 文） • ロールの設定（SET ROLE） • HiRDB との切り離し（DISCONNECT 文）※9	選択可能
権限管理イベント	ユーザ権限の付与又は削除を監査対象にします。次に示すイベントが実行されたときに監査証跡を出力します。 • ユーザ権限の付与（GRANT 文） • ユーザ権限の削除（REVOKE 文）	選択可能※7
オブジェクト定義イベント	オブジェクトの定義，削除，又は変更を監査対象にします。次に示すイベントが実行されたときに監査証跡を出力します。	選択可能※7

イベントの種類	説明及び監査対象になるイベント	選択可否
	• オブジェクトの定義（次に示す SQL が対象になります） CREATE FUNCTION CREATE INDEX CREATE PROCEDURE CREATE PUBLIC VIEW CREATE ROLE CREATE SCHEMA CREATE SEQUENCE CREATE TABLE CREATE TRIGGER CREATE TYPE CREATE VIEW • オブジェクトの削除（次に示す SQL が対象になります） DROP DATA TYPE DROP FUNCTION DROP INDEX DROP PROCEDURE DROP PUBLIC VIEW DROP ROLE DROP SCHEMA DROP SEQUENCE DROP TABLE DROP TRIGGER DROP VIEW • オブジェクトの変更（次に示す SQL が対象になります） ALTER INDEX ALTER PROCEDURE ALTER ROUTINE ALTER TABLE ALTER TRIGGER COMMENT	
オブジェクト操作イベント	オブジェクトの操作を監査対象にします。次に示すイベントが実行されたときに監査証跡を出力します。 • 表の検索（SELECT 文） • 表への行挿入（INSERT 文） • 表の行更新（UPDATE 文） • 表からの行削除（DELETE 文） • 表の全行削除（PURGE TABLE 文） • ストアドプロシジャの実行（CALL 文） • 表の排他制御（LOCK TABLE 文） • リストの作成（ASSIGN LIST 文） • 順序数生成子が生成する値の返却（NEXT VALUE 式）	選択可能※7

イベントの種類	説明及び監査対象になるイベント	選択可否
ユティリティ操作イベント	ユティリティ又はコマンドによるオブジェクト操作に関するセキュリティ対象イベントを監査対象にします。次に示すイベントが実行されたときに監査証跡を出力します。 • データベース作成ユティリティ（pdload コマンド） 対象オブジェクト：TABLE，SEQUENCE • pddefrev コマンド 対象オブジェクト：PROCEDURE，TABLE，TRIGGER，VIEW • データベース再編成ユティリティ（pdrorg コマンド） 対象オブジェクト：TABLE • ディクショナリ搬出入ユティリティ（pdexp コマンド） 対象オブジェクト：PROCEDURE，TABLE，TRIGGER，VIEW • 整合性チェックユティリティ（pdconstck コマンド） 対象オブジェクト：TABLE	選択可能※7※8

注※1

HiRDB/パラレルサーバのサーバ単位の開始又は終了は監査対象イベントになりません。

注※2

正常終了又は計画停止を監査対象イベントとします。強制終了又は異常終了は監査対象イベントになりません。強制終了又は異常終了を監査するには，HiRDB が出力するメッセージ，又は OS が出力するメッセージを使用してください。

監査対象にならない終了コマンドを次に示します。

- pdstop -f
- pdstop -f -q
- pdstop -f -x ホスト名
- pdstop -f -u ユニット識別子
- pdstop -f -s サーバ名
- pdstop -f -u ユニット識別子 -s サーバ名
- pdstop -z
- pdstop -z -q
- pdstop -z -c
- pdstop -z -s サーバ名

注※3

監査証跡ファイルの作成は監査対象イベントになりません。監査証跡ファイルの作成を監査するには OS の監査機能を使用してください。

注※4

データベース定義ユティリティ（pddef コマンド）又は HiRDB SQL Executer で実行した場合も監査証跡を出力します。

注※5

pdaudbegin コマンドの実行時又は HiRDB の開始時から監査証跡を取得する場合に監査証跡を出力します。

注※6

pdaudend コマンドの実行時又は監査証跡を取得している状態で，HiRDB を正常終了又は計画停止する場合に監査証跡を出力します。

注※7

権限管理イベント，オブジェクト定義イベント，オブジェクト操作イベント，及びユティリティ操作イベント中のイベント対象オブジェクトが，監査証跡表，監査証跡表を基表としたビュー表，又は監査証跡表を基としたリストの場合，イベント終了時の監査証跡は無条件に出力されます。権限チェック時の監査証跡を取得するかどうかは選択できます。

注※8

データベース再編成ユティリティ（pdrorg コマンド）でディクショナリ表のリロードをする場合，監査証跡は無条件に出力されます。

注※9

次の場合を監査証跡イベントとします。

- シングルサーバ又はフロントエンドサーバのサーバプロセスが DISCONNECT を検知した場合
- シングルサーバ又はフロントエンドサーバのサーバプロセスが内部的に DISCONNECT を実行する場合

9.3　CONNECT 関連セキュリティ機能

ここでは，CONNECT 関連セキュリティ機能の概要について説明します。CONNECT 関連セキュリティ機能の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

9.3.1　CONNECT 関連セキュリティ機能とは

システムのセキュリティを強化する仕組みの一つにパスワードがあります。HiRDB もユーザごとにパスワードを設定できますが，類推しやすい簡単なパスワードを使用している場合（例えば，認可識別子をそのままパスワードにしたり，誕生日をパスワードにしたりした場合），不正なユーザがそのパスワードを使用してシステムに侵入する可能性が高くなります。パスワードの不正使用を防止するために，**CONNECT 関連セキュリティ機能**の使用をお勧めします。CONNECT 関連セキュリティ機能の概要を次の表に示します。

表 9–4　CONNECT 関連セキュリティ機能の概要

機能	説明
パスワードの文字列制限	パスワードに指定する文字列に制限を設定できます。例えば，012345 や，aaaaa などのパスワードを禁止できます。簡単なパスワードを禁止することでパスワードのセキュリティを強化します。
連続認証失敗回数の制限	不正なパスワードを使用し，ユーザ認証に連続して失敗した場合，そのユーザを HiRDB に接続（CONNECT）できないようにします。連続して失敗した回数の上限を設定し，その回数を超えたユーザは HiRDB に接続できなくなります。 例えば，不正なパスワードを使用して 3 回連続ユーザ認証に失敗した場合，そのユーザは HiRDB に接続できなくなります。
IP アドレスによる接続制限	HiRDB サーバへの接続を，IP アドレス単位で許可又は拒否できます。
パスワードの有効期間の設定	パスワードを使用できる期間（日数）を設定し，その期間までにパスワードを変更しなければ HiRDB に接続（CONNECT）できないようにします。

これらの機能を組み合わせることで，パスワードの類推によるパスワードの不正使用が難しくなり，システムのセキュリティを強化できます。

9.3.2　パスワードの文字列制限

(1)　パスワードに設定できる制限

パスワードに設定できる制限を次の表に示します。

表 9–5　パスワードに設定できる制限

項目	説明
最小許容バイト数の設定	パスワードの最小許容バイト数を設定できます。

項目	説明
認可識別子の指定禁止	パスワードの文字列中に認可識別子を指定することを禁止できます。 大文字，小文字は区別しません。
単一文字種※の指定禁止	パスワードを同じ文字種（例えば，英大文字だけ，数字だけ）だけで構成することを禁止できます。
必須文字種	パスワードを構成する文字に必須となる文字種※を設定します。
再利用禁止	過去に指定したパスワードを，新規パスワードとして再利用することを禁止します。再利用禁止として残すパスワード履歴の個数を指定します。

注※　パスワードに指定できる文字種は次のように分類されています。

- 英大文字：A～Z，#，@，¥
- 英小文字：a～z
- 数字：0～9

参考

パスワードの文字列制限はユーザ単位に設定できません。HiRDB の全ユーザ（DBA 権限保持者や，監査人も含む）に対して一律の適用になります。また，簡易認証ユーザは，パスワードの文字列制限の対象外となります。

(2)　既存ユーザに対する影響

パスワードの文字列制限を設定した場合，制限に違反しているユーザは**パスワード無効アカウントロック状態**になります。パスワード無効アカウントロック状態のユーザは HiRDB に接続（CONNECT）できなくなります。

パスワード無効アカウントロック状態を解除するには，パスワードを変更する必要があります。

また，パスワードの文字列制限を設定する前に，制限に違反するためパスワード無効アカウントロック状態になるユーザがどのくらいいるかを調査できます。

(3)　新規ユーザに対する影響

GRANT DBA，GRANT AUDIT，又は GRANT CONNECT でパスワードを設定しますが，そのパスワードが制限に違反している場合，GRANT 文を実行できません。

(4)　設定方法

CREATE CONNECTION SECURITY でパスワードの文字列制限を設定します。

(5)　データディクショナリ用 RD エリア容量の確認

パスワードの再利用禁止を設定する場合，ディクショナリ表 SQL_USERS の容量の再見積もりをしてください。

9.3.3　連続認証失敗回数の制限

(1)　設定できる制限

不正なパスワードを使用し，ユーザ認証に連続して失敗した場合，そのユーザを HiRDB に接続（CONNECT）できないようにします。連続して失敗できる回数の上限（**連続認証失敗許容回数**）を設定し，その回数を超えたユーザは HiRDB に接続できなくなります。

例えば，連続認証失敗許容回数を 3 と設定した場合，パスワード不正によって 4 回連続でユーザ認証に失敗すると，そのユーザは**連続認証失敗アカウントロック状態**になります。連続認証失敗アカウントロック状態のユーザは HiRDB に接続（CONNECT）できなくなります。

参考

連続認証失敗回数の制限はユーザ単位に設定できません。HiRDB の全ユーザ（DBA 権限保持者や，監査人も含む）に対して一律の適用になります。

また，連続認証失敗アカウントロック状態とする期間（**アカウントロック期間**）を設定できます。例えば，アカウントロック期間を 1 時間とした場合，連続認証失敗アカウントロック状態が 1 時間続きます。1 時間を過ぎると連続認証失敗アカウントロック状態が解除されて，HiRDB に接続できるようになります。

参考

- アカウントロック期間を無期限にすることもできます。
- アカウントロック期間内に連続認証失敗アカウントロック状態を解除することもできます。

(2)　設定方法

CREATE CONNECTION SECURITY で連続認証失敗回数の制限を設定します。

9.4　HiRDB 接続時のパスワード秘匿化機能

ここでは，HiRDB 接続時のパスワード秘匿化機能の概要について説明します。HiRDB 接続時のパスワード秘匿化機能の運用方法については，マニュアル「HiRDB システム運用ガイド」の「HiRDB 接続時のパスワード秘匿化機能の運用」を参照してください。

9.4.1　HiRDB 接続時のパスワード秘匿化機能の概要

HiRDB 接続時に，クライアントとサーバ間のネットワーク上で送受信するパスワードを秘匿化します。この機能によって，悪意のあるユーザがパケットスニファリングなどで不正にパケットを取得し，不正な電文の作成・送信することで HiRDB へ不正にアクセスし，接続することを防止します。

この機能を使用しない場合，パスワード認証方式（PA（Password Authentication）方式）を利用し，パスワードをスクランブルして HiRDB サーバへ送信しますが，この機能を使用する場合，チャレンジハンドシェイク認証方式（CHA（Challenge Handshake Authentication）方式）を使用してパスワードの秘匿化します。CHA 方式では，チャレンジレスポンス方式によってパスワードを秘匿化し，ネットワークにパスワードそのものを流さないようにします。

CHA 方式では，HiRDB サーバへの接続ごとに生成する乱数キーを用いてパスワードを秘匿化するので，HiRDB への不正な接続の防止を強化できます。

この機能を使用するためにはシステム共通定義 pd_connect_auth_type オペランドとクライアント環境定義 PDAUTHTYPE の設定をしてください。

この機能を使用する（CHA 方式を用いて接続する）ためには，Version 09-50-08 以降の HiRDB クライアントを使用してください。

9.4.2　HiRDB 接続時のパスワード秘匿化機能の適用範囲

HiRDB 接続時のパスワード秘匿化機能の適用範囲を次の表に示します。

表 9–6　秘匿化の対象となるパスワードと秘匿化の対象とならないパスワード

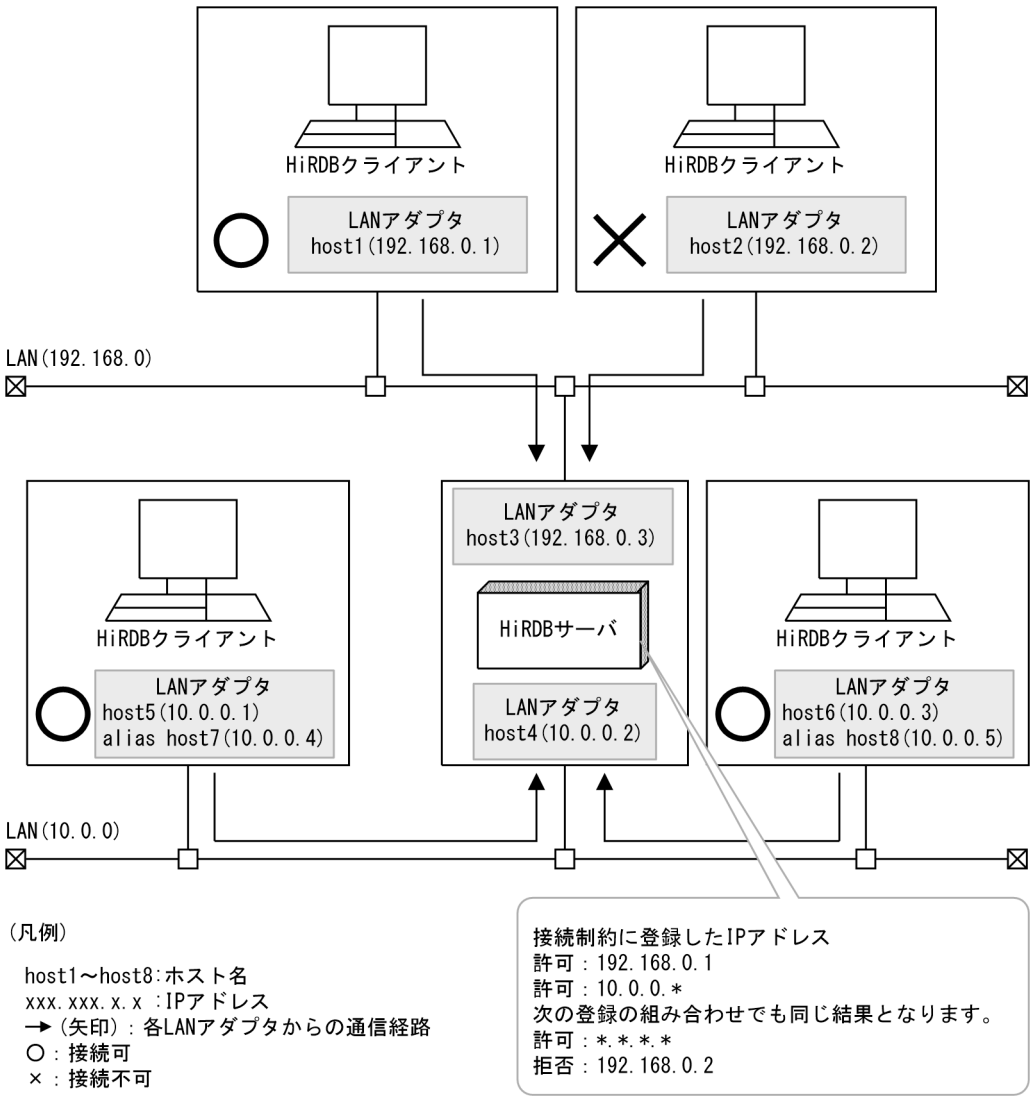
秘匿化の対象となるパスワード	秘匿化の対象とならないパスワード
• CONNECT 文実行時のパスワード • 自動再接続機能による再接続時のパスワード	• GRANT 文のパスワード • SET SESSION AUTHORIZATION 文のパスワード

秘匿化対象とならない SQL は，不正にパケットを取得されないようにセキュアエリアで実行してください。

9.5　IP アドレスによる接続制限

HiRDB クライアントに複数の IP アドレスが設定されている場合の HiRDB クライアント，HiRDB サーバ間通信では，HiRDB サーバに接続を許可する IP アドレス，及び接続を拒否する IP アドレスの登録ができます。登録には IP アドレスを指定します。UAP 実行時の HiRDB サーバへの接続では，登録された IP アドレスを使っての接続であるかどうかの判定によって接続の制限ができます。機能概要図を次に示します。

図 9–4　IP アドレスによる接続制限機能の概要



図の例では，IP アドレス 192.168.0.1 からの接続及びワイルドカード指定の 10.0.0.*を満たす IP アドレスからの接続を許可し，それ以外の IP アドレスからの接続は拒否します。また，拒否登録との組み合わせで同じ結果を実現することもできます。図の例では，すべての IP アドレスを示すワイルドカード指定の*.*.*.*を許可する登録と，IP アドレス 192.168.0.2 を拒否する指定の組み合わせとなります。

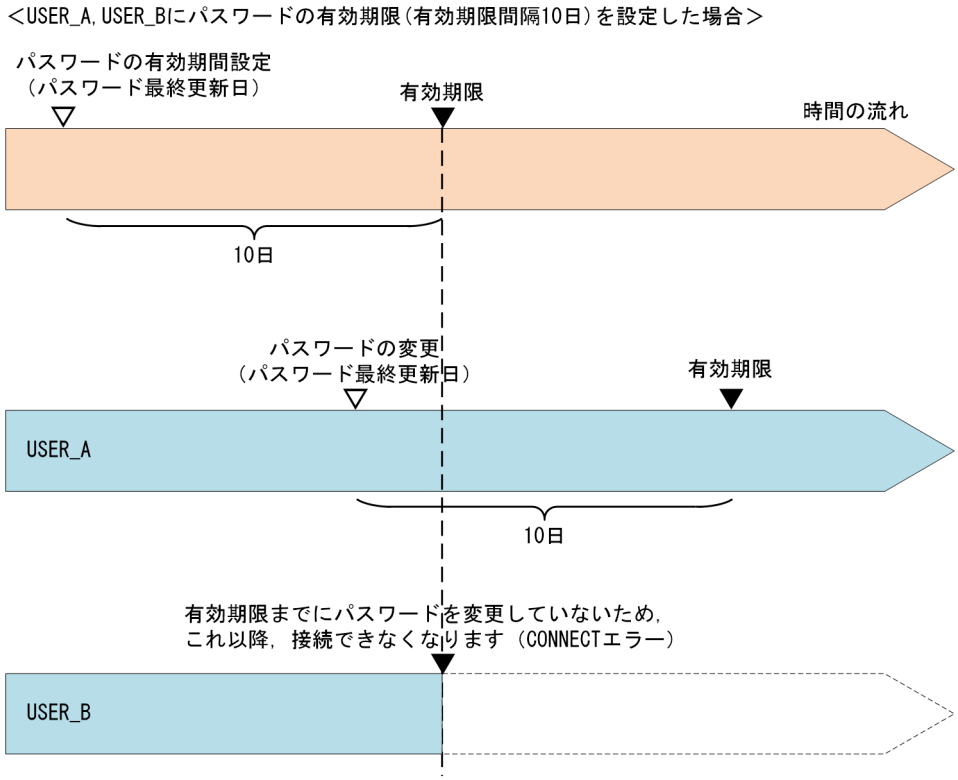
host1 及び host2 のように異なる HiRDB クライアントマシンからの接続について許可及び拒否を決めることができます。また，host5～host8 のように同一のエリアネットワーク LAN（10.0.0）からの接続を許可する設定ができます。

接続制限はすべてのユーザに適用されますが，特定のユーザにだけ接続制限を設定することもできます。

9.6　パスワードの有効期間

任意のユーザに対し，パスワードを使用できる期間（有効期限間隔）を日数で設定します。有効期限間隔を設定した日，または最後にパスワードを設定した日を「パスワードの最終更新日」として，有効期限を算出する起点の日とします。パスワードの最終更新日から有効期限以内は，同じパスワードで HiRDB に CONNECT ができます。概要を以下に示します。

図 9–5　パスワードの有効期間の概要



注　パスワードの最終更新日は，YYYYMMDDHHMMSSの単位で管理します。

図の例では，USER_A，USER_B ともに，有効期間設定日がパスワードの最終更新日となり（2019/04/01 09:00:00 とします），その 10 日経過後がパスワード有効期限（2019/04/11 09:00:00）となります。

USER_A の場合，有効期限以内に GRANT 文でパスワードを変更します。USER_A は，パスワードを変更した日がパスワードの最終更新日となり（2019/04/10 16:47:33 とします），その 10 日経過後がパスワード有効期限（2019/04/20 16:47:33）となります。

USER_B の場合，10 日経過後もパスワードを変更しません。USER_B は有効期限（2019/04/11 09:00:00）からあとの CONNECT はできなくなります。

10 プラグイン

この章では，HiRDB のプラグインの概要，機能及びプラグイン使用時の HiRDB の機能について説明します。

10.1　HiRDB のプラグインの概要

HiRDB では，文書，空間データなどのマルチメディアデータの「抽象データ型」及び「データを高速に操作できる機能」を使える仕組みである**プラグインアーキテクチャ**を採用しています。プラグインアーキテクチャによって，プラグインを HiRDB に登録するだけで，機能を拡張でき，マルチメディアデータを扱えるシステムを構築できます。

また，アプリケーション側では，マルチメディアデータ特有の複雑な操作内容を意識しないで高速に操作できるため，顧客への情報サービスなども効率良く提供できるようになります。HiRDB のプラグインを次の表に示します。

表 10–1　HiRDB のプラグイン

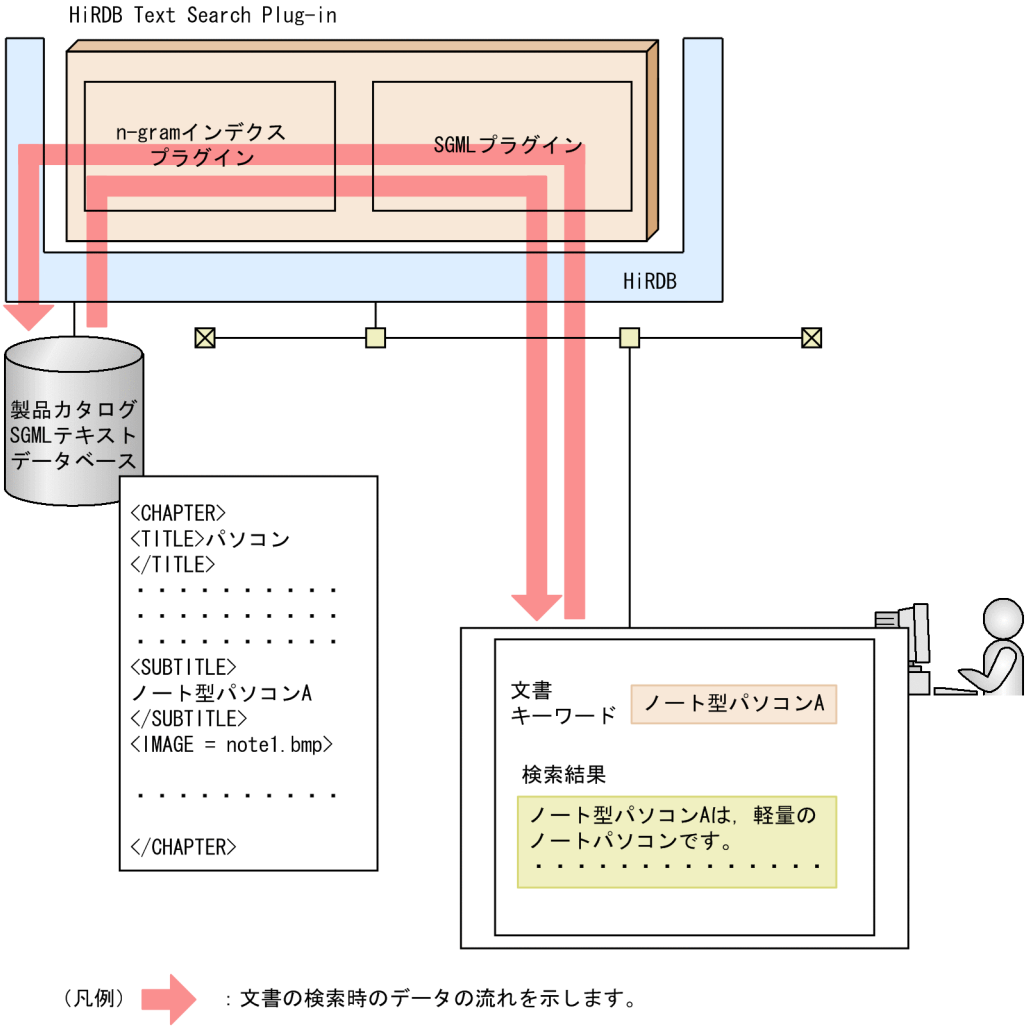
プラグイン名	説明
全文検索プラグイン HiRDB Text Search Plug-in	大量の文書情報を正確かつ高速に検索できる機能を提供するプラグインです。
空間検索プラグイン HiRDB Spatial Search Plug-in	地図情報などの空間データ（2 次元データ）を検索できる機能を提供するプラグインです。
HiRDB XML Extension	XML 文書を，構造を保持したままデータベースに格納できます。

HiRDB のプラグイン製品については，該当する HiRDB のプラグインのマニュアルを参照してください。

10.2　プラグインの業務への適用

例えば，企業で，製品のカタログのデータをデータベースで管理したい場合を想定します。パソコンのカタログには，デスクトップ型，ノート型などの種類，CPU，メモリなどの文書データがあります。これらの文書データを HiRDB のデータベースに格納して，HiRDB Text Search Plug-in を HiRDB に登録して使用すると，知りたいキーワードを含む文書データを高速に検索できます。プラグインの業務への適用を次の図に示します。

図 10–1　プラグインの業務への適用



〔説明〕

- プラグインのパッケージ製品である HiRDB Text Search Plug-in（SGML プラグインと n-gram インデクスプラグイン）を使用した例です。
- 「ノート型パソコン A」というキーワードを基に，その文書データを検索できます。

10.3　HiRDB のプラグインの機能

HiRDB には，次に示すプラグイン製品があります。

- 全文検索プラグイン（HiRDB Text Search Plug-in）
- 空間検索プラグイン（HiRDB Spatial Search Plug-in）
- HiRDB XML Extension

それぞれのプラグインを使用すると，どのようなことができるかを説明します。

10.3.1　全文検索プラグイン（HiRDB Text Search Plug-in）

HiRDB Text Search Plug-in の全文構造検索機能には，次に示す機能があります。

- SGML/XML 文書登録
- フラット文書登録
- 構造指定検索
- 同義語・異表記語検索
- スコア検索
- 検索結果テキストデータ取り出し
- 検索ヒット位置ハイライト表示タグ埋め込み

それぞれの機能について説明します。

(1)　SGML/XML 文書の登録

HiRDB Text Search Plug-in のユティリティを使用して，SGML/XML 文書の構造や要素を表すタグ名称などを定義する DTD ファイルを HiRDB のデータベースに登録できます。登録した DTD ファイルを基に，コンストラクタ関数 SGMLTEXT を使用すると，SGML/XML 文書を文書構造の情報と一緒に HiRDB のデータベースに登録できます。

(2)　フラット文書の登録

構造を持たないフラットな文書を HiRDB のデータベースに登録できます。

(3)　構造指定検索

抽象データ型関数 contains を使用して，検索対象の列と，検索する条件（検索対象の文書構造名，検索したいキーワードを指定した条件式）を指定すると，SGML/XML 文書を全文検索できます。

(4)　同義語・異表記語検索

HiRDB Text Search Plug-in のユティリティを使用して，同義語・異表記語辞書をローカルファイルに登録できます。登録した同義語・異表記語辞書を基に，SGML/XML 文書の全文検索時に，検索したいキーワードの同義語又は異表記語のキーワードを検索できます。例えば，「コンピュータ」というキーワードを検索すると，その同義語の「電子計算機」や「COMPUTER」，異表記語の「コンピューター」や「Computer」なども検索できます。

(5)　スコア検索

HiRDB Text Search Plug-in が提供する抽象データ型関数 contains_with_score 及び score を使用して，検索したキーワードの発生頻度から得点（スコア）を算出し，得点順に検索結果を表示できます。

10.3.2　空間検索プラグイン（HiRDB Spatial Search Plug-in）

HiRDB Spatial Search Plug-in には，次に示す機能があります。

- 空間データの登録
- 空間データの検索

(1)　空間データの登録

HiRDB Spatial Search Plug-in が提供する抽象データ型関数 GEOMETRY を使用して，地図上の駅や銀行などの位置情報を空間データ（X 座標，Y 座標を指定した 2 次元データ）として定義し，HiRDB のデータベースに登録できます。

(2)　空間データの検索

HiRDB Spatial Search Plug-in が提供する抽象データ型関数 Within を使用して，検索条件式に指定した図形と GEOMETRY 型の図形の空間関係を比較して，指定した図形が含まれるかどうかを判定できます。例えば，「駅から半径 100 メートル以内にある銀行を探す」という検索ができます。

また，抽象データ型関数 IntersectsIn を使用して，検索条件式に指定した図形と GEOMETRY 型の図形の空間関係を比較して，指定した図形が含まれる又は交差するかどうかを判定できます。例えば，「ある地域内を通過する道路，線路，河川などを探す」という検索ができます。

10.3.3　HiRDB XML Extension

XML Extension では，業務間のインタフェースやコンテンツの標準的なフォーマットとして利用される XML データを，XML 型という抽象データ型でそのまま利用できます。

XML 型はデータ構造の定義が不要なため，XML 型のデータ構造を変更した場合でも，表の定義を変更する必要はありません。XML の持つ高い柔軟性をそのまま生かすことで，容易にデータを管理できます。このため，企業や様々な研究機関で保有している知的財産としての大量の文書や，取り扱うコンテンツのデータ構造に変更があった場合でも，システムの変更を最小限に抑え業務への影響を少なくできます。

また，検索についても標準的な XML への問い合わせ言語である XQuery を利用する SQL/XML の主要な機能をサポートしています。

10.4　プラグイン使用時に HiRDB で設定する項目

プラグイン使用時に HiRDB で設定する項目を次に示します。

- **プラグインのセットアップ／登録**
  HiRDB にプラグインをセットアップ／登録して，プラグインを使用するための準備をします。
- **レジストリ機能の初期設定**
  レジストリ情報を登録する前に，レジストリ機能の初期設定をします。
- **プラグイン使用時の表の定義**
  プラグインが提供する抽象データ型やインデクス型を使用して，オブジェクトリレーショナルデータベースを作成します。
- **プラグインインデクスの遅延一括作成**
  行データを追加（更新）したときに，プラグインインデクスのデータ追加（更新）処理をしないで，データベース再編成ユティリティ（pdrorg）を使用して後で一括してプラグインインデクスのデータ追加（更新）処理ができます。

ここでは，プラグイン使用時の，HiRDB のそれぞれの設定項目について説明します。

10.4.1　プラグインのセットアップ／登録

プラグインを使用するためには，プラグインをインストールし，HiRDB の環境にセットアップして，登録する必要があります（Windows 版の場合，プラグインのセットアップは不要です）。

HiRDB へのセットアップには **pdplgset** コマンドを，登録には **pdplgrgst** コマンドを使用します。

プラグインのセットアップ及び登録方法についてはマニュアル「HiRDB システム導入・設計ガイド」を，プラグインのインストールについては該当するプラグインのマニュアルを参照してください。

10.4.2　レジストリ機能の初期設定

例えば，SGML 文書など，文書自体に「章・節・項，箇条書き」という構造を持っています。このように個々のデータ自体が構造を持っている場合には，ユーザが文書登録時に「そのデータがどのような構造を持っているか」をプラグインに認識させる必要があります。このような「データ操作時にプラグインが使用するためのプラグイン固有の情報」を**レジストリ情報**といいます。また，レジストリ情報を HiRDB が保持する機能を**レジストリ機能**といいます。

プラグインのレジストリ機能を使用するためには，HiRDB のレジストリ機能初期設定ユティリティ（pdreginit）を使用してレジストリ機能の初期設定をする必要があります。レジストリ機能を初期設定すると，「レジストリ用 RD エリア及びレジストリ LOB 用 RD エリア」が作成され，「レジストリ情報を管理

する表（**レジストリ管理表**）」及び「レジストリ管理表に情報を登録したりする，操作用のストアドプロシジャ」が HiRDB の RD エリアに格納されます。

レジストリ機能についての情報を格納するための RD エリアを次の表に示します。

表 10–2　レジストリ機能についての情報を格納するための RD エリア

レジストリ機能の情報	情報を格納する RD エリア
レジストリ管理表	レジストリ用 RD エリア又はレジストリ LOB 用 RD エリア※
レジストリ管理表に情報を登録するなどの操作用のストアドプロシジャ	データディクショナリ LOB 用 RD エリア

注※

どちらの RD エリアに格納されるかは，登録されるデータの長さ（32,000 バイト未満か以上か）によって自動的に決定されます。

レジストリ機能の初期設定についてはマニュアル「HiRDB システム導入・設計ガイド」を，レジストリ機能設定ユティリティ（pdreginit）についてはマニュアル「HiRDB コマンドリファレンス」を参照してください。

10.4.3　プラグイン使用時の表の定義

プラグインをセットアップ／登録すると，プラグインから抽象データ型やインデクス型が提供されます。これによって，ユーザが抽象データ型で複雑なデータ構造及び操作を定義しなくてもマルチメディア情報などの複雑なデータを扱えるオブジェクトリレーショナルデータベースを構築できます。

表を定義するには，プラグインが提供する抽象データ型を表の列のデータ型として指定します。インデクスを定義するには，プラグインが提供するインデクス型をインデクスとして指定します。プラグインが提供するインデクス型を指定したインデクスを**プラグインインデクス**といいます。プラグインインデクスの機能は，個々のプラグインに依存します。プラグインインデクスについては，該当するプラグインのマニュアルを参照してください。

プラグイン使用時のオブジェクトリレーショナルデータベースの設計や作成方法については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

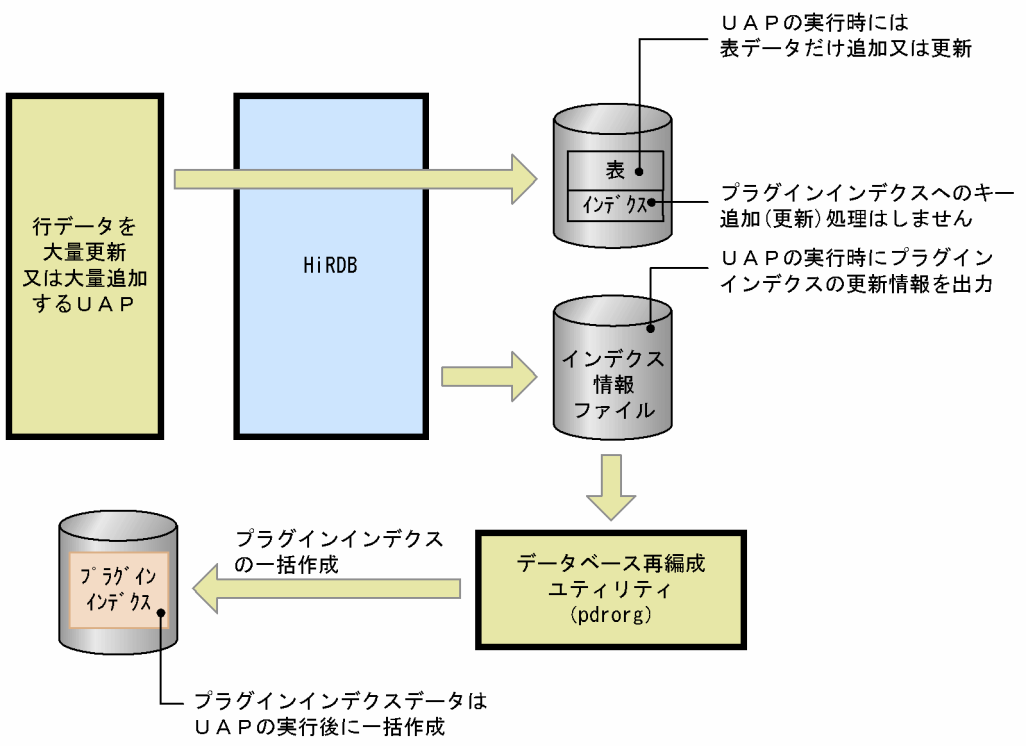
10.4.4　プラグインインデクスの遅延一括作成

プラグインインデクスを定義した表の行データを追加又は更新すると，プラグインインデクスへのキー追加処理が実行されます。このため，大量の行データを追加したり，大量の更新をしたりすると，プラグインインデクスへのキー追加処理によって性能が悪くなることがあります。

行データを追加したときに，プラグインインデクスのデータ追加処理をしないで，データベース再編成ユティリティ（pdrorg）を使用して後で一括してプラグインインデクスのデータ追加処理ができます。これを**プラグインインデクスの遅延一括作成**といいます。プラグインインデクスの遅延一括作成を次の図に示します。

なお，系切り替え機能を使用する場合は，インデクス情報ファイルを共有ディスク上に作成してください。また，UNIX 版で系切り替え機能を使用する場合は，インデクス情報ファイルを共有ディスク上の HiRDB ファイルシステム領域（キャラクタ型スペシャルファイル）に作成してください。

図 10–2　プラグインインデクスの遅延一括作成



前提条件

使用しているプラグインが，プラグインインデクスの遅延一括作成をサポートしているかどうかを確認してください。この機能をサポートしていないプラグインに対しては，この機能を使用できません。

なお，HiRDB Text Search Plug-in は，プラグインインデクスの遅延一括作成をサポートしています。

利点

プラグインインデクスのデータ追加処理をしない分だけ，大量の行データを追加又は更新する UAP の実行時間（表データの作成時間）を短縮できます。

運用方法

プラグインインデクスの遅延一括作成の運用方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

付録

付録 A　今回のサポート項目一覧

今回のサポート項目（強化ポイント）について説明します。表中に参照マニュアルを記載してありますが，マニュアル名を次のように省略して表記しています。

導入：マニュアル「HiRDB システム導入・設計ガイド」

定義：マニュアル「HiRDB システム定義」

運用：マニュアル「HiRDB システム運用ガイド」

コマンド：マニュアル「HiRDB コマンドリファレンス」

UAP：マニュアル「HiRDB UAP 開発ガイド」

付録 A.1　10-12

(1)　性能向上

性能向上に関連するサポート項目を次の表に示します。

表 A–1　性能向上に関連するサポート項目

機能項目名	特長，メリット	参照箇所
デファードライト処理の一括出力機能	デファードライト処理の並列 WRITE 機能で，連続ページの一括出力を促進し，書き込み発行回数を削減する機能をサポートしました。 これによって，デファードライト処理の書き込み性能（スループット）の向上が期待できます。	定義 運用

(2)　開発/移行容易性の向上

開発/移行容易性の向上に関連するサポート項目を次の表に示します。

表 A–2　開発/移行容易性の向上に関連するサポート項目

機能項目名	特長，メリット	参照箇所
表・ストアドプロシジャ定義の移行容易化サポート	ディクショナリ搬出入ユティリティ（pdexp），及び pddefrev（定義系 SQL の生成）コマンドを使用して，次の定義を移行できるようにしました。 • プラグインが提供する抽象データ型を含む表定義（CREATE TABLE） • インデクス型を指定したインデクス定義（CREATE INDEX） • 手続きのパラメタのデータ型として抽象データ型を指定している手続き（CREATE PROCEDURE）	導入 運用 コマンド

機能項目名	特長，メリット	参照箇所
	• 手続き定義の中からほかの手続き，関数を呼び出している手続き（CREATE PROCEDURE） また，エクスポート・インポートによる表定義情報と表データの移行で，次の定義を移行できるようにしました。 • プラグインが提供する抽象データ型を含む表定義 • インデクス型を指定したインデクス定義	

(3)　運用性向上

運用性向上に関連するサポート項目を次の表に示します。

表 A–3　運用性向上に関連するサポート項目

機能項目名	特長，メリット	参照箇所
リードレプリカの運用容易化【Linux 版限定】	リードレプリカ機能の自動反映方式で，参照 DB への反映状況を確認するコマンドを追加しました。 これによって，従来は複数の情報を突き合わせて確認していた反映状況を容易に把握できるようになり，運用負荷の軽減につながります。	運用 コマンド
HiRDB ユニット間の KeepAlive 機能サポート【UNIX 版限定】	サーバのユニット間通信で，定期的に KeepAlive パケットを送信する機能をサポートしました。 これによって，ルータやファイアウォールなどの無通信時間監視機能で，HiRDB のユニット間接続が意図しないで切断されることを防止できます。	定義
複数 IP アドレス存在時の送信 IP 指定機能サポート	クライアントマシンが複数の IP アドレスを持つ環境で，接続時の送信元 IP アドレスを明示的に指定する機能をサポートしました。 これによって，IP アドレスによる接続制限機能や，NAT やファイアウォールなど，送信元 IP アドレスを認識する機能を制御できます。	運用 UAP

(4)　トラブルシュート

トラブルシュートに関連するサポート項目を次の表に示します。

表 A–4　トラブルシュートに関連するサポート項目

機能項目名	特長，メリット	参照箇所
ODBC エラーログの特定エラーの出力抑止オプションのサポート【Linux 版，Windows 版限定】	特定のエラーについて，ODBC エラーログへの出力を抑止するオプションをサポートしました。 これによって，運用上問題のないエラーによる出力を削減できるため，対処が必要なエラーを識別しやすくなり，運用監視及び障害調査時のログ解析の効率向上が期待できます。	UAP

付録 B　プラットフォームごとの HiRDB の機能差

プラットフォームごとの HiRDB の機能差を次の表に示します。稼働環境によっては機能を使用できないこともあります。詳細は各機能の説明で確認してください。

表 B–1　プラットフォームごとの HiRDB の機能差

項目	機能，オペランド，又はコマンド名	AIX	Linux	Windows
機能	サーバプロセスのメモリサイズ監視機能	○	○	×
	ユティリティ専用ユニット	○	○	×
	HiRDB システム定義の共用化	○	○	×
	アンロード統計ログファイルを特定のサーバマシンに作成するシェルスクリプト（pdstjacm）	○	○	×
	システムログの並列出力機能	○	○	×
	HiRDB 稼働中の拡張ユニットの開始及び終了	×	○	×
	論理セクタ長 4K バイトのハードディスク上への HiRDB ファイルシステム領域の作成	○	○	○
	NVMe over Fabrics を用いた複製ディスクの高速系切り替え機能	×	○	×
	ssh/scp コマンドを用いたコマンドのリモート実行・ファイル転送	○	○	×
	インストール時の共用メモリの割り当て先選択機能	×	×	○
	CLUSTERPRO を使用した系切り替え機能	×	○	○
	CLUSTERPRO を使用したサーバモード系切り替え機能	×	×	○
	DataKeeper を使用した複製ディスクを使用した系切り替え機能	×	×	○
	パブリッククラウド上の共有ディスクアクセス機能	×	○	○
	HA モニタと Azure 共有ディスクを用いた系切り替え機能	×	○	×
	OCI（Windows）での CLUSTERPRO サポート	×	×	○
	リードレプリカ機能	×	○	×
	Tableau 対応	×	×	○
	Power BI 対応	×	×	○
	ODBC トラブルシュート強化	×	○	○
	ODBC 関数の SQLPrimaryKeys，SQLForeignKeys サポート	×	○	○
	ODBC エラーログの特定エラーの出力抑止	×	○	○

項目	機能，オペランド，又はコマンド名	AIX	Linux	Windows
	SELinux セキュリティコンテキストの設定及び解除	×	○	×
	pdcpp コマンドでの gcc コンパイラの-finput-charset オプション対応	×	○	×
	HiRDB ユニット間の KeepAlive 機能	○	○	×
付加 PP 関連	インナレプリカ機能（HiRDB Staticizer Option）	○	○	×
	更新可能なオンライン再編成（HiRDB Staticizer Option）	○	○	×
	インナレプリカグループ内の RD エリアの整合性チェックコマンドでの RD エリア構成情報の CSV 形式出力機能	○	○	×
関連製品との連携	リアルタイム SAN レプリケーション	○	○	×
	OLTP（TPBroker for C++）との連携	×	×	○
	OLTP（TUXEDO）との連携	×	×	○
	OLTP（TP1/EE）との連携	○	○	×
	EasyMT，MTguide	○	×	×
	拡張 SYSLOG 機能の適用	×	○	×
HiRDB システム定義	HiRDB 再開始処理時の連続異常終了回数の上限指定 pd_term_watch_count	○	○	×
	HiRDB のプロセスがアクセスするファイル及びパイプの最大数の指定 pd_max_open_fds	○	○	×
	1 サーバプロセスが使用するメモリサイズの上限値指定 pd_svr_castoff_size	○	○	×
	サーバのユニット内通信（TCP UNIX ドメイン）で使用する送受信バッファの最大値の指定 pd_ipc_unix_bufsize	○	○	×
	HiRDB クライアントとの通信（TCP UNIX ドメイン）で使用する送受信バッファの最大値の指定 pd_tcp_unix_bufsize	○	○	×
	リードレプリカ機能に関する指定値 pd_read_replica_use pd_read_replica_workdir	×	○	×
コマンド	pddbchg	○	○	×
	pddivinfgt	×	○	×
	pdgeter	○	○	×
	pditvstop	○	○	×
	pditvtrc	○	○	×

項目	機能，オペランド，又はコマンド名	AIX	Linux	Windows
	pdmemsv	○	○	×
	pdopsetup	○	○	×
	pdorbegin	○	○	×
	pdorcheck	○	○	×
	pdorchg	○	○	×
	pdorcreate	○	○	×
	pdorend	○	○	×
	pdplgset	○	○	×
	pdrpause	○	○	×
	pdrrpref	×	○	×
	pdseset	×	○	×
	pdsetup	○	○	×
	pdkill	×	×	○
	pdntenv	×	×	○
	pdcancel コマンドでのコアファイルの取得	○	○	×
	pdload コマンドの EasyMT の日付チェック	○	×	×
	pdcopy，pdload，pdrstr，pdrorg コマンドでの固定長テープの使用	○	×	○
	pdosinfoget	○	○	×
クライアント	ODBC ドライバ	×	○	○
	OLE DB プロバイダ	×	×	○
	HiRDB データプロバイダ for .NET Framework	×	×	○
	XDM/RD E2 接続機能の XA インタフェース	○	○	○
	複数接続機能（カーネルスレッド）	○	○	○

（凡例）

○：サポート済み

×：未サポート

付録 C　データディクショナリ表

データディクショナリ表とは，表やインデクスなどの定義情報が格納されている表のことをいいます。このデータディクショナリ表は，HiRDB が作成して管理します。データディクショナリ表は，操作系 SQL で参照し，表やインデクスなどの定義情報を確認するために使用します。

HiRDB のデータディクショナリ表の一覧を次の表に示します。データディクショナリ表を検索するときの SQL 記述例及びデータディクショナリ表の列の詳細については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

表 C–1　ディクショナリ表の一覧

項番	表名	内容	情報量 （1 行当たり）
1	SQL_PHYSICAL_FILES	HiRDB ファイルの情報(HiRDB ファイルシステム名，RD エリア名との対応関係)	1HiRDB ファイル分
2	SQL_RDAREAS	RD エリア名称，定義情報，RD エリア種別，格納表数，インデクス数などの情報	1RD エリア分
3	SQL_TABLES	データベース中の各表(ディクショナリ表を含む)の所有者名，表名	1 表分
4	SQL_COLUMNS	列に関する列名，データ型などの定義情報	1 列分
5	SQL_INDEXES	データベース中の各インデクス(ディクショナリ表を含む)の所有者名，インデクス名	1 インデクス分
6	SQL_USERS	ユーザの実行権限，及びデータベースに対するアクセスを許可したユーザの認可識別子	1 ユーザ分
7	SQL_RDAREA_PRIVILEGES	RD エリア利用権限の許可状況	1 認可識別子の 1RD エリア分
8	SQL_TABLE_PRIVILEGES	表に対するアクセス権限の付与状況	1 認可識別子の 1 表分
9	SQL_VIEW_TABLE_USAGE	ビュー表の基の実表名	1 ビュー表分
10	SQL_VIEWS	ビュー定義情報	1 ビュー表分
11	SQL_DIV_TABLE	表の分割情報(CREATE TABLE 時に指定した分割条件，及び格納 RD エリア名)	n 行で 1 表分
12	SQL_INDEX_COLINF	インデクスが定義された列名	n 行で 1 インデクス分
13	SQL_DIV_INDEX	インデクスの分割情報(格納 RD エリア名)	n 行で 1 インデクス分
14	SQL_DIV_COLUMN	BLOB 型列の分割情報(CREATE TABLE 時に指定した格納 RD エリア名)	n 行で 1 列分
15	SQL_ROUTINES	ルーチン定義情報	1 行で 1 ルーチン分

項番	表名	内容	情報量 （1 行当たり）
16	SQL_ROUTINE_RESOURCES	ルーチン中の使用リソース情報	n 行で 1 ルーチン分
17	SQL_ROUTINE_PARAMS	ルーチン中のパラメタ定義情報	n 行で 1 ルーチン分
18	SQL_TABLE_STATISTICS	表の統計情報	1 表分
19	SQL_COLUMN_STATISTICS	列の統計情報	1 列分
20	SQL_INDEX_STATISTICS	インデクスの統計情報	1 インデクス分
21	SQL_DATATYPES	ユーザ定義型の情報	1 ユーザ定義型分
22	SQL_DATATYPE_DESCRIPTORS	ユーザ定義型の構成属性の情報	1 属性分
23	SQL_TABLE_RESOURCES	表で使用するリソース情報	1 リソース分
24	SQL_PLUGINS	プラグイン情報	1 プラグイン分
25	SQL_PLUGIN_ROUTINES	プラグインのルーチン情報	1 プラグインのルーチン分
26	SQL_PLUGIN_ROUTINE_PARAMS	プラグインのルーチンのパラメタ情報	1 パラメタ情報
27	SQL_INDEX_TYPES	インデクス型の情報	1 インデクス型分
28	SQL_INDEX_RESOURCES	インデクスで使用するリソース情報	1 リソース情報分
29	SQL_INDEX_DATATYPE	インデクスの対象項目情報	1 対象項目情報分 （1 段分）
30	SQL_INDEX_FUNCTION	インデクスで利用する抽象データ型関数の情報	一つの抽象データ型関数の情報分
31	SQL_TYPE_RESOURCES	ユーザ定義型で使用するリソース情報	1 リソース情報分
32	SQL_INDEX_TYPE_FUNCTION	インデクス型を定義したインデクスで利用できる抽象データ型関数の情報	n 行で 1 インデクス型分
33	SQL_EXCEPT	インデクスの除外キー値の情報	n 行で 1 インデクスの除外キー群
34	SQL_IOS_GENERATIONS	【UNIX 版の場合】 インナレプリカ機能使用時の HiRDB ファイルシステム領域の世代情報 【Windows 版の場合】 システムが使用する情報（内容は空となります）	【UNIX 版の場合】 1 行で 1HiRDB ファイルシステム領域分 【Windows 版の場合】 なし
35	SQL_TRIGGERS	スキーマ内にあるトリガの情報	1 行で 1 トリガ分
36	SQL_TRIGGER_COLUMNS	UPDATE トリガの契機列リスト情報	1 行で 1 契機列情報
37	SQL_TRIGGER_DEF_SOURCE	トリガ定義ソース情報	n 行で 1 トリガ定義ソース情報

項番	表名	内容	情報量 （1 行当たり）
38	SQL_TRIGGER_USAGE	トリガ動作条件中で参照している資源情報	1 行で，トリガ動作条件中で参照している資源名称一つ
39	SQL_PARTKEY	マトリクス分割表の分割キーの情報	1 行で 1 分割キー情報
40	SQL_PARTKEY_DIVISION	マトリクス分割表の分割条件値の情報	1 行で 1 分割条件値情報
41	SQL_AUDITS	監査対象の情報	1 行で 1 オブジェクト又は 1 ユーザに対する 1 イベント分の情報
42	SQL_REFERENTIAL_CONSTRAINTS	参照制約の対応状況	1 行で 1 制約分の情報
43	SQL_KEYCOLUMN_USAGE	外部キーを構成する列情報	1 行で 1 列分の情報
44	SQL_TABLE_CONSTRAINTS	スキーマ内にある整合性制約の情報	1 行で 1 整合性制約分の情報
45	SQL_CHECKS	検査制約の情報	1 行で 1 検査制約分の情報
46	SQL_CHECK_COLUMNS	検査制約で使用している列の情報	1 行で一つの検査制約で使用している 1 列分の情報
47	SQL_DIV_TYPE	キーレンジ分割とハッシュ分割を組み合わせたマトリクス分割表の分割キーの情報	1 行で 1 分割キー数分の情報
48	SQL_SYSPARAMS	連続認証失敗回数制限，及びパスワードの文字列制限の情報	1 行で 1 設定項目数分，n 行で一つの連続認証失敗許容回数分，又は一つのパスワードの文字列制限分の情報
49	SQL_INDEX_XMLINF	部分構造インデクスのインデクス構成部分構造パス情報	1 行で 1 インデクス分の情報
50	SQL_SEQUENCES	順序数生成子の情報	1 行で 1 順序数生成子分の情報
51	SQL_ACCESS_SECURITY	IP アドレスによる接続制限の情報	1 行で 1 接続制約分の情報
52	SQL_ROLES	ロールの情報	1 行で 1 ロール分
53	SQL_ROLE_PRIVILEGES	ロール利用権限の情報	1 行で 1 認可識別子に付与された 1 ロール利用権限分

付録 D　HiRDB クライアントと HiRDB サーバの接続可否

HiRDB クライアントのバージョン又はプラットフォームによって，接続できる HiRDB サーバが異なります。HiRDB クライアントと HiRDB サーバの接続可否を次の表に示します。

なお，接続できる組み合わせであっても，HiRDB サーバと HiRDB クライアントの文字コード種別によって接続できないことがあります。

表 D–1　HiRDB クライアントと HiRDB サーバの接続可否（HiRDB サーバのバージョンが Version 5.0 以降の場合）

HiRDB クライアントのバージョン（種類）		HiRDB サーバのバージョン（種類）																	
		Version 7 以降						Version 6						Version 5.0					
		E	H	S	A	L	W	E	H	S	A	L	W	E	H	S	A	L	W
Version 9 09-50 以降	H	○	○	○	○	○	○	×	×	×	×	×	×	×	×	×	×	×	×
	A	○	○	○	○	○	○	×	×	×	×	×	×	×	×	×	×	×	×
	L	○	○	○	○	○	○	×	×	×	×	×	×	×	×	×	×	×	×
	W	○	○	○	○	○	○	×	×	×	×	×	×	×	×	×	×	×	×
Version 7 以降	E	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×	×
	H	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×	×
	S	○	○	○	○	○	○	○	○	△	○	○	○	○	○	△	○	×	×
	A	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×	×
	L	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×
	W	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Version 6	E	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×	×
	H	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×	×
	S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×	×
	A	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×	×
	L	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	×
	W	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Version 5.0	E	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	H	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	S	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	A	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	L	○	○	○	○	○	×	○	○	○	○	○	×	○	○	○	○	○	×

HiRDB クライアントのバージョン（種類）		HiRDB サーバのバージョン（種類）																	
		Version 7 以降						Version 6						Version 5.0					
		E	H	S	A	L	W	E	H	S	A	L	W	E	H	S	A	L	W
	W	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Version 4.0 04-03 以降	E	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	H	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	S	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	A	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	W	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Version 4.0 04-02 以前	E	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	H	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	W	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
バージョン 03-03 以前	E	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	H	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	S	○	○	○	○	×	×	○	○	○	○	×	×	○	○	○	○	×	×
	W	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
VOS3 クライアント	V	○※	○※	○※	○※	○※	○※	○	○	○	○	○	○	○	○	×	×	×	×

（凡例）

Ｅ：HI-UX/WE2 版

Ｈ：HP-UX 版

Ｓ：Solaris 版

Ａ：AIX 版

Ｌ：Linux 版

Ｗ：Windows 版

Ｖ：VOS3 版

○：接続できます。

△：HiRDB クライアントのバージョンによって次のようになります。

- バージョン 07-00-/C より前の場合
  接続できます。
- バージョン 07-00-/C 以降の場合
  HiRDB クライアントと HiRDB サーバが別のサーバマシンの場合は接続できます。同一サーバマシンの場合は接続できません。

×：接続できません。

注※

HiRDB サーバのバージョンが Version 9 以降の場合は接続できません。

付録 E　サポートを終了した機能の一覧

HiRDB では，次に示す機能及びプラットフォームのサポートを終了しました。これに伴い，マニュアルからこれらの機能の説明を削除しました。

サポートを終了した機能

- HiRDB External Data Access 機能
- JBuilder を利用した外部 Java ストアドルーチンの開発
- JP1/OmniBack II との連携機能
- システムジェネレータ（pdgen）
- ディレクトリサーバ連携機能
- データマイニング製品との連携（DATAFRONT）
- 特徴量検索プラグイン（HiRDB Image Search Plug-in）
- 分散データベース機能
- マルチメディアデータ連携用プラグイン（HiRDB File Link）
- メモリ DB 機能

サポートを終了したプラットフォーム

- Solaris
- HP-UX

付録 F　用語解説

HiRDB のマニュアルで使用している用語について説明します。

（数字）

1：1 スタンバイレス型系切り替え機能

待機系 HiRDB を準備するスタンバイ型系切り替え機能とは異なり，スタンバイレス型系切り替え機能では待機系 HiRDB を準備する必要がありません。障害が発生した場合は待機系 HiRDB に系を切り替えるのではなく，稼働中のほかのユニットに処理を代行させます。これをスタンバイレス型系切り替え機能といいます。

1：1 スタンバイレス型系切り替え機能では，障害が発生したユニットを 1 対 1 に切り替えて別のバックエンドサーバに処理を代行させることができます。

（英字）

ADT（Abstract Data Type）

→「**抽象データ型**」を参照してください。

aio ライブラリ

非同期にファイル入出力を行うための API 群が収められた，OS が提供するライブラリのことです。AIX の場合は Asynchronous I/O Subsystem，Linux の場合は libaio が該当します。システムログの並列出力機能では，aio ライブラリが提供する API を使用して，二重化されたシステムログファイルに対し並列に出力要求を行います。

この用語は，Windows 版では使用できないシステムログの並列出力機能に関する用語のため，Windows ユーザには関係ありません。

BLOB（Binary Large Object）

文書，画像，音声などの長大なデータのことです。

BLOB データ，BINARY データの部分的な更新・検索

BLOB データ，BINARY データの部分的な更新・検索には，次の三つの機能があります。なお，BINARY データは，定義長が 32,001 バイト以上のデータが対象となります。

- UPDATE 文の SET 句に連結演算を指定して，登録されている BLOB データ，又は BINARY データに対して新たなデータを追加します。
- スカラ関数 SUBSTR を指定して，BLOB データ又は BINARY データから，指定した部分だけを抽出します。

- UPDATE 文の SET 句に，処理対象列，及び開始位置として定数 1 を指定したスカラ関数 SUBSTR を使用することで，BLOB データ又は BINARY データの後方部分だけを削除できます。

この機能を使用すると，メモリ消費量を BLOB データ，BINARY データ追加分，又は BLOB データ，BINARY データ抽出分だけに抑えられます。

BLOB データのファイル出力機能
: 検索した BLOB データをクライアントに返却しないで，シングルサーバ又はフロントエンドサーバがあるユニットのファイルに直接出力し，そのファイル名をクライアントに返却する機能です。この機能を使用すると BLOB データ検索時のメモリ増大を防げます。

CONNECT 関連セキュリティ機能
: パスワードのセキュリティを向上する機能です。パスワードの最小許容バイト数を決めたり，単純なパスワードを禁止したりできます。また，連続認証失敗許容回数を設定できます。

C ストアドファンクション
: 処理手続きを C 言語で記述したストアドファンクションのことをいいます。

C ストアドプロシジャ
: 処理手続きを C 言語で記述したストアドプロシジャのことをいいます。

DB 同期状態
: インメモリデータ処理で，インメモリ RD エリア内のデータとインメモリデータバッファ上のデータの同期が取れている状態のことです。

DB 非同期状態
: インメモリデータ処理で，インメモリ RD エリア内のデータとインメモリデータバッファ上のデータの同期が取れていない状態のことです。

DECIMAL 型の符号正規化機能
: データを入力したときに符号付きパック形式データ（DECIMAL 型，日間隔型，時間隔型）の符号部を「DECIMAL 型の符号正規化機能」に示す規則に従って変換する機能のことです。符号部を変換することを，**符号部を正規化する**といいます。

FIX 属性
: 行長が固定の表に付ける属性のことです。

  列を追加しない，ナル値を持つ列がない，及び可変長の列がない表の場合は FIX 属性を指定してください。アクセス性能の向上が期待できます。

FIX ハッシュ分割

表を横分割する方法のことで，表を構成する列の値をハッシュ関数を使用して，均等に RD エリアに格納し，表を横分割することです。なお，表を横分割するときの条件にした特定の列を分割キーといいます。FIX ハッシュ分割では，表がどの RD エリアに分割されたかを HiRDB が認識します。このため，検索処理では，該当するデータがあると予測されるバックエンドサーバだけが対象になります。

HA モニタ

系切り替え機能を実現するためのクラスタソフトウェアの一つです。Hitachi HA Toolkit Extension の機能を包含しています。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

HiRDB.ini ファイル

HiRDB クライアントから実行するユティリティや HiRDB SQL Executer が，HiRDB サーバに接続するために必要な情報を設定しておくファイルです。このファイルは，HiRDB サーバの PC 及び HiRDB クライアントの PC の両方に必要です。

HiRDB/SD

HiRDB Structured Data Access Facility Version 9（HiRDB/SD）は，リレーショナル DB に加えて，HiRDB で構造型 DB を管理するためのオプション製品です。

詳細については，マニュアル「HiRDB 構造型データベース機能」を参照してください。

HiRDB データプロバイダ for .NET Framework

ADO.NET 対応のアプリケーションからデータベースにアクセスするために必要なデータプロバイダです。HiRDB データプロバイダ for .NET Framework は，ADO.NET 仕様に準拠しています。

HiRDB 管理者

システム管理者用のユーザ ID で OS にログインしたユーザのうち，HiRDB を操作する権利があるユーザのことです。HiRDB のコマンドを実行する権限があり，HiRDB のディレクトリ及びファイルの所有者です。

HiRDB クライアント

HiRDB/Developer's Kit，又は HiRDB/Run Time をインストールしたワークステーション又はパーソナルコンピュータのことをいいます。

HiRDB システム定義ファイル

HiRDB の稼働環境（システムの構成，制御情報及び各サーバの実行環境）を決定するための，HiRDB の定義情報を格納するためのファイルのことです。

HiRDB ファイル

HiRDB が使用するファイルのことです。HiRDB ファイルは，HiRDB ファイルシステム領域上の連続した複数のセグメントから構成されます。HiRDB ファイルには，表，インデクス，障害発生時にシステムの状態を回復させるのに必要な情報などが格納されます。

HiRDB ファイルシステム領域

HiRDB ファイルを作成する領域のことです。HiRDB ファイルシステム領域は，システムファイル用，作業表用ファイル用及び RD エリア用のように用途ごとに作成します。

IP アドレス

IP プロトコルで使われるアドレスを IP アドレスといいます。

IVSIVS（Ideographic Variation Sequence）は，字形が異なる文字（異体字）を区別するための Unicode の規格です。IVS では，基になる文字（基底文字）に対して，字形の種類を表すコード（VS（Variation Selector）：字形選択子）を付加した合成文字で，異体字を表現します。なお，実際に異体字を表示するには IVS に対応したフォントが別途必要です。

IP アドレスによる接続制限

データベースへ接続するクライアントを，IP アドレスによって許可又は拒否する機能のことです。

Java ストアドファンクション

処理手続きを Java で記述したストアドファンクションのことをいいます。

Java ストアドプロシジャ

処理手続きを Java で記述したストアドプロシジャのことをいいます。

JDBC ドライバ

JDBC 規格で定義されたインタフェースを，HiRDB 用に実装したドライバのことです。JDBC ドライバは Java ストアドプロシジャ，Java ストアドファンクションを実行するときに必要となり，HiRDB クライアントのインストール時に選択するとインストールされます。

JP1

バッチジョブ運用，システムの自動運転，帳票出力制御やファイルのバックアップなどの機能を備えた製品群の総称です。JP1 を使うと，システム運用を自動化／省力化できます。

LAN アダプタ

コンピュータと LAN を接続するためのデータ変換用のハードウェアです。

LOB データ

文書，画像，音声などの長大な可変長データのことです。LOB データは，ユーザ LOB 用 RD エリアに格納します。LOB 列構成基表を格納したユーザ用 RD エリアとは別にデータロード及びデータベースの再編成ができます。

LOB 列構成基表

LOB 列を含む表で，表から LOB データを除いた部分のことです。データロード及びデータベース再編成時には，LOB 列構成基表単位及び LOB 列単位に運用できます。

LVM

ディスクデバイスを管理する機能で，一つ以上の実デバイスの集合から，複数の仮想デバイスを作成及び管理できます。LVM を使用すると，一つの実デバイスを複数の小さな仮想デバイスに分割したり，逆に複数の実デバイスを一つの大きな仮想デバイスにまとめたりなど，可用性及び管理容易性の高いディスクデバイス管理ができます。また，性能向上と冗長性確保の機能もあります。

NetBackup 連携機能

データベース複写ユティリティ（pdcopy）又はデータベース回復ユティリティ（pdrstr）で使用するバックアップファイルを NetBackup サーバが管理する媒体上に作成する機能のことです。

OS ログインユーザの簡易認証機能

HiRDB クライアントが動作する OS にログインしているユーザ名で HiRDB サーバに簡易認証接続する機能のことです。

POSIX ライブラリ版

Java ストアドプロシジャ，又は Java ストアドファンクションを使用する場合に必要となる HiRDB の実行環境のことです。POSIX ライブラリ版を使用するには，pdsetup コマンドで-l オプションを指定してください。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

PRF トレース

HiRDB の一連の処理に対して出力される，トラブルシュート用のトレース情報のことです。PRF トレースは，性能検証，及びトラブルシュートの効率向上を目的とし，複数のマシン間，及び複数のプロセス間の処理の流れをトレースできます。

PRF トレースは，PRF トレース情報ファイルに出力されます。また，PRF トレースを取得する際には，PRF トレース取得レベルを選択できます。

PRF トレース取得レベル

PRF トレース取得レベルには，最小レベル（省略値），標準レベル，詳細レベル，保守レベル，及び抑止レベルがあります。PRF トレース取得レベルを変更することで，PRF トレース情報ファイルに出力する PRF トレース情報の出力量を調整できます。

PRF トレース取得レベルは，システム定義の pd_prf_level オペランドで指定します。また，HiRDB 稼働中に PRF トレース取得レベルを変更する場合は，pdprflevel コマンドを使用します。

RAID Manager インスタンス

RAID Manager が操作，管理する範囲を特定するための機能集合のことです。各インスタンスはインスタンス番号によって識別されています。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

RD エリア

データの格納単位の一つで，1〜16 個の HiRDB ファイルから構成されます。RD エリアには，次に示すものがあります。

- マスタディレクトリ用 RD エリア
- データディクショナリ用 RD エリア
- データディレクトリ用 RD エリア
- データディクショナリ LOB 用 RD エリア
- ユーザ用 RD エリア
- ユーザ LOB 用 RD エリア
- レジストリ用 RD エリア
- レジストリ LOB 用 RD エリア
- リスト用 RD エリア

RD エリア障害

インメモリデータ処理で，インメモリ RD エリアに障害が発生した状態です。

RD エリアのクローズ

HiRDB からの RD エリアに対するアクセスを停止する場合，RD エリアをクローズします。RD エリアを作り直す一部のユティリティを実行する場合，RD エリアをクローズしておく必要があります。

RD エリアをクローズすると，HiRDB がメモリ上に保持している RD エリアの一部の定義情報，及びデータ格納情報はメモリから破棄され，次回 RD エリアをオープンするときに再度作成されます。

例えば，pdmod での RD エリアの再初期化，削除などを行う場合，及び pdrstr でデータベースを回復する場合，事前に RD エリアをクローズする必要があります。

なお，オープン属性が SCHEDULE 以外の RD エリアをクローズするには，クローズする前に RD エリアを閉塞しておく必要があります。

RD エリアの自動増分

RD エリアが容量不足になったとき，HiRDB ファイルシステム領域内に空き領域があれば，自動的にセグメントを追加して RD エリアの容量を拡張します。これを RD エリアの自動増分といいます。

RD エリアの閉塞

UAP やユティリティからの RD エリアのアクセスを制限する場合，RD エリアを閉塞します（**コマンド閉塞**）。また，RD エリアに入出力障害などの障害が発生した場合，HiRDB が自動的に RD エリアを閉塞します（**障害閉塞**）。

閉塞には，参照・更新できるモードもあり，運用の目的に合った閉塞モードを選択できます。

例えば，pdload でのデータロードや，pdrorg での表の再編成を行う場合，これらの処理は長時間になることが多いため，事前に対象となる RD エリアを閉塞するとよいです。

SDB データベース

レコード，インデクス，親子集合など，HiRDB/SD が管理する構造型 DB のデータ群の総称です。

詳細については，マニュアル「HiRDB 構造型データベース機能」を参照してください。

SDB データベースを操作する API 又は DML

HiRDB/SD が管理する SDB データベースを操作するためのインタフェースです。TP1/FSP 経由で SDB データベースにアクセスする場合は，SDB データベースを操作する API を使用します。埋込み型 UAP から SDB データベースにアクセスする場合は，埋込み型 UAP 中に記述した DML を使用します。

詳細については，マニュアル「HiRDB 構造型データベース機能」を参照してください。

SQL オブジェクト

SQL を定義して実行すると，HiRDB でコンパイルされるオブジェクトのことです。

SQL 拡張最適化オプション

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための SQL 実行時の最適化方法を指定するオプションのことです。

次の最適化方法があります。

1. コストベース最適化モード 2 の適用

2. ハッシュジョイン，副問合せのハッシュ実行
3. 値式に対する結合条件適用機能
4. パラメタを含む XMLEXISTS 述語への部分構造インデクスの有効化

SQL 最適化オプション

データベースの状態を考慮して，最も効率的なアクセスパスを決定するための SQL 実行時の最適化方法を指定するオプションのことです。

次の最適化方法があります。

1. ネストループジョイン強制
2. 複数の SQL オブジェクト作成
3. フロータブルサーバ対象拡大（データ取り出しバックエンドサーバ）
4. ネストループジョイン優先
5. フロータブルサーバ候補数の拡大
6. OR の複数インデクス利用の優先
7. 自バックエンドサーバでのグループ化，ORDER BY，DISTINCT 集合関数処理
8. AND の複数インデクス利用の抑止
9. グループ分け高速化処理
10. フロータブルサーバ対象限定（データ取り出しバックエンドサーバ）
11. データ収集用サーバの分離機能
12. インデクス利用の抑止（テーブルスキャン強制）
13. 複数インデクス利用の強制
14. 更新 SQL の作業表作成抑止
15. 探索高速化条件の導出
16. スカラ演算を含むキー条件の適用
17. プラグイン提供関数からの一括取得機能
18. 導出表の条件繰り込み機能

SQL 最適化指定

SQL 文の検索効率を向上させるための最適化を SQL 文に指定できます。SQL 最適化指定には次に示す三つの項目があります。

- 使用インデクスの SQL 最適化指定
- 結合方式の SQL 最適化指定
- 副問合せ実行方式の SQL 最適化指定

なお，SQL 最適化指定は，SQL 最適化オプション及び SQL 拡張最適化オプションの指定よりも優先されます。

SQL 実行時間警告出力機能
: SQL の実行後に HiRDB が SQL の実行時間を調べます。その結果，SQL の実行時間が設定した時間（この時間を PDCWAITTIME に対する比率で設定します）以上であった場合，その SQL に対して次に示す警告情報を出力する機能です。

  - SQL 実行時間警告情報ファイル
  - 警告メッセージ（KFPA20009-W）

SQL ストアドファンクション
: 処理手続きを SQL で記述したストアドファンクションのことをいいます。

SQL ストアドプロシジャ
: 処理手続きを SQL で記述したストアドプロシジャのことをいいます。

SQL プリプロセサ
: 高級言語のコンパイラでコンパイルできるように，SQL 文を高級言語用の記述に変換するプログラムのことです。

（ア行）

アカウントロック期間
: CONNECT 関連セキュリティ機能で，連続認証失敗アカウントロック状態とする期間のことです。

空きセグメント
: データを格納していないセグメントです。使用中空きセグメントと未使用セグメントは空きセグメントになります。

空きページ
: データを格納していないページです。使用中空きページと未使用ページは空きページになります。

空きページ再利用モード
: HiRDB がデータを格納するとき，格納するための RD エリアの空き領域をサーチする方式です。セグメントが満杯になると，未使用セグメントを確保する前に使用中ページ内の空き領域をサーチします。また，次回サーチ開始位置を記憶し，次に空き領域をサーチするときそこからサーチを開始します。

空き領域の再利用機能

表にデータを格納するとき，ユーザが指定したセグメント数に達し，そのセグメントが満杯になるとページサーチモードを空きページ再利用モードに切り替えて，使用中セグメントの空き領域を使うようにする機能です。指定したセグメント数のすべてに空き領域がなくなると，新規ページ追加モードに切り替わり，新規に未使用セグメントを確保します。

アクセス権限

表にアクセスするために必要な権限のことで，表単位に設定します。アクセス権限には，次に示す 4 種類があります。

- SELECT 権限
- INSERT 権限
- DELETE 権限
- UPDATE 権限

値（インスタンス）

抽象データ型の具体的な値のことです。

圧縮表

圧縮列が定義されている表のことです。

圧縮列

表へのデータ格納時，データを圧縮して格納する圧縮指定（COMPRESSED オプション）が定義されている列のことです。

アンバランスインデクススプリット

通常のページスプリットとは異なり，インデクスページ中のデータを均等に 2 分割しないで，不均等に分割する方法です。昇順又は降順の中間キーを追加する場合にインデクスの格納効率が向上します。

アンロード状態のチェックを解除する運用

システムログファイルのスワップ先にできる条件を次の状態だけにします。

- 上書きできる状態
- 抽出完了状態（HiRDB Datareplicator）
- オンライン再編成上書き可能状態（HiRDB Staticizer Option）【UNIX 版限定】

アンロードの状態は，システムログファイルをスワップ先にできるかどうかの条件に関係なくなります。この運用をすると，システムログファイルのアンロード操作又は解放操作が不要になります。

アンロード統計ログファイル

統計ログファイルの内容をアンロードして作成したファイルのことです。

アンロードレスシステムログ運用

システムログのアンロードをしない運用です。データベースを回復するときのデータベース回復ユティリティの入力情報としてアンロードログを使用しないで，システムログを直接入力します。

アンロードログファイル

システムログファイルの内容（システムログ）をアンロードして作成したファイルのことです。

一意性制約

一意性制約とは，列中のデータの重複を許さない（列中のデータが常に一意である）制約のことです。

一時表

トランザクション又は SQL セッションの期間中だけ存在する実表です。一時表に定義されたインデクスを，一時インデクスといいます。

一時表用 RD エリア

一時表及び一時インデクスを格納するユーザ用 RD エリアです。

位置付け子機能

位置付け子機能は，BLOB データ又は BINARY データ検索時の，クライアント側のメモリ資源圧迫，及びデータ転送量削減を目的とするものです。

位置付け子とは，サーバ上のデータを識別する 4 バイトの値のデータであり，1 行 SELECT 文や FETCH 文の INTO 句などに位置付け子の埋込み変数を指定することで，検索結果としてデータの実体ではなく，そのデータを識別する位置付け子の値を受け取ります。また，データを識別する位置付け子の埋込み変数をほかの SQL 文中に指定すると，位置付け子が識別するデータを扱う処理ができます。

インタフェース領域

HiRDB と UAP との間で情報をやり取りするための領域のことです。インタフェース領域として，次の七つがあります。

- SQL 連絡領域
- SQL 記述領域
- 列名記述領域
- 型名記述領域
- 埋込み変数

- 標識変数
- パラメタ

インデクス

表を検索するためのキーとして列に付けた索引のことで，キーとキー値から構成されます。キーとは列の内容を示した列名のことで，キー値とは列の値のことです。

インデクスには，単一列インデクスと複数列インデクスがあります。単一列インデクスとは，表の一つの列に作成した一つのインデクスのことです。また，複数列インデクスとは，表の複数の列で作成した一つのインデクスのことです。

インデクスの再編成

インデクスのキー情報を検索してインデクス情報ファイルを作成し，その情報を基にインデクスを再配置します。これをインデクスの再編成といいます。インデクスの再編成は，インデクス単位又はインデクス格納 RD エリア単位に実行できます。

データの削除（DELETE）及び更新（UPDATE）を繰り返すと，インデクスの格納効率が悪くなり，インデクスを使用した検索をするときの性能が低下します。これを防ぐためにインデクスの再編成を実行します。

インデクスページスプリット

HiRDB のインデクスは，B-tree 構造をしています。このため，インデクスページにキーを追加しようとした場合，追加するインデクスページに空き領域がないと，インデクスページスプリットが発生します。インデクスページスプリットとは，キーを追加するインデクスページに空き領域がない場合に，HiRDB が空き領域を確保するために，このインデクスページのインデクス情報を二つに分割して，後ろの半分を新しいページに移すことです。

インナレプリカ機能

複製した RD エリア（レプリカ RD エリア）を定義，及び操作する機能です。インナレプリカ機能を使用すると，ミラーリング機能を持つディスクシステム又はソフトウェアを使用して複製したデータベースをアクセスできます。インナレプリカ機能の詳細については，マニュアル「インナレプリカ機能 HiRDB Staticizer Option」を参照してください。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

隠蔽

抽象データ型のインタフェースと実現方法を分離し，実現方法を隠して意識させないことです。

インメモリ RD エリア

インメモリデータ処理の対象となっている RD エリアのことです。この RD エリア内の全データがメモリ上に常駐します。

インメモリ化

インメモリデータ処理で，RD エリア内の全データをメモリ上に一括して読み込み，常駐させることです。

インメモリデータ処理

RD エリア内の全データをメモリ（インメモリデータバッファ）上に常駐させて，ディスク入出力回数を抑える処理方式のことです。

インメモリデータバッファ

インメモリデータ処理で使用するデータバッファのことです。

受け入れユニット

影響分散スタンバイレス型系切り替え機能のゲスト BES があるユニットのことです。

埋込み型 UAP

高級言語（C 言語又は COBOL 言語）でコーディングしたソースプログラムに，直接 SQL を記述する UAP のことです。

影響分散スタンバイレス型系切り替え機能

スタンバイレス型系切り替え機能の一つです。障害発生時に障害ユニット内のバックエンドサーバへの処理要求を，複数の稼働中ユニットに分散して実行させる機能を影響分散スタンバイレス型系切り替え機能といいます。

永続実表

一時表以外の実表のことです。マニュアルで使用している「表」は，基本的には永続実表を指します。

エイリアス IP アドレス

一つの LAN アダプタに複数の IP アドレスを割り当てることで，異なる IP アドレスで一つの LAN アダプタを共用できる機能です。

オーバロード

パラメタの数やデータ型が異なる場合，名前が同じ複数のストアドファンクションを定義できます。名前が同じストアドファンクションを「互いにオーバロードされている」といいます。

オブジェクト

言語コンパイラで機械語に翻訳されたプログラムのことです。

オブジェクトリレーショナルデータベース

リレーショナルデータモデルにオブジェクト指向の概念を取り込んで拡張したデータベースのことです。HiRDB では，マルチメディアデータなどの複雑な構造を持つデータとそのデータに対する操作を一本化してオブジェクトとして扱って，データベースで管理できます。

（カ行）

カーソル

データの検索時及び更新時に，複数行の検索結果を 1 行ずつ取り出すために，最新の取り出し位置を保持するものです。カーソルを宣言するには DECLARE CURSOR を，カーソルをオープンするには OPEN 文を，検索結果を取り出してカーソルを次の行へ進めるには FETCH 文を，カーソルをクローズするには CLOSE 文を使用します。

改竄防止機能

表のデータを誤って又は不当に更新されることを防ぐため，更新可能列以外の表データの更新を禁止する機能です。改竄防止機能が適用された表を改竄防止表といいます。改竄防止オプションを指定すると，その表に対する行の追加，検索，行削除禁止期間を経過したデータの削除，及び更新可能列の更新だけを許可します。なお，行削除禁止期間を省略した場合，表及び表データ共に削除できません。

外部キー

参照表に定義されている列，又は複数の列のことです。外部キーの値は参照する主キーの値と同じか，又はナル値を含む値であるように制約されます。

回復不要 FES

フロントエンドサーバがあるユニットで障害が発生して異常終了すると，そのフロントエンドサーバから実行していたトランザクションは未決着状態になることがあります。未決着状態のトランザクションは，データベースの排他を確保しているため，一部のデータベースに対する参照又は更新が制限されます。通常，未決着状態のトランザクションの決着処理をするためには，フロントエンドサーバの障害を取り除いて再開始する必要がありますが，異常終了したフロントエンドサーバが回復不要 FES であれば，HiRDB が自動的に未決着状態になっていたトランザクションを決着します。これによって，ほかのフロントエンドサーバやバックエンドサーバを使用して，データベースの更新を再開できます。回復不要 FES があるユニットを回復不要 FES ユニットといいます。

簡易認証ユーザ

OS ログインユーザの簡易認証機能を使用する HiRDB ユーザを簡易認証ユーザといいます。

監査権限

監査人に必要な権限です。次に示す操作をするときなどに監査権限が必要になります。

- 監査証跡表へのデータロード
- 監査証跡ファイルのスワップ
- 監査証跡表の検索，更新，及び削除

監査証跡
: HiRDB に対する操作のうち，監査対象のコマンド及び SQL の実行記録をファイルに出力します。この記録を監査証跡といいます。また，監査証跡の出力先ファイルを監査証跡ファイルといいます。

監査証跡表
: 監査人が監査を実施するときに利用する表のことです。監査証跡ファイルに保存した情報をデータロードして作成します。

監査人
: HiRDB システムの監査を実施する人のことです。監査人には監査権限が必要になります。監査人には，次に示す 2 種類があります。

  - 主監査人
    監査証跡表を管理及び操作する監査人です。監査証跡表の作成及び削除ができます。
  - 副監査人
    監査証跡表を操作する監査人です。監査証跡表の作成及び削除はできません。
    監査を実施する人に OS ログインアカウントを与える必要はありません。

キーレンジ分割
: 表を横分割する方法のことで，表を構成する列のうち，特定の列が持つ値の範囲を条件として表を横分割することです。なお，表を横分割するときの条件にした特定の列を分割キーといいます。キーレンジ分割には，格納条件指定と境界値指定の 2 種類の分割方法があります。

既定義型
: HiRDB が提供するデータ型のことです。既定義型には，INTEGER，CHARACTER，DATE などがあります。

基本行
: すべての列の基本となるデータを格納する行です。分岐行が作成される場合，その分岐先の情報が格納されます。

機密保護機能
: 必要な権限がないとデータベースにアクセスできないようにする機能のことです。

行識別子

HiRDB に格納されている行データを一意に識別するために，システムが割り当てる情報（行のアドレス）のことです。

業務サイト

ログ同期方式で，トランザクションを受け付けるサイトのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

共用 RD エリア

すべてのバックエンドサーバから参照できるユーザ用 RD エリアのことです。共用 RD エリアを定義できるのは HiRDB/パラレルサーバだけです。

共用インデクス

共用 RD エリアに格納されたインデクスで，すべてのバックエンドサーバから参照できるインデクスのことです。HiRDB/パラレルサーバと SQL 及び UAP の互換性を保つため，HiRDB/シングルサーバでも共用インデクスを定義することはできますが，HiRDB/シングルサーバでは共用 RD エリアを定義できないため，ユーザ用 RD エリアに格納されます。

共用表

共用 RD エリアに格納された表で，すべてのバックエンドサーバから参照できる表のことです。HiRDB/パラレルサーバと SQL 及び UAP の互換性を保つため，HiRDB/シングルサーバでも共用表を定義することはできますが，HiRDB/シングルサーバでは共用 RD エリアを定義できないため，ユーザ用 RD エリアに格納されます。

切り出し列

予備列から切り出して追加した列のことです。切り出し列は，ALTER TABLE の列追加定義で指定します。

空白変換機能

表データ中に混在している全角の空白と半角の空白を統一するための機能のことです。全角空白とは，次に示すコードのことです。半角空白 2 文字とは，X'2020'のことです。

- シフト JIS 漢字コードの場合：X'8140'
- EUC 日本語漢字コードの場合：X'A1A1'【UNIX 版限定】
- EUC 中国語漢字コードの場合：X'A1A1'
- 中国語漢字コード（GB18030）の場合：X'A1A1'
- Unicode（UTF-8）の場合：X'E38080'

クライアントグループの接続枠保証機能

HiRDB に接続するクライアントをグループ化して，各クライアントグループの HiRDB への接続枠を保証する機能です。

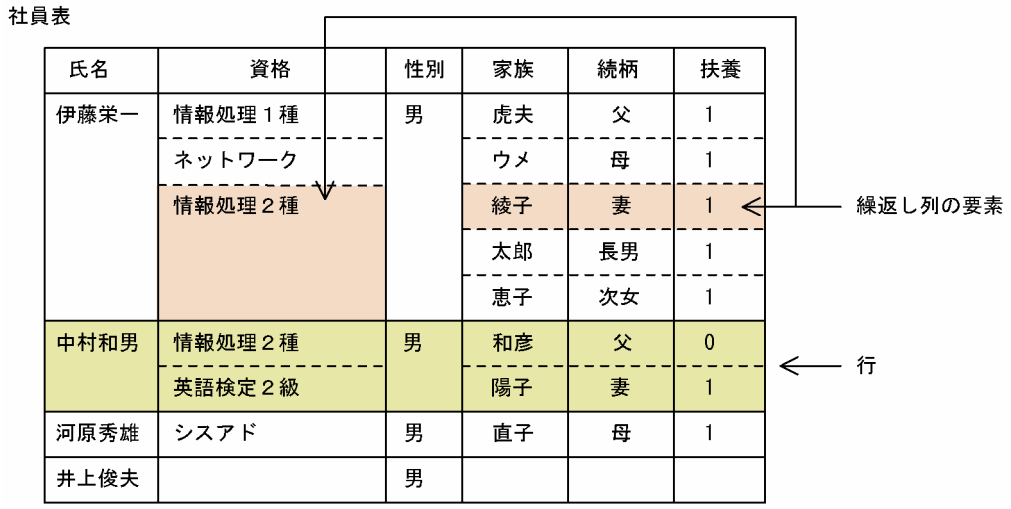
クラスタキー

特定の列の値を，昇順又は降順に行を格納するためのキーとして指定した列のことです。

繰返し列

複数の要素から構成される列のことをいいます。要素とは，繰返し列中で繰り返されている各項目のことをいいます。繰返し列は CREATE TABLE で定義し，要素数も同時に定義します。ただし，要素数は後から ALTER TABLE で増やせます。繰返し列を含む表の例を次の図に示します。

図 F–1　繰返し列を含む表の例



注　空白の箇所は，ナル値を表します。

クリティカル状態のプロセス

pdls -d rpc コマンドの実行結果の m（クリティカル状態）が Y となっているプロセスのことを，クリティカル状態のプロセスといいます。

グループ分け高速化機能

SQL の GROUP BY 句を指定してグループ分け処理をする場合，ソートしてからグループ分けをする処理にハッシングを組み合わせてグループ分け処理をする機能のことです。

グローバルデッドロック

HiRDB/パラレルサーバのサーバ間で発生するデッドロックのことをいいます。

グローバルバッファ

ディスク上の RD エリアに格納されているデータを入出力するためのバッファのことで，共用メモリ上に確保されます。RD エリア又はインデクスには必ずグローバルバッファを割り当てます。

また，次に示す場合に LOB 用グローバルバッファを割り当てます。

- プラグインインデクスを格納している場合
- データ量がそれほど多くなく，バッファリング効果が望める場合
- アクセス頻度の高い LOB データを格納している場合

グローバルバッファの先読み入力

指定した表やインデクスをあらかじめグローバルバッファに先読みしておく機能です。指定した表やインデクスを先読み入力しておくため，バッファヒット率が高くなります。

グローバルバッファの動的変更

HiRDB の稼働中に pdbufmod コマンドでグローバルバッファを追加，変更，又は削除することをグローバルバッファの動的変更といいます。

継承

下位の抽象データ型が上位の抽象データ型に定義された属性とルーチンを引き継ぐことです。

ゲスト BES

影響分散スタンバイレス型系切り替え機能で，障害発生時に該当ユニットに移動したバックエンドサーバのことです。また，ゲスト BES のユニットを受け入れユニットといいます。

ゲスト用領域

ゲスト BES に対応付けられるバックエンドサーバ用のリソースのことです。

言語コンパイラ

SQL プリプロセサが作成したポストソースプログラムを，機械語のプログラム（オブジェクト）に変換するプログラムのことです。

検査制約

データ挿入又は更新時に制約条件をチェックし，条件を満たさないデータの場合は操作を抑止することで表データの整合性を保つ制約のことです。

検査保留状態

参照制約及び検査制約が定義された表に対する SQL やユティリティの実行などで，表データの整合性を保証できなくなった場合，HiRDB は参照表又は検査制約表に対するデータ操作を制限します。このように，整合性を保証できないためにデータ操作を制限された状態のことです。

現在ロール

UAP を HiRDB から切り離す，又は SET SESSION AUTHORIZATION 文を実行するまでに，現在有効となっているロールのことです。

更新 DB

リードレプリカ構成のうち，更新業務を行う HiRDB システムのことです。

更新可能なオンライン再編成

データベースの再編成中に，データベースを参照及び更新できる機能のことをいいます。データベースを参照及び更新する処理は，レプリカのデータベースに対して行います。更新可能なオンライン再編成を実行する場合，HiRDB Staticizer Option を組み込んで，更に関連する HiRDB システム定義のオペランドを指定する必要があります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

更新可能バックエンドサーバ

共用 RD エリアの共用表や共用インデクスを更新できるバックエンドサーバです。

更新コピー

正ボリュームにデータの更新が発生した場合，その更新内容を副ボリュームにも反映（コピー）することを更新コピーといいます。更新コピーによって，正ボリュームと副ボリュームの整合性が保たれます。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

更新バッファ

データを更新する場合，グローバルバッファ上で更新してからデータベースに反映します。このとき，データベースに未反映のバッファを更新バッファといいます。

更新前ログ取得モード

UAP 又はユティリティを実行するときのデータベースの更新ログ取得方式の一つです。UAP 又はユティリティが RD エリアの内容を更新するときに，ロールバックに必要なデータベース更新ログだけを取得する方式のことです。

高速系切り替え機能

待機系 HiRDB のサーバプロセスをあらかじめ起動しておいて，系の切り替え時にサーバプロセスの起動処理をしません。系の切り替え時にサーバプロセスの起動処理がない分，系の切り替え時間を短縮できます。高速系切り替え機能のほかに，系の切り替え時間を短縮する機能としてユーザサーバホットスタンバイがあります。高速系切り替え機能の方がユーザサーバホット

スタンバイより系の切り替え時間を短縮できます（高速系切り替え機能はユーザサーバホットスタンバイの機能を包括しています）。

互換モード

オペランド省略時動作の一つです。次に示す指定を省略した場合に，特定のバージョンの省略値を仮定するモードです。

- HiRDB システム定義のオペランド
- ユティリティのオプション
- SQL のオペランド
- クライアント環境定義の環境変数
- プリプロセスオプション

なお，バージョン 09-04 の省略値を仮定することを 0904 互換モードといいます。

コストベースの最適化

ある表にインデクスが複数作成されている場合，HiRDB は表の検索で指定された探索条件を基にして，最もアクセスコストの少ないインデクスを選択します。コストベースの最適化とは，このように HiRDB が最適と判断したインデクスを優先して選択する処理のことです。

コマンドアクセスリスト

コマンド実行権限変更機能で管理する OS ログインユーザに対するコマンド実行権限のリストのことです。

コマンド代行ログ

pdcmdset コマンド及び pdcmdact コマンドの実行結果を記録するコマンド実行権限変更機能のログのことです。

コミット時反映処理

グローバルバッファ上で更新されたページを COMMIT 文発行時にディスクに書き込む処理のことです。

コンシステンシーグループ

S-VOL への更新順序の整合性が維持されて保証されるグループのことです。一つのコンシステンシーグループにグループ化された複数の非同期ペアボリュームで，P-VOL へのデータ書き込み順序と，S-VOL へのデータ更新順序が維持されます。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

コンストラクタ関数

抽象データ型の値を生成する関数のことです。

コンパイル

SQL プリプロセサで作成したポストソースプログラムを変換して，機械語のプログラム（オブジェクト）を生成する作業のことです。

（サ行）

サーバマシン

HiRDB サーバが動作するワークステーション，又はパーソナルコンピュータのマシン 1 台のことです。

サーバモード

系切り替え機能の運用方法にはモニタモードとサーバモードがあります。モニタモードの場合は系障害だけを監視対象とし，サーバモードの場合は系障害及びサーバ障害（HiRDB の異常終了など）を監視対象とします。また，サーバモードでは待機系 HiRDB を事前に開始しておくため，モニタモードに比べて系の切り替え時間を短縮できます。また，次に示す機能を使用する場合はサーバモードでの運用が前提条件になります。

- ユーザサーバホットスタンバイ
- 高速系切り替え機能
- スタンバイレス型系切り替え機能

最大同時接続数

HiRDB サーバ（HiRDB/パラレルサーバの場合は 1 フロントエンドサーバ）に同時に接続できる接続要求数のことで，pd_max_users オペランドで指定します。最大同時接続数を超えた場合，それ以上の接続要求は CONNECT エラーになります。

サイト状態

ログ同期方式でのサイトの状態です。初期，準備，業務，及びログ適用の四つの状態があります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

再編成時期予測機能

表，インデクスの再編成，又は RD エリアの拡張は，出力されたメッセージや pddbst コマンドの実行結果から，ユーザが再編成時期，及び再編成要否を判断する必要がありました。この場合，再編成の必要がない表を再編成したり，出力されたメッセージを見逃したため，再編成の必要がある表を再編成しなかったりする可能性がありました。

これらの運用を簡単にするために，HiRDB が再編成時期の予測を行うようにしたのが再編成時期予測機能です。

作業表用ファイル
SQL 文を実行するときに必要とする一時的な情報を格納するファイルのことです。

サブタイプ
ある抽象データ型を基にして，その型を特化した抽象データ型のことです。

差分バックアップ管理ファイル
差分バックアップ機能を使用するときに必要なファイルです。差分バックアップ管理ファイルには差分バックアップ取得時の情報が格納されていて，バックアップの取得時及びバックアップを使用したデータベースの回復時に HiRDB が使用します。

差分バックアップ機能
差分バックアップ機能とは，前回のバックアップ取得時点からの差分情報だけをバックアップとして取得する機能です。このため，バックアップの取得処理時間を短縮できます。データベースの容量が多くてデータ更新量が少ない場合に，差分バックアップ機能の使用を検討してください。

参照 DB
リードレプリカ構成のうち，参照業務だけを行う HiRDB システムのことです。

参照制約
表定義時に特定の列（外部キー）に制約を定義し，表間のデータの参照整合性を保つ制約のことです。参照制約及び外部キーを定義された表を参照表，外部キーによって，参照表から参照される表を被参照表といいます。なお，被参照表には外部キーによって参照される主キーを定義しておく必要があります。

参照専用バックエンドサーバ
共用 RD エリアの共用表や共用インデクスを参照できるバックエンドサーバです。更新はできません。

参照バッファ
データを参照する場合，データをグローバルバッファ上で参照します。データを参照するためのバッファ，及びデータベースに更新済みのバッファを参照バッファといいます。

システムファイル
障害時にシステムの状況を回復するための情報などを格納するためのファイルのことで，次に示すファイルの総称です。

- システムログファイル

- シンクポイントダンプファイル
- ステータスファイル

システム用 RD エリア
次に示す RD エリアの総称です。

- マスタディレクトリ用 RD エリア
- データディレクトリ用 RD エリア
- データディクショナリ用 RD エリア

システムログ適用化
業務サイトとログ適用サイトのデータベースを一時的に同期化することでデータベースの整合性を取り，ログ適用の実行に必要なファイルの情報を正しい状態にして，ログ適用ができる状態にすることをいいます。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

システムログファイル
データベースの更新履歴情報を格納するためのファイルです。一般的にはジャーナルファイルともいいます。データベースの更新履歴情報をシステムログといいます。システムログは，HiRDB 又は UAP が異常終了した場合，HiRDB がデータベースを回復するときに使用します。また，データベースを回復するときの入力情報にも使用します。

システムログファイルの空き容量監視機能
システムログファイルの空き率が警告値未満になった場合に警告メッセージを出力するか，又は新規トランザクションのスケジューリングを抑止して，サーバ内の全トランザクションを強制終了する機能のことです。

システムログをアンロードする運用
システムログをアンロードする運用です。データベースを回復するときのデータベース回復ユティリティの入力情報として，システムログをアンロードしたアンロードログを使用します。

実表
ビュー表を除く表のことです。実表には，永続実表と一時表があります。

自動再接続機能
サーバプロセスダウン，系切り替え，ネットワーク障害などの要因で HiRDB サーバとの接続が切断された場合に，自動的に再接続する機能のことです。自動再接続機能を使用すると，ユーザは HiRDB サーバとの接続の切断を意識しないで，UAP の実行を継続できます。

自動採番機能

採番業務を行う場合に，順序数生成子が生成する順序番号を使用して自動的に採番を行う機能のことです。

自動ログアンロード機能

システムログをアンロードする運用では，HiRDB 管理者が pdlogunld コマンドでアンロード待ち状態のシステムログファイルをアンロードする必要があります。このシステムログファイルのアンロード作業を自動化する機能を自動ログアンロード機能といいます。

絞込み検索

段階的に対象レコードを絞り込む検索のことをいいます。

絞込み検索をする場合，操作系 SQL の ASSIGN LIST 文でリストを作成します。

ある条件で作成したリストがあれば，そのリストを使用することで処理速度の向上が図れます。また，複数の条件を指定する場合は，複数のリストを組み合わせた検索もできます。

主キー（プライマリキー）

表中の行を一意（ユニーク）に識別するためのキーとして主キーがあります。主キーを定義した列には，一意性制約と非ナル値制約が適用されます。一意性制約とは，キー（列又は複数の列の組）中のデータの重複を許さない（キー中のデータが常に一意である）制約のことです。非ナル値制約とは，キー中の各列の値にナル値を許さない制約のことです。

縮退起動

開始できないユニットがあるときに，正常なユニットだけで HiRDB を開始することです。縮退起動を指定しておくと，あるユニットに障害が発生して起動できなくなっても，残りのユニットだけで HiRDB/パラレルサーバを開始できます。

順序数生成子

順序数生成子とは，データベース中でデータを呼び出すごとに一連の整数値（これを現在値といいます）を返す機能です。例えば，「データを挿入するごとに特定の列の値を自動的にカウントアップする」といった使い方をします。順序数生成子を使用すれば，ロールバックが発生しても現在値は変更されないため，トランザクションの状態とは無関係にデータを一意に識別したり，処理の順序性を確認したりするなどの運用ができます。

ジョイン

二つ以上の表の突き合わせ処理です。

使用中セグメント

表又はインデクスのデータを格納しているセグメントです。データが満杯でセグメント内にデータを追加できないセグメントを満杯セグメント，それ以外の使用中セグメントを空きありセグメントといいます。空きありセグメントのうち，データの削除でセグメント内の全ページが空

きページ（使用中空きページ又は未使用ページ）のセグメントを使用中空きセグメントといいます。

使用中ページ

表又はインデクスのデータを格納しているページです。特に，データが満杯でページ内にデータを追加できないページを満杯ページといい，データの削除でページ内にデータがなくなったページを使用中空きページといいます。

除外キー値

インデクス定義時に，余分なインデクスキーを作成しないために除外するキー値のことです。HiRDB では，すべての構成列の値がナル値で，キー値の重複が多いインデクスに対してナル値を除外キー値として指定できます。

新規ページ追加モード

HiRDB がデータを格納するとき，格納するための RD エリアの空き領域をサーチする方式です。セグメントが満杯になると，まず新規に未使用セグメントを確保します。RD エリア中に未使用セグメントがなくなると，既にその表に割り当てられたセグメント（使用中セグメント）の空き領域を先頭からサーチして空き領域にデータを格納します。

シンクポイントダンプファイル

HiRDB が異常終了した場合，システムログだけで回復処理をすると，HiRDB 開始からのすべてのシステムログが必要になって，回復に多大な時間が掛かります。そこで，HiRDB 稼働中に一定の間隔でポイントを設けて，そのポイントで回復に必要な HiRDB 管理情報を保存することで，ポイント以前のシステムログを不要にして回復時間を短縮できます。このポイントで取得する HiRDB 管理情報を格納するファイルをシンクポイントダンプファイルといいます。

シンクポイントダンプ有効化のスキップ回数監視機能

システム共通定義の pd_spd_syncpoint_skip_limit オペランドで，1 トランザクション中のシンクポイントダンプ有効化処理のスキップ回数の上限値を指定できます。

UAP の無限ループなどが発生すると，シンクポイントダンプの有効化処理が連続してできないことがあります（シンクポイントダンプの有効化処理が連続してスキップされることがあります）。このスキップ回数がユーザの指定した値以上の回数となった場合，対象トランザクションを強制的に中断してロールバックをします。

スーパタイプ

特化した抽象データ型（サブタイプ）に対する上位の抽象データ型のことです。

推奨モード

オペランド省略時動作の一つです。次に示す指定を省略した場合に，推奨値を仮定するモードです。

- HiRDB システム定義のオペランド
- ユティリティのオプション
- SQL のオペランド
- クライアント環境定義の環境変数
- プリプロセスのオプション

スキーマ
: 表，インデクス，抽象データ型（ユーザ定義型），インデクス型，ストアドプロシジャ，ストアドファンクション，トリガ，及びアクセス権限を包括した概念のことです。

スキーマ操作権限
: 他ユーザが所有しているスキーマに対して，リソース（表，インデクスなど）の定義や変更を行うために必要な権限です。

スキーマ単位の再編成
: スキーマ内のすべての表を一括して再編成します。所有する表の再編成を一括して行う場合にスキーマ単位の再編成を実行します。データベース再編成ユティリティの-t オプションで再編成対象のスキーマの認可識別子を指定します。指定形式は**-t 認可識別子.all** です。

  再編成の処理単位には，スキーマ単位の再編成のほかに次のものがあります。

  - 表単位の再編成
  - RD エリア単位の再編成

スキーマ定義権限
: スキーマを定義するために必要な権限です。

スタンバイ型系切り替え機能
: 業務処理中の HiRDB のほかに待機用の HiRDB を準備して，業務処理中のサーバマシン又は HiRDB に障害が発生した場合，待機用の HiRDB に業務処理を自動的に切り替えます。これをスタンバイ型系切り替え機能といいます。

スタンバイレス型系切り替え機能
: 待機系 HiRDB を準備するスタンバイ型系切り替え機能とは異なり，スタンバイレス型系切り替え機能では待機系 HiRDB を準備する必要がありません。障害が発生した場合は待機系 HiRDB に系を切り替えるのではなく，稼働中のほかのユニットに処理を代行させます。これをスタンバイレス型系切り替え機能といいます。

  スタンバイレス型系切り替え機能は，次のように分類できます。

  - 1：1 スタンバイレス型系切り替え機能

- 影響分散スタンバイレス型系切り替え機能

ステータスファイル

HiRDB を再開始するときに必要なシステムステータス情報を格納するファイルのことです。次に示すファイルから構成されます。

- サーバ用ステータスファイル
- ユニット用ステータスファイル

ストアドファンクション

SQL，Java，又は C 言語で記述したデータの加工などのデータ処理を関数としてデータベースに登録しておく機能です。0 個以上の入力パラメタを設定して戻り値を返せるため，SQL 中の値式として指定して呼び出せます。ストアドファンクションは，CREATE FUNCTION 又は CREATE TYPE 中の関数本体で定義できます。SQL，Java，又は C 言語で記述したデータ加工処理は，定義時にコンパイルされ，処理手順を記述した SQL オブジェクトが作成されて，定義情報とともにデータベースに格納されます。

ストアドプロシジャ

SQL，Java，又は C 言語で記述した一連のデータベースのアクセス手順を手続きとしてデータベースに登録しておく機能です。0 個以上の入力，出力又は入出力パラメタを持ち，SQL の CALL 文で呼び出せます。ストアドプロシジャは，CREATE PROCEDURE 又は CREATE TYPE 中の手続き本体で定義できます。SQL，Java，又は C 言語で記述されたデータベース操作は，定義時にコンパイルされ，アクセス手順を記述した SQL オブジェクトが作成されて，定義情報とともにデータベースに格納されます。

ストアドルーチン

ストアドプロシジャとストアドファンクションを総称して，ストアドルーチン又はルーチンといいます。

正規 BES

1：1 スタンバイレス型系切り替え機能で障害発生時に処理を代行してもらうバックエンドサーバのことです。また，正規 BES のユニットを正規 BES ユニットといいます。

正規表現

文字列の一部をパターン化して，任意の文字列，文字列の繰り返し，又は幾つかの文字を記述し，複数の文字列を一つのパターンで表現できる表現方法です。正規表現は，SIMILAR 述語のパターン文字列で指定します。

正規ユニット

影響分散スタンバイレス型系切り替え機能のホスト BES があるユニットのことです。

整合性制約

データベースのデータが正しい状態であることを保証するために必要な制約のことです。

整合性制約には，次に示す 2 種類があります。

- 非ナル値制約（指定した列の値に，ナル値を許さない制約）
- 一意性制約（指定した列の値がすべての行で一意であり，列中での重複を許さない制約）

正シンクポイントダンプファイル

ログ同期方式で必要となるシンクポイントダンプファイルです。ログ適用サイトでログ適用処理を行うときに使用します。業務サイトには，正シンクポイントダンプファイルと対になる副シンクポイントダンプファイルが必要となります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

正ステータスファイル

ログ同期方式で必要となるステータスファイルです。ログ適用サイトでログ適用処理を行うときに使用します。業務サイトには，正ステータスファイルと対になる副ステータスファイルが必要となります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

静的 SQL

UAP を作成するとき，プログラム中に SQL を記述する方法のことです。これに対し，UAP を実行するときに SQL を組み立てる方法のことを動的 SQL といいます。

セグメント

データの格納単位の一つで，連続した複数のページから構成されます。表やインデクスを格納するための割り当て単位です。一つのセグメントには，1 種類の表又は 1 種類のインデクスを格納できます。

全同期方式

リアルタイム SAN レプリケーションの処理方式の一つです。メインサイトのデータベースファイル又はシステムファイルに更新が発生した場合，リモートサイトのファイルに同期コピーを行います。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

全非同期方式

リアルタイム SAN レプリケーションの処理方式の一つです。メインサイトのデータベースファイル又はシステムファイルに更新が発生した場合，リモートサイトのファイルに非同期コピーを行います。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

（タ行）

代替 BES

1：1 スタンバイレス型系切り替え機能で障害発生時に処理を代行するバックエンドサーバのことです。また，代替 BES のユニットを代替 BES ユニットといいます。

代替可能性

下位の抽象データ型の値が上位の抽象データ型の値としてみなされることです。

多重定義

上位の抽象データ型で定義されたルーチンと同じ名前のルーチンを，下位の抽象データ型を定義するときに上書きして定義することです。

注釈

注釈には，単純注釈と囲み注釈があります。

単純注釈とは，SQL 文中で，「--」から開始し改行コードで終了する注釈のことをいいます。

囲み注釈とは，SQL 文中で，「/*」と「*/」で囲んで指定する注釈のことをいいます。

抽象データ型

複雑な構造を持つデータとその操作を独自に定義して，SQL で扱えるようにしたデータ型のことです。

抽象データ型列構成基表

抽象データ型の列を含む表のうち，抽象データ型の列を除いた部分で構成される表のことです。

通信情報ファイル

HiRDB が通信に使用する UNIX ドメインソケットファイル及びパイプファイルのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

通信情報ファイルディレクトリ

通信情報ファイルを格納するディレクトリのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

通信トレース

障害調査に使用するトラブルシュート情報の一種であり，HiRDB のプロセスが通信を行った際に，関数などのイベント情報を記録したものです。HiRDB の多くのプロセスでは，トラブルシュートのために，実行した通信制御用の関数などの情報をプロセス固有メモリに記録します。プロセスが異常終了すると，プロセス固有メモリの内容がコアファイルに出力されるため，コアファイルから通信トレースを取得できます。

ディクショナリサーバ

HiRDB/パラレルサーバの構成要素の一つで，データベースの定義情報であるデータディクショナリを一括管理するサーバのことです。

ディレードリラン

システム回復時に，ロールバックと新規トランザクションの受け付けを同時に開始する方式のことです。

データウェアハウス

メインフレーム，UNIX サーバ，PC サーバそれぞれのデータベースを連携して，異なる OS でもその差異を意識しないでアクセスできるデータベースシステムの概念のことです。エンドユーザが PC の環境からアクセスしたり，システム管理者がシステム分析ツールを使ってアクセスしたり，必要に応じたデータベース管理を提供します。HiRDB では，データウェアハウスのデータベースに活用できるようにするレプリケーション機能（HiRDB Datareplicator，HiRDB Dataextractor）を使用できます。

データ操作言語

適用業務プログラムがデータベースを操作するときの，データベース操作を規定する言語です。

データ定義言語

データベースの構成や内容を定義する言語です。

データディクショナリ

データベースのテーブル構造，列定義，インデクス定義などを含むデータベース設計仕様を格納したものです。HiRDB/シングルサーバの場合はシングルサーバが，HiRDB/パラレルサーバの場合はディクショナリサーバがデータディクショナリを一括管理します。

データディクショナリ LOB 用 RD エリア

ストアドプロシジャ又はストアドファンクションを格納する RD エリアのことです。ストアドプロシジャ又はストアドファンクションの定義ソース格納用と，オブジェクト格納用の二つがあります。

データディクショナリ用 RD エリア

定義系 SQL の解析結果を管理するディクショナリ表及びディクショナリ表のインデクスを格納するための RD エリアのことです。

データディレクトリ用 RD エリア

インデクスについての情報（列名，データ型など）を HiRDB のデータ形式で格納するための RD エリアです。

データベース引き継ぎ

サイト切り替えを行う場合に，ログ適用処理を完了させ，サイト状態を変更してから HiRDB を終了することをいいます。データベース引き継ぎは，pdrisedbto コマンドで行います。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

データロード

表にデータを格納することです。データベース作成ユティリティ（pdload）で実行します。

テープ装置アクセス機能

DAT，DLT，又は LTO 上にあるファイルをアクセスできます。次に示すファイルに対してテープ装置アクセス機能が適用されます。

- 入力データファイル（pdload コマンドの source 文に指定する入力データファイル）
- アンロードデータファイル（pdrorg コマンドの unload 文に指定するアンロードデータファイル）
- アンロードデータファイル（pdrorg コマンドの lobunld 文に指定する LOB データのアンロードデータファイル）
- バックアップファイル（pdcopy コマンド又は pdrstr コマンドの-b オプションに指定するバックアップファイル）

この用語は，UNIX 版では使用できない機能に関する用語のため，UNIX ユーザには関係ありません。

デッドロック

複数のトランザクションが複数の資源を競合して，互いに相手のトランザクションが確保している資源の解放を待っている状態です。

デファードライト処理

グローバルバッファ上で更新されたページを COMMIT 文が発行されてもディスクに書き込まないで，更新ページ数がある一定の値に達した時点でディスクに書き込む処理のことです。なお，更新ページ数がある一定の値（HiRDB が決定する値）に達した時点をデファードライトトリガといいます。

デファードライト処理の並列 WRITE 機能

デファードライトの書き込み時間を複数のプロセスで並列に処理する機能です。書き込み処理を複数のプロセスで並列実行するため，書き込み時間が短縮されます。

デフォルトコンストラクタ関数

抽象データ型と同じ名前で識別される，HiRDB で自動的に作成される関数のことです。引数はありません。

同期グループ

同期ペアボリュームだけで構成されているグループのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

同期コピー

リモートサイトにデータを更新コピーするときの処理方式の一つです。リモートサイトの更新処理が完了した後にメインサイトの更新処理を完了します（メインサイトの更新処理がリモートサイトの更新処理を待ち合わせます）。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

同期点

トランザクションを決着した時点を同期点といいます。トランザクションによる更新処理を有効にする場合の同期点処理をコミットといい，無効にする場合の同期点処理をロールバックといいます。

同期点指定の再編成

通常，表の再編成処理では全データの格納処理を完了するまでトランザクションを決着できません。このため，データベース再編成ユティリティ実行中はシンクポイントダンプを有効化できません。したがって，大量データの再編成処理中に HiRDB が異常終了すると，HiRDB の再開始処理に長い時間を必要とします。これを防ぐために，データ格納時（リロード処理時）に任意の件数で同期点を設定してトランザクションを決着できます。これを同期点指定の再編成といいます。

同期点指定の再編成をするには，データベース再編成ユティリティの option 文で同期点行数（何件データを格納したら同期点を取得するか）を指定してください。大量データを格納した表を再編成する場合，同期点指定の再編成を実施するかどうかを検討してください。

なお，データベース作成ユティリティでも同期点指定ができます。これを同期点指定のデータロードといいます。

同期点指定のデータロード

通常，データロード処理では全データの格納処理を完了するまでトランザクションを決着できません。このため，データベース作成ユティリティ実行中はシンクポイントダンプを有効化できません。したがって，大量データのロード中に HiRDB が異常終了すると，HiRDB の再開始処理に長い時間を必要とします。これを防ぐために，データロード時に任意の件数で同期点を設定してトランザクションを決着できます。これを同期点指定のデータロードといいます。

同期点指定のデータロードをするには，データベース作成ユティリティの option 文で同期点行数（何件データを格納したら同期点を取得するか）を指定してください。大量のデータを表にロードする場合，同期点指定のデータロードを実施するかどうかを検討してください。

なお，データベース再編成ユティリティでも同期点指定ができます。これを同期点指定の再編成といいます。

同期ペアボリューム

フェンスレベルに data 又は never を指定して作成したペアボリュームのことです。P-VOL へのデータ書き込みと，S-VOL へ反映を同期して行います。ペア状態の場合は，P-VOL のデータと S-VOL のデータに差が発生しません。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

統計ログファイル

HiRDB が出力する統計情報（統計ログ）を格納するファイルです。

動的 SQL

UAP を実行するときに SQL を組み立てる方法のことです。これに対し，UAP を作成するときに，プログラム中に SQL を記述する方法を静的 SQL といいます。

トランザクション

論理的な仕事の単位で，一連のデータベース操作などの集まりです。また，回復や排他制御の基本単位でもあります。

トランザクション欠損なし（データ欠損なし）

トランザクション欠損なしの場合，メインサイトでコミットしたトランザクションの更新処理をリモートサイトのデータベースに反映することを保証します。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

トリガ

トリガとは，ある表への操作（更新，挿入，削除）を契機に自動的に SQL 文が実行される動作のことです。自動的に実行させる SQL 文をトリガ SQL 文といいます。トリガを使用すると，ある表が更新されたときにその更新を契機に関連するほかの表も自動的に更新するなどの運用ができます。

（ナ行）

ナル値

値が設定されていないことを示す特殊な値のことです。

認可識別子

権限の集合を識別するための名前です。次に示す二つの名前があります。

- ユーザ識別子（ユーザを識別するための名前）
- ロール名（ロールを識別するための名前）

ノースプリットオプション

可変長文字列型の実際のデータ長が 256 バイト以上の場合でも，1 行のデータを 1 ページに格納するオプションです。通常は可変長文字列型の 256 バイト以上のデータ部は異なるページに格納されます。したがって，ノースプリットオプションを指定すると，データの格納効率が向上します。ノースプリットオプションを指定するには，CREATE TABLE 又は CREATE TYPE で NO SPLIT を指定します。

（ハ行）

排他制御

データベースの整合性を保つために HiRDB が管理する制御のことです。

ハイブリッド方式

リアルタイム SAN レプリケーションの処理方式の一つです。メインサイトのシステムファイルに更新が発生した場合，リモートサイトのファイルに同期コピーを行います。メインサイトのデータベースファイルに更新が発生した場合，リモートサイトのファイルに非同期コピーを行います。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

配列を使用した機能

配列型の変数を使用すると，一つの SQL の実行で複数回分の処理ができます。この機能を使用した場合，HiRDB クライアントと HiRDB サーバとの間の通信回数を削減できます。

配列を使用した機能は，FETCH 文，INSERT 文，UPDATE 文，及び DELETE 文で利用できます。

パスワード無効アカウントロック状態

CONNECT 関連セキュリティ機能でパスワードの文字列制限を設定した場合，制限に違反しているユーザはパスワード無効アカウントロック状態になります。パスワード無効アカウントロック状態のユーザは HiRDB に接続（CONNECT）できなくなります。

バックアップ取得モード

データベース複写ユティリティでバックアップを取得中に，バックアップ対象 RD エリアに対するほかの UAP からの参照及び更新要求を受け付けるかどうかの指定です。バックアップ取得モードはデータベース複写ユティリティの−Mオプションで指定し，次に示す三つのモードがあります。

- **参照・更新不可能モード（-M x 指定）**
  バックアップ取得中に，バックアップ対象 RD エリアを参照及び更新できません。
- **参照可能モード（-M r 指定）**
  バックアップ取得中に，バックアップ対象 RD エリアは参照だけできます。更新はできません。
- **更新可能モード（-M s 指定）**
  バックアップ取得中に，バックアップ対象 RD エリアを参照及び更新できます。

バックアップファイル

障害発生時に備えて，RD エリアを回復するときに必要な RD エリアのバックアップを格納するファイルのことです。

バックアップ閉塞

pdcopy コマンド以外（ほかの製品の機能）でバックアップを取得する場合，RD エリアをバックアップ閉塞してください。RD エリアをバックアップ閉塞すると，HiRDB の稼働中にほかの製品のバックアップ機能でバックアップを取得できます。RD エリアをバックアップ閉塞するには，pdhold コマンドで-b オプションを指定します。バックアップ閉塞には次の表に示す四つの種類があります。

表 F–1　バックアップ閉塞の種類

バックアップ閉塞の種類	説明
参照可能バックアップ閉塞	バックアップ閉塞中，バックアップ閉塞 RD エリアを参照できるが，更新は SQL エラー（−920）になります。

バックアップ閉塞の種類	説明
参照可能バックアップ閉塞（更新 WAIT モード）	バックアップ閉塞中，バックアップ閉塞 RD エリアを参照できます。更新はバックアップ閉塞が解除されるまで排他待ち状態になります。
更新可能バックアップ閉塞	バックアップ閉塞中，バックアップ取得対象 RD エリアを参照及び更新できます。更新トランザクションが実行中でも，pdhold コマンドを待ち状態にしないで，RD エリアをすぐに更新可能バックアップ閉塞状態にします。
更新可能バックアップ閉塞（WAIT モード）	バックアップ閉塞中，バックアップ取得対象 RD エリアを参照及び更新できます。更新トランザクションが実行中の場合は，更新トランザクションが終了するまで pdhold コマンドを待ち状態にします。

なお，次に示す閉塞を総称して**データベースの静止化**といいます。

- 参照可能バックアップ閉塞
- 参照可能バックアップ閉塞（更新 WAIT モード）

バックエンドサーバ
: HiRDB/パラレルサーバの構成要素の一つで，フロントエンドサーバからの実行指示に従ってデータベースのアクセス，排他制御，演算処理などをするサーバのことです。

ハッシュ分割表のリバランス機能
: ハッシュ分割表のデータ量が増加したため RD エリアを追加すると（表の横分割数を増やすと），既存の RD エリアと新規追加した RD エリアとの間でデータ量の偏りが生じます。ハッシュ分割表のリバランス機能を使用すると，表の横分割数を増やすときにデータ量の偏りを修正できます。ハッシュ分割表のリバランス機能は，FIX ハッシュ及びフレキシブルハッシュのどちらにも適用できます。

バッファ障害
: インメモリデータ処理で，インメモリデータバッファに障害が発生した状態です。

パブリックルーチン
: すべての利用者を示す PUBLIC を所有者として定義するストアドルーチンのことです。パブリックルーチンとして定義すると，他ユーザが定義したストアドルーチンを使用する場合，UAP 中からストアドルーチンを呼び出すときに，所有者の認可識別子を指定する必要がなくなります（ルーチン識別子だけ指定します）。パブリックルーチンは，CREATE PUBLIC PROCEDURE 又は CREATE PUBLIC FUNCTION で定義できます。

範囲指定の回復
: データベース回復ユティリティでデータベースを回復するときの方法の一つです。バックアップ取得時点から障害発生時点までの間の任意の同期点にデータベースを回復することをいいます。データベース回復ユティリティの-T オプションで回復する時刻を指定します。

非同期 READ 機能

プリフェッチ機能使用時に一括入力用のバッファを 2 面用意し，DB 処理が一つのバッファを使用中に DB 処理とは非同期に非同期 READ プロセスがもう一つのバッファに先読み入力をする機能のことです。DB 処理と先読み入力をオーバラップさせることで処理時間を短縮できます。

非同期グループ

非同期ペアボリュームだけで構成されているグループのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

非同期コピー

リモートサイトにデータを更新コピーするときの処理方式の一つです。リモートサイトの更新処理の完了を待たないでメインサイトの更新処理を完了します。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

非同期ペアボリューム

フェンスレベルに async を指定して作成したペアボリュームのことです。P-VOL へのデータ書き込みと S-VOL へ反映を同期して行いません。ペア状態であっても，P-VOL のデータと S-VOL のデータに差が発生することがあります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

非ナル値制約

非ナル値制約とは，列の値にナル値を許さない制約のことです。

ビュー表

実表から特定の行や列を選択して，新たに定義した仮想の表のことです。

表の再編成

表に対して，データの追加や更新を繰り返し実行すると，行の配置が乱れ，むだな空き領域が生じます。表の再編成とは，このようなむだな空き領域をなくすために，ユーザ用 RD エリア内の表データ，又はユーザ LOB 用 RD エリア内の LOB データを編成し直すことです。表を再編成するには，データベース再編成ユティリティ（pdrorg コマンド）を使用します。

フェンスレベル

ペアボリュームのコピー方法，及び更新コピー時に障害が発生した場合のペアボリュームのデータ整合性を保証するレベルのことです。フェンスレベルには data，never，async があります。各レベルの詳細については，RAID Manager のマニュアルを参照してください。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

副シンクポイントダンプファイル

ログ同期方式で必要となるシンクポイントダンプファイルです。ログ適用サイトでログ適用処理を行っているときのシンクポイントを取るためのファイルです。業務サイトには，副シンクポイントダンプファイルと対になる正シンクポイントダンプファイルが必要となります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

複数接続機能

一つの HiRDB クライアントの UAP から，同時に複数の HiRDB サーバに接続できる機能のことです。

副ステータスファイル

ログ同期方式で必要となるステータスファイルです。ログ適用サイトでログ適用処理を行っているときのシステムステータス情報を取るためのファイルです。業務サイトには，副ステータスファイルと対になる正ステータスファイルが必要となります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

プラグイン

文書，空間データなどのマルチメディアデータを定義した「抽象データ型」及び「複雑なデータを高速に操作できる機能」を提供する HiRDB のパッケージ製品のことです。

プラグインインデクス

プラグインが提供するインデクス型を定義したインデクスのことです。

プラグインインデクスの遅延一括作成

プラグインインデクスを定義した表に行データを追加したとき，プラグインインデクスのデータ追加処理をしないで，データベース再編成ユティリティ（pdrorg）を使用して，後で一括してプラグインインデクスのデータ追加処理をする機能です。

プラグインインデクスを定義した表の行データを大量追加（又は大量更新）するときにこの機能を使用します。

プリフェッチ機能

グローバルバッファ又はローカルバッファ上に複数のページを一括して入力することです。raw I/O 機能を適用した HiRDB ファイルシステム領域（UNIX 版の場合はキャラクタ型スペシャルファイル）を使用して，大量検索をする場合に入出力時間を短縮できます。特にインデクスを

使用しない検索，又はインデクスを使用して昇順検索をする表で，データ件数が多い場合に有効です。

プリプロセス

言語コンパイラでコンパイルする前の処理のことです。プリプロセスをすると，SQL 文が高級言語用の記述に変換されます。

フレキシブルハッシュ分割

表を横分割する方法のことで，表を構成する列の値をハッシュ関数を使用して，均等に RD エリアに格納し，表を横分割することです。なお，表を横分割するときの条件にした特定の列を分割キーといいます。フレキシブルハッシュ分割では，表を分割して RD エリアに格納する場合，どの RD エリアに分割されるか定まりません。このため，検索処理では，該当する表があるすべてのバックエンドサーバが対象になります。

フロータブルサーバ

HiRDB サーバ全体の処理性能を向上させるために，処理負荷が高いソートやマージ専用に設定し，表のデータへのアクセス処理に使用しないバックエンドサーバのことです。フロータブルサーバとして設定するバックエンドサーバには，表データを格納しません。また，トランザクションの内容によってデータアクセスをしないバックエンドサーバができる場合，一時的にフロータブルサーバとして使用されることもあります。フロータブルサーバは HiRDB/パラレルサーバにだけ設置できます。

プロセス

プログラムの実行単位のことです。各プロセス単位に仮想空間，及び時分割された CPU 資源が割り当てられます。複数のプロセスを並列実行する（マルチプロセス）ことで，スループットの向上が図れます。

プロセス間メモリ通信機能

HiRDB サーバと HiRDB クライアントが同一サーバマシンにある場合，プロセス間の通信にメモリを使用して HiRDB サーバと HiRDB クライアントとの通信を高速にします。これをプロセス間メモリ通信機能といいます。プロセス間メモリ通信機能を使用する場合は，クライアント環境定義の PDIPC オペランドに MEMORY を指定します。

プロセスの異常終了回数監視機能

サーバプロセスの異常終了回数が一定時間内に pd_down_watch_proc オペランドに指定した値を超えた場合，HiRDB（HiRDB/パラレルサーバの場合は該当するユニット）を異常終了させる機能です。この機能は系切り替え機能を使用する場合に使用することをお勧めします。サーバプロセスの異常終了が多発した場合，HiRDB を異常終了するため，すぐに系を切り替えられます。この機能を使用しないと，HiRDB が異常終了しないため，系が切り替わりません。

ブロック転送機能

HiRDB サーバから HiRDB クライアントにデータを転送するときに，任意の行数単位で転送する機能のことです。

フロントエンドサーバ

HiRDB/パラレルサーバの構成要素の一つで，実行された SQL からデータベースへのアクセス方法を決定し，バックエンドサーバに実行内容を指示するサーバのことです。

分割格納条件の変更

キーレンジ分割※で分割した表の分割格納条件を，ALTER TABLE で変更できます。表の分割格納条件を変更することで，古いデータを格納していた RD エリアを再利用でき，作業時間を短くできます。

注※

次に示す分割方法の場合に，表の分割格納条件を ALTER TABLE で変更できます。

・境界値指定

・格納条件指定（格納条件の比較演算子に＝だけを使用している場合）

分割キーインデクス

インデクスがある一定の条件を満たすと，そのインデクスは**分割キーインデクス**になります。条件を満たさないインデクスは**非分割キーインデクス**になります。ここでは，その条件について説明します。

この条件は，表が単一列分割か複数列分割かによって異なります。表の分割条件に一つの列だけを使用している場合を**単一列分割**といい，表の分割条件に複数の列を使用している場合を**複数列分割**といいます。

●単一列分割の場合

次に示すどちらかの条件を満たす場合，そのインデクスは分割キーインデクスになります。

〈条件〉

- 表を横分割するときに格納条件を指定した列（分割キー）に定義した単一列インデクス
- 表を横分割するときに格納条件を指定した列（分割キー）を第 1 構成列とした複数列インデクス

ZAIKO 表を例にして，インデクスが分割キーインデクスになる場合を次の図に示します。

図 F–2　分割キーインデクスになる場合（単一列分割の場合）

ZAIKO

SCODE	SNAME	COL	TANKA	ZSURYO
101L	ブラウス	青	3500	62
101M	ブラウス	白	3500	85
201M	ポロシャツ	白	3640	29

分割条件に指定した列（分割キー）

〔説明〕

```
CREATE INDEX A12 ON ZAIKO (SCODE ASC)                1
CREATE INDEX A12 ON ZAIKO (SCODE ASC,TANKA DESC)     2
CREATE INDEX A12 ON ZAIKO (TANKA DESC,SCODE ASC)     3
```

1. 分割キーである SCODE 列をインデクスとした場合，そのインデクスは**分割キーインデクス**になります。そのほかの列をインデクスとした場合，そのインデクスは**非分割キーインデクス**になります。
2. 分割キーである SCODE 列を複数列インデクスの第 1 構成列にすると，その複数列インデクスは**分割キーインデクス**になります。
3. 分割キーである SCODE 列を第 1 構成列以外に指定すると，その複数列インデクスは**非分割キーインデクス**になります。

●複数列分割の場合

次に示す条件を満たす場合，そのインデクスは分割キーインデクスになります。

〈条件〉

- 分割キーを先頭とし，分割に指定した列を先頭から同順にすべて含んで，複数の列に作成したインデクスです。

ZAIKO 表を例にして，インデクスが分割キーインデクスになる場合を次の図に示します。

図 F–3　分割キーインデクスになる場合（複数列分割の場合）

ZAIKO

SCODE	SNAME	COL	TANKA	ZSURYO
101L	ブラウス	青	3500	62
101M	ブラウス	白	3500	85
201M	ポロシャツ	白	3640	29

分割条件に指定した列（分割キー）

```
CREATE TABLE ZAIKO ～
     HASH HASH1 BY SCODE,TANKA ～
```

〔説明〕

```
CREATE INDEX A12 ON ZAIKO (SCODE ASC,TANKA DESC)                1
CREATE INDEX A12 ON ZAIKO (SCODE ASC,TANKA DESC,ZSURYO ASC)     2
CREATE INDEX A12 ON ZAIKO (TANKA DESC,SCODE ASC)                3
CREATE INDEX A12 ON ZAIKO (SCODE ASC,ZSURYO DESC,TANKA ASC)     4
```

1. すべての分割キー（SCODE 及び TANKA 列）を指定し，かつ分割キーの指定順序が表定義時と同じため，この複数列インデクスは**分割キーインデクス**になります。
2. すべての分割キー（SCODE 及び TANKA 列）を指定し，かつ分割キーの指定順序が表定義時と同じため，この複数列インデクスは**分割キーインデクス**になります。
3. すべての分割キー（SCODE 及び TANKA 列）を指定しているが，分割キーの指定順序が表定義時と異なるため，この複数列インデクスは**非分割キーインデクス**になります。
4. すべての分割キー（SCODE 及び TANKA 列）を指定しているが，分割キーの指定順序が表定義時と異なるため，この複数列インデクスは**非分割キーインデクス**になります。

分岐行

1 行のデータ長が 1 ページより大きい場合，別のページに分割して格納されます。また，ノースプリットオプションを指定していない場合にも，別のページに分割して格納されることがあります。このように，基本行とは別のページに格納された行を分岐行といいます。

ペア状態

ペア論理ボリューム，又はペア論理ボリュームグループがペア化されている状態をいいます。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ペアステータス

ペアボリュームの状態を意味しています。ペアステータスには PAIR，SMPL，PSUS などがあります。各状態の詳細については，RAID Manager のマニュアルを参照してください。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ペアボリューム

メインサイトとリモートサイトの日立ディスクアレイシステムで，対となるボリュームのことです。ペアボリュームには正ボリュームと副ボリュームがあります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ペア論理ボリューム

サーバ間でペアとなるボリュームを論理的に名前付けし，構成定義したボリュームのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ペア論理ボリュームグループ

ペア論理ボリュームをグループ化したものです。ファイルを配置したペア論理ボリュームをペア論理ボリュームグループ単位で扱います。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ページ

データの格納単位の一つで，データベースの入出力動作の最小単位です。ページの種類を次に示します。

- データページ：表の中の行を格納するページです。
- インデクスページ：インデクスのキー値を格納するページです。
- ディレクトリページ：RD エリアの状態の管理情報を格納するページです。

保護モード

リモートサイトへの同期コピーができないときの処理方式を保護モードで選択します。保護モードには data と never があり，各モードの処理方式を次に示します。

- **data**：メインサイトの更新処理（同期コピーができないファイルがあるボリュームの更新処理）を中止します。
- **never**：メインサイトの更新処理を続行します。全同期方式又はハイブリッド方式の場合に保護モードを選択してください。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ホスト BES

影響分散スタンバイレス型系切り替え機能で，該当ユニットに定義されたバックエンドサーバのことです。また，ホスト BES のユニットを正規ユニットといいます。

ポストソース

埋込み型で記述した SQL 文をプリプロセスしたときに生成されるソースプログラムのことです。

ボリューム属性

ボリュームには，正ボリューム（P-VOL），副ボリューム（S-VOL），シンプレックスボリューム（SMPL）の 3 種類があります。ボリューム属性とはこの 3 種類の属性のことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

（マ行）

マスタディレクトリ用 RD エリア

ディクショナリ表，ユーザが作成した表やインデクスなどを格納する RD エリアの情報，RD エリアの登録場所（サーバ）の情報などを管理する RD エリアのことです。

マトリクス分割

表の二つの列を分割キーとして，分割方法の指定を組み合わせて分割することです。一つ目の分割キーとなる列を**第 1 次元分割列**，二つ目の分割キーとなる列を**第 2 次元分割列**といいます。マトリクス分割は，第 1 次元分割列で境界値指定のキーレンジ分割をし，分割されたデータをさらに第 2 次元分割列で分割する方法です。マトリクス分割によって分割された表をマトリクス分割表といいます。

マルチ HiRDB

一つのサーバマシンで複数の HiRDB サーバを稼働させる形態のことです。

マルチフロントエンドサーバ

フロントエンドサーバを複数設定した構成のことです。SQL 処理の CPU 負荷が高く，一つのフロントエンドサーバで処理しきれない場合に設定します。

未使用セグメント

使用されたことがないセグメントです。このセグメントは RD エリア内のすべての表（又はインデクス）が使用できます。

未使用ページ

使用されたことがないページです。

メインサイト

リアルタイム SAN レプリケーションで，通常使用しているシステムがあるサイトのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

メッセージキュー監視機能

サーバプロセスの沈み込みを監視するための機能です。HiRDB では，サーバプロセスの割り当て処理でメッセージキューを使用しています。サーバプロセスの沈み込みが発生すると，メッセージキューからメッセージを取り出せなくなります。HiRDB では，ある一定時間（これをメッセージキュー監視時間といいます）を超えてもメッセージキューからメッセージを取り出

せない場合，警告メッセージ又はエラーメッセージ（KFPS00888-W 又は KFPS00889-E）を出力します。このメッセージが出力されると，サーバプロセスが沈み込んでいる可能性があります。

文字コード

文字コードとは，文字をコンピュータで扱うために個々の文字に割り当てられた符号のことをいいます。また，文字と符号との対応関係（文字コード体系）のことも，文字コードといいます。UNIX 版 HiRDB と Windows 版 HiRDB とで，使用できる文字コードが異なります。使用できる文字コードを次に示します。

文字コード	UNIX 版	Windows 版
sjis（シフト JIS 漢字コード）	○	○
ujis（EUC 日本語漢字コード）	○	×
chinese（EUC 中国語漢字コード）	○	○
utf-8（Unicode）	○	○
utf-8_ivs（Unicode（IVS 対応 UTF-8））	○	○
chinese-gb18030（中国語漢字コード GB18030）	○	○
lang-c（単一バイト文字コード）	○	○

（凡例）

○：使用できます。

×：使用できません。

文字集合

文字データに対する属性です。次の三つの属性を持ちます。

- 使用形式（文字を表現する使用規約）
- 文字レパートリ（その文字集合で表現できる文字の集合）
- 既定の照合順（二つの文字列データを比較する場合の規約）

HiRDB で使用できる文字集合を次に示します。

- EBCDIK
- UTF16

モジュールトレース

障害調査に使用するトラブルシュート情報の一種です。HiRDB の多くのプロセスは，実行した関数やマクロの履歴をプロセス固有メモリ中に記録します。プロセスが異常終了してコアファイルが出力されると，プロセス固有メモリの内容がコアファイルに出力されるため，コアファイルからモジュールトレースを取得できます。

モニタモード

系切り替え機能の運用方法にはモニタモードとサーバモードがあります。モニタモードの場合は系障害だけを監視対象とし，サーバモードの場合は系障害及びサーバ障害（HiRDB の異常終了など）を監視対象とします。また，サーバモードでは待機系 HiRDB を事前に開始しておくため，モニタモードに比べて系の切り替え時間を短縮できます。また，次に示す機能を使用する場合はサーバモードでの運用が前提条件になります。

- ユーザサーバホットスタンバイ
- 高速系切り替え機能
- スタンバイレス型系切り替え機能

（ヤ行）

ユーザ識別子

ユーザを識別するための名前です。データベースを操作するための権限をユーザ識別子に対して付与します。HiRDB では，HiRDB ユーザを識別する名前がユーザ識別子となり，データベース操作時には，実行ユーザのユーザ識別子に対して，操作に必要な権限があるかどうかがチェックされます。

ユーザ LOB 用 RD エリア

文書，画像，音声などの長大な可変長データを格納するための RD エリアのことです。次に示すデータをユーザ LOB 用 RD エリアに格納する必要があります。

- BLOB 型を指定した列（BLOB 列）
- 抽象データ型内の，BLOB 型を指定した属性
- プラグインインデクス

ユーザサーバホットスタンバイ

待機系 HiRDB のサーバプロセスをあらかじめ起動しておいて，系の切り替え時にサーバプロセスの起動処理をしません。系の切り替え時にサーバプロセスの起動処理がない分，系の切り替え時間を短縮できます。

ユーザサーバホットスタンバイのほかに，系の切り替え時間を短縮する機能として高速系切り替え機能があります。高速系切り替え機能の方がユーザサーバホットスタンバイより系の切り替え時間を短縮できます（高速系切り替え機能はユーザサーバホットスタンバイの機能を包括しています）。

ユーザ定義型

ユーザが任意に定義できるデータ型のことです。ユーザ定義型には，抽象データ型があります。

ユーザ用 RD エリア

ユーザが作成する表とインデクスを格納するための RD エリアのことです。

ユティリティ専用ユニット

次に示すユティリティ実行時に使用する入出力装置だけを設定するサーバマシンのことです。HiRDB/シングルサーバにだけ設定できます。

- データベース作成ユティリティ（pdload）
- データベース再編成ユティリティ（pdrorg）
- ディクショナリ搬出入ユティリティ（pdexp）
- データベース複写ユティリティ（pdcopy）
- データベース回復ユティリティ（pdrstr）

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ユニット

一つのサーバマシン内の，HiRDB の動作環境のことです。

ユニットコントローラ

ユニット内でサーバの実行制御や監視をしたり，ユニット間での通信制御をしたりするシステムのことです。

横分割

一つの表，インデクス又は LOB 列を複数のユーザ用 RD エリア又はユーザ LOB 用 RD エリアに分割して格納することです。表を横分割する場合，この表に作成するインデクスも，横分割する表に対応させて横分割できます。また，表に LOB 列が含まれる場合，横分割する表に対応して複数のユーザ LOB 用 RD エリアに分割して格納できます。表を横分割するには，定義系 SQL の CREATE TABLE で，横分割のための格納条件を指定します。また，インデクスを横分割するには，定義系 SQL の CREATE INDEX で，横分割するインデクスを格納するユーザ用 RD エリアを指定します。

予備列

将来の列追加を見込んで，FIX 表に定義する予備用の列のことをいいます。予備列を定義しておくと，FIX 表にデータを格納したまま列追加ができます。

予約語

SQL 文で使用するキーワードとして登録されている文字列を**予約語**といいます。

表や列などの名前には，予約語と同じ名前を指定できません。ただし，引用符（"）で囲んだ場合は，予約語と同じ名前を指定できます。

SQL 予約語削除機能を使用すると，予約語として登録されているキーワードを削除できます。ただし，削除すると使用できなくなる機能もあります。

（ラ行）

ラージファイル

2 ギガバイト以上の HiRDB ファイルシステム領域のことをラージファイルといいます。通常，HiRDB ファイルシステム領域長の上限は 2,047 メガバイト（約 2 ギガバイト）です。これを超える HiRDB ファイルシステム領域を作成する場合は，HiRDB ファイルシステム領域をラージファイルとして作成する必要があります。ラージファイルとして作成すると，HiRDB ファイルシステム領域長の上限が 1,048,575 メガバイトとなります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

リアルタイム SAN レプリケーション

通常使用しているシステムが地震，火災などの災害によって物理的に回復困難な状況になった場合，遠隔地に準備している予備のシステムで業務を続行する機能のことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

リードレプリカ

更新業務を行う HiRDB システム（更新 DB）と同一の内容を参照でき，参照業務だけを行う HiRDB システム（参照 DB）を提供する機能のことです。

リスト

適当な件数になるまで条件を指定して段階的にデータを絞り込んでいくような情報検索をするために，その途中段階のデータの集合を一時的に名前（リスト名）を付けて保存したもの，又は保存したデータの集合を意味します。

リスト用 RD エリア

リスト用 RD エリアには，ASSIGN LIST 文で作成するリストを格納します。絞込み検索をする場合に，リスト用 RD エリアを作成します。

リモートサイト

リアルタイム SAN レプリケーションで，遠隔地に準備している予備のシステムがあるサイトのことです。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

リモート操作ログ

HiRDB がリモートシェル，又はセキュアシェルを用いて行った操作時に，実行情報を記録したファイルのことです。

リンカ

言語コンパイラが作成したオブジェクトモジュールを編集し，実行形式のプログラム（ロードモジュール）を作成するプログラムのことです。

リンケージ

言語コンパイラが作成したオブジェクトモジュールを編集し，実行形式のプログラム（ロードモジュール）を作成することをいいます。

ループバックアドレス

127.0.0.0～127.255.255.255 の範囲の IP アドレス（例：127.0.0.1）のことです。ループバックアドレスとして使用できる IP アドレスは OS の仕様に依存します。また，HiRDB では，localhost を通常のホスト名として扱うため，システム定義などにホスト名として localhost を指定する場合は，名前解決しておく必要があります。

レジストリ LOB 用 RD エリア

レジストリ情報を管理する表（レジストリ管理表）を格納するための RD エリアです。レジストリ機能を使用する場合に必要です。ただし，プラグインの種類によっては，レジストリ機能を使用しないものがあります。登録されるデータの長さによって，レジストリ LOB 用 RD エリアに格納するかどうかをシステムが自動的に決定します。また，レジストリ管理表に情報を登録したりする，操作用のストアドプロシジャもこの RD エリアに格納します。

レジストリ機能

データ操作時にプラグインが使用するためのプラグイン固有の情報を HiRDB が保持する機能のことです。

レジストリ用 RD エリア

レジストリ情報を管理する表（レジストリ管理表）を格納するための RD エリアです。レジストリ機能を使用する場合に必要です。ただし，プラグインの種類によっては，レジストリ機能を使用しないものがあります。

レプリケーション機能

日立のメインフレームのデータベースと HiRDB のデータベース又は HiRDB のデータベースと HiRDB のデータベースの表データを連携して，データウェアハウスでデータを有効に利用できるようにする機能のことです。

連続認証失敗アカウントロック状態

不正なパスワードを使用し，ユーザ認証に連続して失敗した場合，そのユーザは連続認証失敗アカウントロック状態になります。連続認証失敗アカウントロック状態のユーザは HiRDB に接続（CONNECT）できなくなります。

ローカルバッファ

ローカルバッファとは，ディスク上の RD エリアに格納されているデータを入出力するためのバッファのことで，プロセス固有メモリ上に確保されます。

ロードモジュール

プリプロセス，コンパイル，及びリンケージまで終わり，UAP ソースが実行できるファイルの状態であることをいいます。

ロール（role）

権限の集合を識別するための名前です。表の権限をロールに付与することで，ロールを表の権限のグループとして運用できます。それぞれのロールはロール名で識別します。

ロールバック

トランザクションに何らかの要因で異常が発生したとき，該当するトランザクションによるデータベース処理を無効にすることです。

ログ取得モード

UAP 又はユティリティを実行するときのデータベースの更新ログ取得方式の一つです。UAP 又はユティリティが RD エリアの内容を更新するときに，ロールバック及びロールフォワードに必要なデータベース更新ログを取得する方式のことです。

ログ適用

ログ同期方式で，業務サイトからコピーされたシステムログを基に行う，データベースの更新処理のことです。ログ適用は，ログ適用サイトで行います。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ログ適用可能状態

業務サイトのデータベースとログ適用サイトのデータベースの整合性が取れていて，かつログ適用に必要な情報が正しくコピーできる状態のことをいいます。この状態で災害などが発生し，業務サイトが消失したとしても，データ欠損なしでログ適用サイトに業務を引き継げます。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ログ適用サイト

ログ同期方式で，業務サイトからコピーされたシステムログを基に，データベースの更新処理を行うサイトです。ログ適用サイトは，業務サイトと同様に常に稼働しておく必要があります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ログ適用不可能状態

業務サイトのデータベースとログ適用サイトのデータベースの整合性が取れていない状態，又はログ適用に必要な情報が正しくコピーできない状態のことをいいます。この状態で災害などが発生し，業務サイトが消失した場合，データ欠損が発生します。

ログ適用不可能状態のときにシステムログ適用化を実行すると，ログ適用可能状態になります。

この用語は，Windows 版では使用できない機能に関する用語のため，Windows ユーザには関係ありません。

ログレスモード

UAP 又はユティリティを実行するときのデータベースの更新ログ取得方式の一つです。UAP 又はユティリティが RD エリアの内容を更新するときに，データベース更新ログを取得しない方式のことです。

索引

記号

-M オプション〔pdcopy コマンド〕 272

数字

1：1 系切り替え構成 317
1：1 スタンバイレス型系切り替え機能 310
1：1 スタンバイレス型系切り替え機能〔用語解説〕 382
2：1 系切り替え構成 317
24 時間連続稼働 37, 229

A

ADO.NET 対応のアプリケーション 47
ADT〔用語解説〕 382
aio ライブラリ〔用語解説〕 382
ALTER TRIGGER 187
AND の複数インデクス利用の抑止 200
ASSIGN LIST 文 83, 204

B

BINARY 88
BLOB 88
BLOB データ，BINARY データの後方削除更新 251
BLOB データ，BINARY データの追加更新 250
BLOB データ，BINARY データの部分抽出 251
BLOB データ，BINARY データの部分的な更新・検索 250
BLOB データ，BINARY データの部分的な更新・検索〔用語解説〕 382
BLOB データのファイル出力機能 247
BLOB データのファイル出力機能〔用語解説〕 383
BLOB〔用語解説〕 382
BOOLEAN 88
B-tree インデクス 118
B-tree 構造 118

C

CALL 文 176
CHARACTER 88
Class ファイル 181
CLOSE 文 161
CLUSTER KEY オプション〔CREATE TABLE〕 91
CLUSTERPRO 314
CONNECT 関連セキュリティ機能 353
CONNECT 関連セキュリティ機能〔用語解説〕 383
CONNECT 権限 341
CONNECT 文 255
Cosminexus との連携 79
CREATE CONNECTION SECURITY 354
CREATE INDEX 120
create rdarea 文 294
create rdarea 文の指定例 149
CREATE SCHEMA 86
CREATE TABLE 87
CREATE TRIGGER 187
CREATE TYPE 88
CREATE VIEW 109
C ストアドファンクション 183
C ストアドファンクション〔用語解説〕 383
C ストアドプロシジャ 183
C ストアドプロシジャ〔用語解説〕 383
C ライブラリファイル 184

D

DABroker 77
DATE 88
DBA 権限 340
DBMS 32
DBPARTNER2 75
DB アクセス部品 208
DB 同期状態〔用語解説〕 383
DB 非同期状態〔用語解説〕 383
DB メンテナンス予定日 286

DECIMAL 型の符号正規化機能　305
DECIMAL 型の符号正規化機能〔用語解説〕　383
DECLARE CURSOR　161
DELETE 権限　342
DELETE 文　163
DISCONNECT 文　255
DISTINCT　166
DocumentBroker　77
DROP TRIGGER　187
DTP　62

E

EMPTY オプション〔CREATE INDEX〕　128
EXCEPT　166
EXCEPT VALUES オプション〔CREATE INDEX〕　128
EXISTS 述語　165
expand rdarea 文　294
EX モード　262

F

FETCH 文　161
FIX オプション〔CREATE TABLE〕　90
FIX 属性　90
FIX 属性〔用語解説〕　383
FIX ハッシュ分割　96
FIX ハッシュ分割〔用語解説〕　384
FLOAT　88
FOREIGN KEY　189
FROM 句の導出表のマージ適用　202

G

GROUP BY 句　166

H

HA グループ　312
HA モニタ　314
HA モニタ〔用語解説〕　384
HiRDB/Developer's Kit　33
HiRDB/Run Time　34
HiRDB/SD〔用語解説〕　384
HiRDB/シングルサーバ　33
HiRDB/シングルサーバの構成　38
HiRDB/パラレルサーバ　33
HiRDB/パラレルサーバの構成　39
HiRDB.ini ファイル〔用語解説〕　384
HiRDB Advanced High Availability　50, 100, 104, 229
HiRDB Control Manager　68
HiRDB Dataextractor　57
HiRDB Datareplicator　57
HiRDB Spatial Search Plug-in　77, 361, 364
HiRDB SQL Executer　67, 160
HiRDB SQL Tuning Advisor　69
HiRDB Staticizer Option　51
HiRDB Text Search Plug-in　77, 361, 363
HiRDB XA ライブラリ　62
HiRDB XA ライブラリが提供する機能　63
HiRDB XML Extension　77, 361, 364
HiRDB 管理者〔用語解説〕　384
HiRDB クライアント　33
HiRDB クライアントと HiRDB サーバの接続可否　378
HiRDB クライアント〔用語解説〕　384
HiRDB サーバ　33
HiRDB システム　32
HiRDB システム定義　226
HiRDB システム定義ファイルの作成方法　228
HiRDB システム定義ファイル〔用語解説〕　384
HiRDB システムの概要　32
HiRDB システムの構成　38
HiRDB 接続時のパスワード秘匿化機能　356
HiRDB データプロバイダ for .NET Framework　47
HiRDB データプロバイダ for .NET Framework〔用語解説〕　384
HiRDB で使える SQL　159
HiRDB との切り離し　255
HiRDB のアーキテクチャ　210
HiRDB の環境設定　211
HiRDB の特長　32
HiRDB の付加 PP　50

HiRDB ファイル　148, 213
HiRDB ファイルシステム領域　148, 213
HiRDB ファイルシステム領域の最大長　216
HiRDB ファイルシステム領域の作成〔作業表用ファイル〕　224
HiRDB ファイルシステム領域の種類　215
HiRDB ファイルシステム領域〔用語解説〕　385
HiRDB ファイル〔用語解説〕　385
HiRDB への接続　255
HiRDB を導入するメリット　35

I

inheritance　141
INSERT ONLY オプション　106
INSERT 権限　342
INSERT 文　163
IN SHARE MODE〔LOCK TABLE 文〕　264
INTEGER　88
INTERVAL HOUR TO SECOND　88
INTERVAL YEAR TO DAY　88
IP アドレスによる接続制限　357, 385
IP アドレス〔用語解説〕　385

J

JAR ファイル　181
Java 仮想マシン　182
Java 仮想マシンの位置づけ　179
Java ストアドファンクション　178
Java ストアドファンクション〔用語解説〕　385
Java ストアドプロシジャ　178
Java ストアドプロシジャ〔用語解説〕　385
JDBC 対応のアプリケーション　46
JDBC ドライバ　46
JDBC ドライバ〔用語解説〕　385
JP1/NETM/Audit　72
JP1/NETM/Audit 用監査ログ出力ファイル　73
JP1/NETM/Audit 用監査ログ出力ユティリティ　72
JP1/Performance Management - Agent Option for HiRDB　71
JP1/VERITAS NetBackup Agent for HiRDB License　281
JP1〔用語解説〕　385

L

LAN アダプタ〔用語解説〕　385
LARGE DECIMAL　88
LifeKeeper　314
LOB データ〔用語解説〕　386
LOB 用 RD エリア　81
LOB 用グローバルバッファ　236
LOB 列構成基表〔用語解説〕　386
LOCK PAGE〔CREATE TABLE〕　262
LOCK TABLE 文　264
LOCK 文　264
LRU 管理方式　242
LVM　278
LVM〔用語解説〕　386

M

MCHAR　88
Microsoft Failover Cluster　314
move rdarea 文　300
MSFC　314
MVARCHAR　88

N

NCHAR　88
NetBackup 連携機能　281
NetBackup 連携機能〔用語解説〕　386
NO SPLIT オプション〔CREATE TABLE 又は CREATE TYPE〕　92
NOT NULL オプション〔CREATE TABLE〕　188
NVARCHAR　88
n-gram インデクスプラグイン　362
n-gram インデクス方式　77

O

ODBC 対応のアプリケーション　44
ODBC ドライバ　44

OLE DB 対応のアプリケーション 45
OLE DB プロバイダ 45
OLTP 環境での UAP のトランザクション管理 260
OLTP 製品との連携 62
OLTP 連携の適用可否 62
OpenTP1 62
OpenTP1/Server Base Enterprise Option 62
OPEN 文 161
ORDER BY 句 166
OR の複数インデクス利用の優先 200
OS ログインユーザの簡易認証機能〔用語解説〕 386
override 142

P

PCTFREE オプション〔セグメント内の空きページ比率〕 151
PCTFREE オプション〔ページ内の未使用領域の比率〕 156
pd_assurance_table_no オペランド 154
pd_check_pending オペランド 189, 192
pd_dbbuff_lru_option オペランド 242, 243
pd_dbbuff_rate_updpage オペランド 242
pd_dbsync_point オペランド 241, 242
pd_dfw_awt_process オペランド 242
pd_indexlock_mode オペランド 129
pd_max_ard_process オペランド 241
pd_max_list_count オペランド 206
pd_max_list_users オペランド 206
pd_mode_conf オペランド 232
pd_pageaccess_mode オペランド 243
pd_rdarea_extension_file オペランド 295
pd_trn_commit_optimize オペランド 258
pdaudput コマンド 72
pdbuffer オペランド 237, 241
pdbuffer オペランドの-m オプション 240
pdbuffer オペランドの-p オプション 240
pdbuffer オペランドの-w オプション 241, 243
pdbufls コマンド 294
pdbufmod コマンド 237
pdchgconf コマンド 229
pdconfchk コマンド 229
pdconstck コマンド 193
pddbfrz コマンド 279
PDDBLOG オペランド 265
pdfmkfs コマンド 216
pdlbuffer オペランド 246
pdlbuffer オペランドの-p オプション 240
pdloginit コマンド〔システムログファイルの作成〕 218
pdloginit コマンド〔シンクポイントダンプファイルの作成〕 219
pdlogls コマンド 275
pdpgbfon 244
pdplgrgst コマンド 366
pdplgset コマンド 366
pdrbal コマンド 99
pdreclaim コマンド 290
pdstart コマンド 230
pdstop コマンド 230
pdstsinit コマンド〔サーバ用ステータスファイルの作成〕 221
pdstsinit コマンド〔ユニット用ステータスファイルの作成〕 220
POSIX ライブラリ版〔用語解説〕 386
PowerHA 314
PRF トレース取得レベル〔用語解説〕 387
PRF トレース〔用語解説〕 386
PRIMARY KEY オプション〔CREATE TABLE〕 90
PRIVATE 144
PROTECTED 144
PR モード 262
PUBLIC 145
PURGE TABLE 文 163
PU モード 263

R

RAID Manager 331
RAID Manager インスタンス〔用語解説〕 387
raw I/O 機能 214

RD エリア 81
RD エリア障害〔用語解説〕 387
RD エリア単位の再編成 284
RD エリアと表及びインデクスの関係 81
RD エリアの移動 300
RD エリアの拡張 294
RD エリアのクローズ〔用語解説〕 387
RD エリアの作成方法 84
RD エリアの自動増分 294
RD エリアの自動増分〔用語解説〕 388
RD エリアの種類 81
RD エリアの追加 294
RD エリアのバックアップ閉塞 278
RD エリアの閉塞〔用語解説〕 388
RD エリア〔用語解説〕 387
RD エリア利用権限 341
RECOVERY オペランド〔CREATE TABLE〕 266
RETURN 文 176
RM 62

S

SDB データベース〔用語解説〕 388
SDB データベースを操作する API 又は DML〔用語解説〕 388
SEGMENT REUSE オプション 154
SELECT 権限 342
SELECT 文 161
SGML 構造化文書データ 133
SGML プラグイン 362
SGML 文書の登録 363
Shared Nothing 方式 32
SMALLFLT 88
SMALLINT 88
SQL 43, 159
SQL オブジェクト 82, 177
SQL オブジェクト〔用語解説〕 388
SQL 拡張最適化オプション 199
SQL 拡張最適化オプション〔用語解説〕 388
SQL 最適化オプション 199
SQL 最適化オプション〔用語解説〕 389
SQL 最適化指定 199
SQL 最適化指定〔用語解説〕 389
SQL 実行時間警告出力機能〔用語解説〕 390
SQL ストアドファンクション 172
SQL ストアドファンクション〔用語解説〕 390
SQL ストアドプロシジャ 172
SQL ストアドプロシジャ〔用語解説〕 390
SQL セッション固有一時表 114
SQL によるデータベースアクセス 158
SQL の最適化 199
SQL プリプロセサ〔用語解説〕 390
SQL 予約語定義 226, 227
SQL を実行する方法 159
SR モード 262
substitutability 141
subtyping 139
SUPPRESS オプション〔CREATE TABLE〕 91
SU モード 262

T

TIME 88
TIMESTAMP 88
TM 62
TPBroker for C++ 62
TrueCopy 329
TUXEDO 62

U

UAP 環境定義 226, 227
UNBALANCED SPLIT オプション〔CREATE INDEX 又は CREATE TABLE〕 127
UNION 166
UNIQUE CLUSTER KEY オプション〔CREATE TABLE〕 188
UNIQUE オプション〔CREATE INDEX〕 188
Universal Replicator 329
UNTIL DISCONNECT〔LOCK 文〕 264
UPDATE 権限 342

UPDATE 文　162

V

VARCHAR　88

W

WITHOUT LOCK〔SELECT 文〕　264
WITHOUT ROLLBACK オプション〔CREATE TABLE〕　107
WRITE 指定　248

X

X/Open XA インタフェース　62
X/Open に準拠した API　260
XA インタフェースに準拠したトランザクション制御　255
XDM/RD E2 接続機能　34
XML 構造化文書データ　133
XML 文書の登録　363

あ

アカウントロック期間　355
アカウントロック期間〔用語解説〕　390
空きありセグメント　150
空きセグメント　150
空きセグメント〔用語解説〕　390
空きページ解放ユティリティ　290
空きページ再利用モード　152
空きページ再利用モード〔用語解説〕　390
空きページ〔用語解説〕　390
空き領域の再利用機能　152
空き領域の再利用機能〔用語解説〕　391
アクセス権限　342
アクセス権限〔用語解説〕　391
アクセス処理方式　235
値式に対する結合条件適用機能　202
値〔用語解説〕　391
圧縮表　111
圧縮表〔用語解説〕　391
圧縮分割サイズ　112
圧縮列　111
圧縮列〔用語解説〕　391
アンバランスインデクススプリット　126
アンバランスインデクススプリット〔用語解説〕　391
アンロード　283
アンロード状態のチェックを解除する運用〔用語解説〕　391
アンロード統計ログファイル〔用語解説〕　392
アンロード待ち状態　273
アンロードレスシステムログ運用〔用語解説〕　392
アンロードログファイル　269
アンロードログファイルの作成　273
アンロードログファイルの保管　275
アンロードログファイル〔用語解説〕　392

い

異常終了　231
一意性制約　188
一意性制約〔用語解説〕　392
一時インデクス　114
一時表　114
一時表用 RD エリア　83
一時表用 RD エリア〔用語解説〕　392
一時表〔用語解説〕　392
位置付け子機能　253
位置付け子機能〔用語解説〕　392
一相コミット　258
一相最適化　63, 258
意図共用モード　262
意図排他モード　262
インスタンス〔用語解説〕　391
インタフェース領域〔用語解説〕　392
インデクス　118
インデクスキー値無排他　128
インデクスの基本構造　118
インデクスの効果　119
インデクスの再編成　287
インデクスの再編成〔用語解説〕　393
インデクスの横分割　120

インデクスの横分割指針〔HiRDB/シングルサーバの場合〕 121
インデクスの横分割指針〔HiRDB/パラレルサーバの場合〕 122
インデクスの横分割の例〔HiRDB/シングルサーバの場合〕 121
インデクスの横分割の例〔HiRDB/パラレルサーバの場合〕 122
インデクスページ 148
インデクスページスプリット 125
インデクスページスプリット〔用語解説〕 393
インデクス用グローバルバッファ 235
インデクス〔用語解説〕 393
インデクス用ローカルバッファ 245
インデクス利用の抑止 201
インナレプリカ機能 51
インナレプリカ機能〔用語解説〕 393
隠蔽 144
隠蔽〔用語解説〕 393
隠蔽レベル 144
インメモリ RD エリア 238
インメモリ RD エリア〔用語解説〕 393
インメモリ化 238
インメモリ化〔用語解説〕 394
インメモリデータ処理 238
インメモリデータ処理〔用語解説〕 394
インメモリデータバッファ 238
インメモリデータバッファ〔用語解説〕 394

う

受け入れユニット 312
受け入れユニット〔用語解説〕 394
埋込み型 UAP 160
埋込み型 UAP〔用語解説〕 394
運用履歴表 287

え

影響分散スタンバイレス型系切り替え機能 311
影響分散スタンバイレス型系切り替え機能のシステム構成例 320
影響分散スタンバイレス型系切り替え機能〔用語解説〕 394
永続実表〔用語解説〕 394
エイリアス IP アドレス〔用語解説〕 394

お

追い付き反映処理 53
オーバロード 176
オーバロード〔用語解説〕 394
オブジェクト操作イベント 350
オブジェクト定義イベント 349
オブジェクト〔用語解説〕 394
オブジェクトリレーショナルデータベースへの拡張 133
オブジェクトリレーショナルデータベース〔用語解説〕 395

か

カーソル 161
カーソル〔用語解説〕 395
改竄防止機能〔用語解説〕 395
改竄防止表 105
開始モード 230
解析情報表 286
外部 C ストアドルーチン 183
外部 C ストアドルーチンの実行手順 183
外部 C ストアドルーチンの特長 183
外部 Java ストアドルーチン 178
外部 Java ストアドルーチンの実行手順 180
外部 Java ストアドルーチンの特長 178
外部キー 189
外部キー〔用語解説〕 395
回復〔データベース〕 269
回復不要 FES 327
回復不要 FES ユニット 327
回復不要 FES〔用語解説〕 395
格納条件指定 93
片方向代替構成 320
各国文字データ 88
カプセル化 135

可変長各国文字列　88
可変長混在文字列　88
可変長文字列　88
簡易セットアップツール　37
簡易認証ユーザ〔用語解説〕　395
監査権限　341, 343
監査権限〔用語解説〕　395
監査証跡　343
監査証跡表　347
監査証跡表〔用語解説〕　396
監査証跡ファイル　347
監査証跡〔用語解説〕　396
監査対象イベント　348
監査人　343
監査人セキュリティイベント　349
監査人〔用語解説〕　396
監査ログ〔JP1/NETM/Audit〕　72
監査ログ管理サーバ〔JP1/NETM/Audit との連携〕　73
監査ログ収集対象サーバ〔JP1/NETM/Audit との連携〕　73
関数　172
関数呼出し　176

き

キー値　118
キーレンジ分割　93
キーレンジ分割〔格納条件指定の例〕　93
キーレンジ分割〔境界値指定の例〕　94
キーレンジ分割〔用語解説〕　396
既定義型　88
既定義型〔用語解説〕　396
基本行〔用語解説〕　396
機密保護機能　340
機密保護機能〔用語解説〕　396
行　87
境界値指定　94
行削除禁止期間　106
行識別子〔用語解説〕　397
強制開始　230
強制終了　231
行単位の排他制御　262
行排他　262
業務サイト　336
業務サイト〔用語解説〕　397
共有ディスク装置　314
共用 RD エリア　83
共用 RD エリア〔用語解説〕　397
共用意図排他モード　263
共用インデクス〔用語解説〕　397
共用表　110
共用表〔用語解説〕　397
共用モード　262
切り出し列〔用語解説〕　397

く

空間検索プラグイン　77, 361, 364
空間データ　133
空白変換機能　302
空白変換機能〔用語解説〕　397
クライアントグループの接続枠保証機能〔用語解説〕　398
クラスタキー　90
クラスタキー〔用語解説〕　398
クラスタシステム　308
クラスタソフトウェア　314
繰返し列　107
繰返し列〔用語解説〕　398
クリティカル状態のプロセス〔用語解説〕　398
グループ分け高速化機能　196
グループ分け高速化機能〔用語解説〕　398
グループ分け高速化処理　201
グローバルデッドロック〔用語解説〕　398
グローバルバッファ　235
グローバルバッファ常駐化ユティリティ　244
グローバルバッファの LRU 管理方式　242
グローバルバッファの先読み入力　244
グローバルバッファの先読み入力〔用語解説〕　399
グローバルバッファの動的変更　237

グローバルバッファの動的変更〔用語解説〕 399
グローバルバッファの割り当て方法 237
グローバルバッファ〔用語解説〕 399

け

計画系切り替え 317
計画停止 231
系切り替え機能 308
系切り替え機能の形態 316
系切り替え失敗時の自動再起動機能 324
系切り替えの実行時間監視機能 325
継承 141
系障害 315
継承〔用語解説〕 399
系の切り替え時間の比較 323
系の切り戻し 311
ゲスト BES 312
ゲスト BES〔用語解説〕 399
ゲスト用領域 312
ゲスト用領域〔用語解説〕 399
結合方式の SQL 最適化指定 200
権限管理イベント 349
言語コンパイラ〔用語解説〕 399
現在ロール〔用語解説〕 400
検査制約 192
検査制約表 192
検査制約〔用語解説〕 399
検査保留状態 193
検査保留状態〔用語解説〕 399
限定述語 165
現用系 309
現用ファイル 273

こ

更新 DB 400
更新 SQL の作業表作成抑止 201
更新可能状態 280
更新可能なオンライン再編成 52
更新可能なオンライン再編成〔用語解説〕 400
更新可能バックアップ閉塞 279
更新可能バックアップ閉塞（WAIT モード） 279
更新可能バックエンドサーバ〔用語解説〕 400
更新可能モード 272
更新可能列 106
更新コピー 329
更新コピーの対象になるファイル 331
更新コピー〔用語解説〕 400
更新凍結コマンド 279
更新凍結状態 280
更新バッファ 235
更新バッファ〔用語解説〕 400
更新前ログ取得モード 265
更新前ログ取得モード〔用語解説〕 400
更新ログ取得方式 265
構造化繰返し述語 108
高速系切り替え機能 323
高速系切り替え機能〔用語解説〕 400
候補キー 90
公用 RD エリア 82, 83, 341
互換モード〔用語解説〕 401
コストベース最適化モード 2 の適用 202
コストベースの最適化 120
コストベースの最適化〔用語解説〕 401
固定小数点数 88
固定長各国文字列 88
固定長混在文字列 88
固定長文字列 88
コマンドアクセスリスト〔用語解説〕 401
コマンド代行ログ〔用語解説〕 401
コミット 256, 270
コミット時反映処理 242
コミット時反映処理〔用語解説〕 401
コミット処理 258
コミットメント制御 257
混在文字データ 88
コンシステンシーグループ〔用語解説〕 401
コンストラクタ関数 136
コンストラクタ関数〔用語解説〕 402

コンパイル〔用語解説〕 402

さ

サーバ間横分割 97, 123
サーバ共通定義 226, 227
サーバ障害 315
サーバ内横分割 97, 122
サーバマシン〔用語解説〕 402
サーバモード 315
サーバモード〔用語解説〕 402
サーバ用ステータスファイル 220
再開始 230
最小許容バイト数の設定 353
最大同時接続数〔用語解説〕 402
サイト状態〔用語解説〕 402
採番業務 107
再編成〔RD エリア単位〕 284
再編成時期の予測データの解析 287
再編成時期の予測データの取得 286
再編成時期予測機能 286
再編成時期予測機能〔用語解説〕 402
再編成〔スキーマ単位〕 285
再編成〔同期点指定〕 285
再編成〔表単位〕 284
作業表用ファイル 224
作業表用ファイル〔用語解説〕 403
サブタイプ 139
サブタイプ〔用語解説〕 403
サプレスオプション 91
差分バックアップ 277
差分バックアップ管理ファイル〔用語解説〕 403
差分バックアップ機能 276
差分バックアップ機能〔用語解説〕 403
差分バックアップグループ 277
サポート終了 381
参照 DB 403
参照可能バックアップ閉塞 279
参照可能バックアップ閉塞（更新 WAIT モード） 279
参照可能モード 272
参照制約 189
参照制約〔用語解説〕 403
参照専用バックエンドサーバ〔用語解説〕 403
参照バッファ 235
参照バッファ〔用語解説〕 403
参照表 189
参照・更新不可能モード 272

し

時間隔データ 88
時刻印データ 88
時刻データ 88
システム管理者セキュリティイベント 348
システム共通定義 226, 227
システム構成変更コマンド 229
システムファイル 217
システムファイルの構成〔HiRDB/シングルサーバの場合〕 222
システムファイルの構成〔HiRDB/パラレルサーバの場合〕 223
システムファイルの構成単位 221
システムファイル〔用語解説〕 403
システムマネジャ 41
システム用 RD エリア 81
システム用 RD エリア〔用語解説〕 404
システムログ 217
システムログ適用化〔用語解説〕 404
システムログのアンロード 273
システムログファイル 217
システムログファイルの空き容量監視機能〔用語解説〕 404
システムログファイルの自動拡張機能 218
システムログファイルのスワップ 273
システムログファイルの二重化 217
システムログファイル〔用語解説〕 404
システムログをアンロードする運用〔用語解説〕 404
実行系 309
実体化〔一時表〕 114
実表 109
実表〔用語解説〕 404

自動開始　232
自動系切り替え　317
自動再接続機能　260
自動再接続機能〔用語解説〕　404
自動採番機能　207
自動採番機能〔用語解説〕　405
自動的な排他制御　263
自動ログアンロード機能〔用語解説〕　405
自バックエンドサーバでのグループ化，ORDER BY，DISTINCT 集合関数処理　200
絞込み検索　204
絞込み検索〔用語解説〕　405
ジャーナル　217
終了モード　231
主キー　90
主キー〔用語解説〕　405
縮退起動　232
縮退起動〔用語解説〕　405
手動開始　232
順序数生成子　207
順序数生成子〔用語解説〕　405
ジョイン〔用語解説〕　405
私用 RD エリア　82, 83, 341
使用インデクスの SQL 最適化指定　199
障害発生直前の最新の同期点に回復する場合　270
障害閉塞　267
使用中空きセグメント　150
使用中空きセグメントの解放　151, 293
使用中空きセグメントの再利用　293
使用中空きページ　155
使用中空きページが作成される処理　292
使用中空きページの解放　157, 290
使用中空きページの再利用　290
使用中セグメント　150
使用中セグメント〔用語解説〕　405
使用中ページ　155
使用中ページ〔用語解説〕　406
使用中満杯ページ　155
除外キー値　128
除外キー値〔用語解説〕　406
処理性能の向上　195
新規ページ追加モード　152
新規ページ追加モード〔用語解説〕　406
シンクポイント　218
シンクポイントダンプ　218
シンクポイントダンプ処理　36
シンクポイントダンプファイル　218
シンクポイントダンプファイルの二重化　219
シンクポイントダンプファイル〔用語解説〕　406
シンクポイントダンプ有効化のスキップ回数監視機能〔用語解説〕　406
シングルサーバ　38
シングルサーバ定義　226

す

推奨モード〔用語解説〕　406
数データ　88
スーパタイプ　140
スーパタイプ〔用語解説〕　406
スカラ演算を含むキー条件の適用　201
スカラ演算を含む条件に対するサーチ条件適用　202
スキーマ　86
スキーマ操作権限　341
スキーマ操作権限〔用語解説〕　407
スキーマ単位の再編成　285
スキーマ単位の再編成〔用語解説〕　407
スキーマ定義権限　341
スキーマ定義権限〔用語解説〕　407
スキーマ〔用語解説〕　407
スタンバイ型系切り替え機能　308
スタンバイ型系切り替え機能〔用語解説〕　407
スタンバイレス型系切り替え機能　310
スタンバイレス型系切り替え機能〔用語解説〕　407
ステータスファイル　220
ステータスファイル〔用語解説〕　408
ストアドファンクション　82, 172
ストアドファンクションのオーバロード　176
ストアドファンクションの再作成　177

ストアドファンクションの適用　175
ストアドファンクションの呼び出し　176
ストアドファンクション〔用語解説〕　408
ストアドプロシジャ　82, 172
ストアドプロシジャの再作成　177
ストアドプロシジャの呼び出し　176
ストアドプロシジャ〔用語解説〕　408
ストアドルーチン　172
ストアドルーチン〔用語解説〕　408
スナップショット方式　243

せ

正規 BES　310
正規 BES ユニット　310
正規 BES〔用語解説〕　408
正規化　89
正規表現〔用語解説〕　408
正規ユニット　312
正規ユニット〔用語解説〕　408
整合性制約　188
整合性制約〔用語解説〕　409
整合性チェックユティリティ　193
正常開始　230
正常終了　231
正シンクポイントダンプファイル〔用語解説〕　409
整数　88
正ステータスファイル〔用語解説〕　409
静的 SQL〔用語解説〕　409
静的登録　66
セキュリティ監査機能　343
セグメント　148
セグメント内の空きページ比率　150
セグメントの確保と解放　151
セグメントの状態　150
セグメントの設計　150
セグメント〔用語解説〕　409
セッションセキュリティイベント　349
全同期方式　332
全同期方式〔用語解説〕　409
全非同期方式　333
全非同期方式〔用語解説〕　410
全文検索プラグイン　77, 361, 363

そ

相互系切り替え構成　318
相互代替構成　319
挿入履歴保持列　106
外結合内結合変換機能　202

た

第 1 次元分割列　100
第 2 次元分割列　100
待機系　309
代替 BES　310
代替 BES ユニット　310
代替 BES〔用語解説〕　410
代替可能性　141
代替可能性〔用語解説〕　410
代替中　311
代替部　311
大量データの再編成　285
ダイレクトディスクアクセス（raw I/O）　214
多重定義　142, 143
多重定義〔用語解説〕　410
多段系切り替え　312
単一文字種の指定禁止　354
単一列インデクス　119
単一列分割　123
探索高速化条件の導出　201
探索条件　164
単精度浮動小数点数　88

ち

中間ページ　118
注釈〔用語解説〕　410
抽象データ型　133
抽象データ型の定義　133
抽象データ型のナル値　137

抽象データ型の列の検索　170
抽象データ型の列の更新　170
抽象データ型の列の削除　170
抽象データ型〔用語解説〕　410
抽象データ型列構成基表〔用語解説〕　410
抽象データ型を含む表データの操作　167
長大データ　88

つ

通信情報ファイル　410
通信情報ファイルディレクトリ　411
通信トレース〔用語解説〕　411
通信のオーバヘッド　195

て

定義ソース　82
ディクショナリサーバ　41
ディクショナリサーバ定義　227
ディクショナリサーバ〔用語解説〕　411
ディクショナリ表　82
ディクショナリ表のインデクス　82
ディザスタリカバリ　329
ディレードリラン　233
ディレードリラン方式　36
ディレードリラン〔用語解説〕　411
ディレクトリページ　148
データ圧縮機能　111
データウェアハウス〔用語解説〕　411
データ型　88
データ型としての抽象データ型　135
データ型の種類　88
データ収集用サーバの分離機能　201
データ操作言語〔用語解説〕　411
データ定義言語〔用語解説〕　411
データディクショナリ LOB 用 RD エリア　82
データディクショナリ LOB 用 RD エリア〔用語解説〕　412
データディクショナリ表　375
データディクショナリ用 RD エリア　82
データディクショナリ用 RD エリア〔用語解説〕　412
データディクショナリ〔用語解説〕　411
データディレクトリ用 RD エリア　82
データディレクトリ用 RD エリア〔用語解説〕　412
データの演算　166
データの加工　166
データの基本操作　161
データの検索　161
データの検索〔抽象データ型〕　167
データの更新　162
データの更新〔抽象データ型〕　167
データの削除　162
データの削除〔抽象データ型〕　168
データの挿入　163
データの挿入〔抽象データ型〕　168, 171
データ反映方式〔リアルタイム SAN レプリケーション〕　332
データページ　148
データベース回復ユティリティ　269
データベース管理システム　32
データベース再編成ユティリティ　283
データベース抽出・反映サービス機能　57
データベースの回復　269
データベースの管理　268
データベースの更新ログを取得しないときの運用　265
データベースの障害に備えた運用　271
データベースの静止化　279
データベースの物理構造　147
データベースの論理構造　80
データベース引き継ぎ〔用語解説〕　412
データベース複写ユティリティ　271
データベースへのアクセス形態　43
データ有効期間〔一時表〕　115
データ用グローバルバッファ　235
データ用ローカルバッファ　245
データ連携製品との連携　57
データ連動機能　57
データロード〔用語解説〕　412
テープ装置アクセス機能〔用語解説〕　412

テーブルスキャン強制　201
手続き　172, 176
デッドロック　264
デッドロック回避の例　129
デッドロックの例〔インデクスキー値無排他を使用しない場合〕　130
デッドロック〔用語解説〕　412
デファードライト処理　241
デファードライト処理の並列 WRITE 機能　241
デファードライト処理の並列 WRITE 機能〔用語解説〕　413
デファードライト処理〔用語解説〕　413
デファードライトトリガ　241
デフォルトコンストラクタ関数　136
デフォルトコンストラクタ関数〔用語解説〕　413

と

同期グループ〔用語解説〕　413
同期コピー　331
同期コピー〔用語解説〕　413
同期点　255, 270
同期点行数　285
同期点指定の再編成　285
同期点指定の再編成〔用語解説〕　413
同期点指定のデータロード　285
同期点指定のデータロード〔用語解説〕　414
同期点〔用語解説〕　413
同期ペアボリューム〔用語解説〕　414
統計ログファイル〔用語解説〕　414
導出表の条件繰り込み機能　202
動的 SQL〔用語解説〕　414
動的登録　66
動的トランザクションの登録　63
特定データの探索　164
トランザクション欠損なし（データ欠損なし）〔用語解説〕　414
トランザクション固有一時表　114
トランザクション制御　255
トランザクションの移行　63, 260
トランザクションの開始と終了　256
トランザクションマネジャ　62
トランザクションマネジャへの登録　66
トランザクション〔用語解説〕　414
トリガ　186
トリガ SQL 文　186
トリガ契機となる SQL　186
トリガ動作の探索条件　186
トリガ〔用語解説〕　415

な

ナル値〔用語解説〕　415

に

二相コミット　258
日間隔データ　88
認可識別子の指定禁止　354
認可識別子〔用語解説〕　415

ね

ネストループジョイン強制　200
ネストループジョイン優先　200

の

ノースプリットオプション　91
ノースプリットオプション〔用語解説〕　415
ノンストップサービスへの対応　37

は

パーティション　213
倍精度浮動小数点数　88
排他資源　261
排他制御　261
排他制御の期間　264
排他制御の単位　261
排他制御モード　262
排他制御〔用語解説〕　415
排他モード　262
バイナリデータ　88
バイナリデータ列　88

ハイブリッド方式　334
ハイブリッド方式〔用語解説〕　415
配列を使用した DELETE 機能　198
配列を使用した FETCH 機能　197
配列を使用した INSERT 機能　197
配列を使用した UPDATE 機能　198
配列を使用した機能〔用語解説〕　416
パスワードに設定できる制限　353
パスワードの文字列制限　353
パスワードの有効期間　359
パスワード無効アカウントロック状態　354
パスワード無効アカウントロック状態〔用語解説〕416
バックアップ取得時間の短縮〔ユーザ LOB 用 RD エリア〕　279
バックアップ取得時点に回復する場合　270
バックアップ取得モード　272
バックアップ取得モード〔用語解説〕　416
バックアップ〔データベースの更新ログを取得しない運用〕　267
バックアップとアンロードログファイルの関係　275
バックアップの取得　271
バックアップの取得単位　271
バックアップファイル　269
バックアップファイルの格納　272
バックアップファイル〔用語解説〕　416
バックアップ閉塞　278
バックアップ閉塞〔用語解説〕　416
バックエンドサーバ　41
バックエンドサーバ定義　227
バックエンドサーバ〔用語解説〕　417
ハッシュグループ　99
ハッシュジョイン，副問合せのハッシュ実行　202
ハッシュ分割　95
ハッシュ分割表のリバランス機能　98
ハッシュ分割表のリバランス機能〔用語解説〕　417
バッチ高速化機能　56
バッチファイル　37
バッファ障害〔用語解説〕　417
パブリックルーチン　172
パブリックルーチン〔用語解説〕　417
パラメタを含む XMLEXISTS 述語への部分構造インデクスの有効化　202
範囲指定の回復〔用語解説〕　417

ひ

被参照表　189
日立ディスクアレイシステム　329
日付データ　88
非同期 READ 機能　241
非同期 READ 機能〔用語解説〕　418
非同期 XA 呼び出し　64
非同期グループ〔用語解説〕　418
非同期コピー　331
非同期コピー〔用語解説〕　418
非同期ペアボリューム〔用語解説〕　418
非ナル値制約　188
非ナル値制約〔用語解説〕　418
非分割キーインデクス　123
ビュー表　109
ビュー表〔用語解説〕　418
表　87
表単位の再編成　284
表の再編成　283
表の再編成の実行単位　283
表の再編成〔用語解説〕　418
表の正規化　89
表の分割格納条件の変更　104
表のマトリクス分割　100
表の横分割　92
表の横分割の定義　98
表の横分割の例　97
表のリバランス　99

ふ

ファイルグループ〔システムログファイル〕　217
ファイルグループ〔シンクポイントダンプファイル〕　219
フェンスレベル〔用語解説〕　418

付加 PP　50
副シンクポイントダンプファイル〔用語解説〕　419
複数インデクス利用の強制　201
複数接続機能　64, 255
複数接続機能〔用語解説〕　419
複数の SQL オブジェクト作成　200
複数列インデクス　119
複数列分割　123
副ステータスファイル〔用語解説〕　419
副問合せ　165
副問合せ実行方式の SQL 最適化指定　200
プライマリキー　90
プライマリキー〔用語解説〕　405
プラグイン　360
プラグインアーキテクチャ　361
プラグインインデクス　118, 367
プラグインインデクスの遅延一括作成　367
プラグインインデクスの遅延一括作成〔用語解説〕419
プラグインインデクス〔用語解説〕　419
プラグイン提供関数からの一括取得機能　202
プラグインのセットアップ　366
プラグイン〔用語解説〕　419
プラットフォームごとの HiRDB の機能差　372
プリフェッチ機能　240
プリフェッチ機能〔用語解説〕　419
プリプロセス〔用語解説〕　420
プリペア処理　258
フルバックアップ　277
フレキシブルハッシュ分割　96
フレキシブルハッシュ分割〔用語解説〕　420
フロータブルサーバ　41
フロータブルサーバ候補数の拡大　200
フロータブルサーバ対象拡大　200
フロータブルサーバ対象限定　201
フロータブルサーバ〔用語解説〕　420
プロセス間メモリ通信機能〔用語解説〕　420
プロセスの異常終了回数監視機能〔用語解説〕　420
プロセス〔用語解説〕　420
ブロック転送機能　195
ブロック転送機能〔用語解説〕　421
フロントエンドサーバ　41
フロントエンドサーバ定義　227
フロントエンドサーバ〔用語解説〕　421
分割格納条件の変更〔用語解説〕　421
分割キー　93, 95
分割キーインデクス　123
分割キーインデクス〔用語解説〕　421
分岐行〔用語解説〕　423
分散トランザクション処理　62

へ

ペア状態〔用語解説〕　423
ペアステータス〔用語解説〕　423
ペアボリューム〔用語解説〕　423
ペア論理ボリュームグループ〔用語解説〕　424
ペア論理ボリューム〔用語解説〕　423
ページ　148
ページコンパクション　291
ページ単位の排他制御　262
ページ内の未使用領域の比率　155
ページの解放　157
ページの確保　156
ページの状態　155
ページの設計　155
ページ排他　262
ページ〔用語解説〕　424

ほ

ホールダブルカーソル　199
保護モード〔用語解説〕　424
ホスト BES　312
ホスト BES〔用語解説〕　424
ポストソース〔用語解説〕　424
ボリューム属性〔用語解説〕　424

ま

マスタディレクトリ用 RD エリア　82
マスタディレクトリ用 RD エリア〔用語解説〕　425

マトリクス分割　100
マトリクス分割表　100
マトリクス分割〔用語解説〕　425
マルチ HiRDB　42
マルチ HiRDB〔用語解説〕　425
マルチスタンバイ構成　318
マルチフロントエンドサーバ　41
マルチフロントエンドサーバ〔用語解説〕　425
マルチメディア情報を扱う製品との連携　76
満杯セグメント　150

み

未完状態のインデクス　128
未使用セグメント　150
未使用セグメント〔用語解説〕　425
未使用ページ　155
未使用ページ〔用語解説〕　425

め

メインサイト　329
メインサイト〔用語解説〕　425
メッセージキュー監視機能〔用語解説〕　425

も

文字コード〔用語解説〕　426
文字集合〔用語解説〕　426
文字データ　88
モジュールトレース〔用語解説〕　426
モニタモード　315
モニタモード〔用語解説〕　427

ゆ

有効保証世代数　219
ユーザ LOB 用 RD エリア　83
ユーザ LOB 用 RD エリア〔用語解説〕　427
ユーザ権限　340
ユーザサーバホットスタンバイ　323
ユーザサーバホットスタンバイ〔用語解説〕　427
ユーザ識別子〔用語解説〕　427
ユーザ定義型　88
ユーザ定義型〔用語解説〕　427
ユーザ定義関数　176
ユーザ定義のコンストラクタ関数　136
ユーザ用 RD エリア　82
ユーザ用 RD エリア〔用語解説〕　428
ユティリティ専用ユニット　38
ユティリティ専用ユニット〔用語解説〕　428
ユティリティ操作イベント　351
ユニット　38, 40
ユニットコントローラ〔用語解説〕　428
ユニット制御情報定義　226, 227
ユニット〔用語解説〕　428
ユニット用ステータスファイル　220

よ

要素　107
横分割　92, 120
横分割インデクス　120
横分割表　92
横分割〔用語解説〕　428
予測レベル 1〔再編成時期予測機能〕　287
予測レベル 2〔再編成時期予測機能〕　287
予備系　309
予備列〔用語解説〕　428
読み取り専用　63
予約語〔用語解説〕　428

ら

ラージファイル〔用語解説〕　429

り

リアルタイム SAN レプリケーション　329
リアルタイム SAN レプリケーション〔用語解説〕　429
リードレプリカ　429
リーフページ　118
リスト　204
リスト用 RD エリア　83
リスト用 RD エリアの作成　206

リスト用 RD エリア〔用語解説〕 429
リスト〔用語解説〕 429
リストを使用した検索の例 204
リソースマネジャ 62
リバランス機能 98
リバランスユティリティ 99
リモートサイト 329
リモートサイトへのデータ反映方式 332
リモートサイト〔用語解説〕 429
リモート操作ログ〔用語解説〕 430
リロード 283
リンカ〔用語解説〕 430
リンケージ〔用語解説〕 430

る

ルーチン 177
ルートページ 118
ループバックアドレス〔用語解説〕 430

れ

レジストリ LOB 用 RD エリア 83
レジストリ LOB 用 RD エリア〔用語解説〕 430
レジストリ管理表 367
レジストリ機能の初期設定 366
レジストリ機能〔用語解説〕 430
レジストリ情報 83, 366
レジストリ用 RD エリア 83
レジストリ用 RD エリア〔用語解説〕 430
列 87
レプリケーション機能 57
レプリケーション機能に必要な製品 60
レプリケーション機能の適用例 58
レプリケーション機能〔用語解説〕 430
連続認証失敗アカウントロック状態 355
連続認証失敗アカウントロック状態〔用語解説〕 431
連続認証失敗回数の制限 355
連続認証失敗許容回数 355
連動系切り替え 317

ろ

ローカルバッファ 245
ローカルバッファ〔用語解説〕 431
ロードモジュール〔用語解説〕 431
ロール（role）〔用語解説〕 431
ロールバック 256, 270
ロールバック処理 233
ロールバック〔用語解説〕 431
ロールフォワード処理 233
ログ取得モード 265
ログ取得モード〔用語解説〕 431
ログ適用 337
ログ適用可能状態〔用語解説〕 431
ログ適用サイト 336
ログ適用サイト〔用語解説〕 432
ログ適用不可能状態〔用語解説〕 432
ログ適用〔用語解説〕 431
ログ同期方式 336
ログレス閉塞 267
ログレスモード 265
ログレスモード〔用語解説〕 432
論理値 88
論理データ 88
論理ファイル 220